Efficiency and Pseudo-Randomness of a Variant of Zémor-Tillich Hash Function

Christophe Petit, Nicolas Veyrat-Charvillon, Jean-Jacques Quisquater
Université catholique de Louvain
UCL Crypto Group
Bâtiment Mawell, Place du Levant 3
Louvain-la-Neuve, Belgium
christophe.petit,nicolas.veyrat, jjq@uclouvain.be

Abstract

Recent breakthroughs concerning the current standard SHA-1 prompted NIST to launch a competition for a new secure hash algorithm [1,13]. *Provably secure* hash functions (in the sense that their security relates to the hardness of some mathematical problems [5,7,9,12]) are particularly interesting from a theoretical point of view but are often much slower than heuristic functions like SHA.
In this paper, we consider a variant of ZT hash, a provably secure hash function designed by Zémor and Tillich proposed in 1994 [12]. Despite some attack proposals, its security has not been fundamentally challenged to this day. Our function is twice as fast as ZT hash and has enhanced security properties. We propose optimized parameters and algorithms to increase the speed of both hash functions. This makes our function one of the most efficient "provably secure" hash functions to this day. Finally, we show that our hash function successfully passes most pseudo-randomness tests in the Dieharder suite [2].

1 Introduction

Hash functions are widely used in cryptographic applications such as commitment schemes, digital signature schemes, key derivation, message authentication codes or password encryption. Typical properties required for a hash function are:

- nearly uniform output distribution;
- preimage resistance : it must be computationally hard to find a preimage to a given hash value;
- collision resistance : it must be computationally hard to find two messages hashing to the same hash value.

Additionally, a hash function is often required to behave indistinguishably from a random function.

The SHA family has spread since its publication in 1993 as a cryptographic hash standard [3]. However, recently discovered vulnerabilities in SHA-1 [13] prompted NIST to launch a competition for a New Cryptographic Hash Algorithm [1].

NIST competition is stimulating research on hash functions in the cryptographic community and a lot of new schemes have recently been designed and put forward. Particularly interesting from a theoretical point of view, some of these schemes are *provably secure*, in the sense that their security relies on the hardness of some mathematical problems [5, 7, 9, 12]. A good reduction to a simply formulated mathematical challenge facilitates the evaluation process and increases the confidence once the function has resisted first cryptanalytic attempts. However, it also gives the cryptanalyst a lead on how to break the scheme, especially when the mathematical problem involved has not been studied for years or even decades. Indeed, the LPS hash presented in [5] is now completely broken [10, 14].

In this paper, we consider a variant of ZT hash, a provably secure hash function proposed by Zémor and Tillich at CRYPTO'94 [12]. Since 1994, a few attacks have been presented against ZT hash [4, 6, 8, 11]. Although these attacks are either unpractical or targeting particular weak parameters, they still lead to a lack of confidence in the function. The ZT hash has been forgotten for more than ten years, until it recently regained interest due to the LPS hash proposal [5].

Indeed, the design principles of ZT and LPS hashes are very similar. The input messages are decomposed into bits or k-its, and a walk is performed in some regular (Cayley) graph according to the k-its, the hash value being the last vertex reached by the walk. The security of both functions can be expressed both in algebraic form and in terms of some properties of the corresponding graphs. LPS and ZT hashes only differ in the graphs that are used, but as the attacks discovered in [10, 14] exploit structural properties specific to LPS graphs, they do not seem to generalize to ZT hashes. Today, 14 years after its publication, the collision and preimage resistance of the Zémor-Tillich hash function has not been significantly broken.

Besides its positive security properties, ZT hash has two major weaknesses compared to classical hash functions like SHA: first, it is inherently malleable, *i.e.* the hash values of related messages are also related, and second, it is actually not preimage resistant when the images of short messages are considered. The variant we propose is aimed to solve these problems and to be twice as fast.

As observed by Zémor and Tillich in their paper [12], the arithmetic of the ZT hash function operates in a field of characteristic 2, hence it is efficient in both software and hardware. However and to the best of our knowledge, no running times for an actual implementation of ZT hash has been given so far. Here, we present computational optimizations for our variant, including a careful choice of the parameters and appropriate data paths, and give running time results. These optimizations allow us to reduce the computation time to less than 25 times the one for SHA-256. We point out that our optimizations are fully applicable to the original ZT hash that will however be twice as slow.

Cryptographic hash functions have been used for key derivation: assuming it behaves indistinguishably from a random function, a hash function is used to derive many uncorrelated subkeys from a single seed key. Using a malleable hash function like the ZT hash in this application might create serious security threats. In order to certify the use of our variant for this application, we tested its pseudo-randomness using the Dieharder battery of tests [2]. Our variant passes these tests.

This paper is organized as follows. In Sections 2 and 3, we remind Zémor and Tillich hash function and present our variant. In Section 4, we describe an implementation of our construction and the algorithmic tricks we use to improve its efficiency. Section 5 discusses the pseudo-randomness of our construction and Section 6 concludes and discusses work in progress and future improvements of our work.

2 The Zémor-Tillich hash function

We now describe the Zémor-Tillich hash function, following [12]. Let $m = m_1 m_2 ... m_k$ be the bit string representation of the message to be hashed. Let $P_n(X)$ be an irreducible polynomial of degree n and let $\mathbb{F}_{2^n} = \mathbb{F}_2/(P_n(X))$. Let A_0 and A_1 be the following matrices

$$A_0 = \begin{pmatrix} X & 1 \\ 1 & 0 \end{pmatrix} \qquad A_1 = \begin{pmatrix} X & X+1 \\ 1 & 1 \end{pmatrix}.$$

Define the following mapping

$$\begin{array}{rcl} \pi : \{0,1\} & \rightarrow & \{A_0, A_1\} \\ 0 & \rightarrow & A_0 \\ 1 & \rightarrow & A_1. \end{array}$$

The hashcode of m is just the matrix product

$$h_{ZT}(m) := \pi(m_1)\pi(m_2)...\pi(m_k)$$

where the arithmetic is made modulo $P_n(X)$, that is in the field $\mathbb{F}_{2^n}$. For a message m of length k, computing $h_{ZT}(m)$ seems to require k matrix-by-matrix multiplications but multiplying by A_0 or A_1 can actually be done with only a few SHIFT and XOR operations.

The security of ZT hash can be related to both algebraic and graph-theoretical problems that we will not detail here [5, 12]. Existing attacks against the ZT hash function are either unpractical, giving long messages at the price of discrete logarithm computations in $\mathbb{F}_{2^{2n}}$ [8], or targeting particular weak parameters [6, 11]. The largest binary field where discrete logarithms have been computed is $\mathbb{F}_{2^{607}}$, but as the collisions produced in [8] have expected length $2^{\frac{3n}{4}+2}$ we will not consider this attack as practical even for $n \approx 128$. In particular in the rest of the paper we will consider $n = 127, 251, 509, 1021$ and 2039 that are safe with respect to the attacks developed in [6, 11].

Although fundamentally unbroken in the sense of preimage and collision resistance, the Zémor-Tillich hash function has two main security problems: it is malleable in a sense we will explain below, and it is invertible when short messages are hashed. The malleability property directly results from the design strategy of dealing with the message bits one after the other. Given the hash $h(m)$ of an unknown message m, the hash value of any message $x_1||m||x_2$ can easily be computed.

The invertibility for short messages was observed in [11]. When the message size k is smaller than n, no polynomial reduction is performed. The hash value has the form

$$\begin{pmatrix} a_k(X) & b_{k-1}(X) \\ c_{k-1}(X) & d_{k-2}(X) \end{pmatrix} \text{ or } \begin{pmatrix} a_k(X) & b_k(X) \\ c_{k-1}(X) & d_{k-1}(X) \end{pmatrix}$$

depending on whether the last message bit is 0 or 1 (the subscript indices are the degrees of the polynomials). Consequently, an adversary can easily recover the message bits one by one.

For particular polynomials with very sparse coefficients (like the polynomials in Figure 1), the polynomial reductions change very few bits. For such polynomials we observed that the hash values of large messages (up to a few times n-length) with long sequences of bits 0 also have long sequences of bits 0, an artifact that can be seen both as an additional malleability or as invertibility of larger message sets.

None of these weaknesses contradicts the preimage nor the collision resistance property, because the formal definition of preimage resistance only requires the function to be computationally non-invertible on randomly chosen inputs. However, ZT hash can certainly not be used for some important applications of hash functions that require a good random function. The variant of ZT hash that we present now solves the security problems of the original function.

3 A variant of Zémor-Tillich

We propose to use the following hash function:

$$H(m) := h^{vec}_{ZT}(m||\sigma(h^{vec}_{ZT}(m)))$$

where $h^{vec}_{ZT}(m)$ is the concatenation of the entries $(1,1)$ and $(1,2)$ of $h_{ZT}(m)$ with the same parameters n and $P_n(X)$, and $\sigma : \{0,...2^{2n}-1\} \rightarrow \{0,...2^{2n}-1\} : x \rightarrow x \oplus c$. The parameter c is some fixed constant whose bits "look like random"; in our implementations c is the binary representation of pi.

The function $h^{vec}_{ZT}(m)$ corresponds to the first row of $h_{ZT}(m)$. It can be computed one bit at a time like $h_{ZT}(m)$, and each matrix-by-matrix multiplication is now replaced by a vector-by-matrix multiplication that is twice as fast. The function σ is of course very efficient. The second instance of h^{vec}_{ZT}, namely computing $h^{vec}_{ZT}(m||\sigma(h^{vec}_{ZT}(m))) = h^{vec}_{ZT}(m)h_{ZT}(\sigma(h^{vec}_{ZT}(m)))$, only requires $2n$ additional vector-by-matrices multiplications for the bits of $\sigma(h^{vec}_{ZT}(m))$. It is thus negligible for long messages.

Like the original ZT hash, our variant can be related to both algebraic and graph-theoretical problems, which are very close and partially equivalent to the original problems. We now argue on its non-malleability and its preimage resistance even for short messages.

In the Zémor-Tillich hash function, the preimage problem is easy for messages of length $k < n$ because in that case no polynomial reduction is done, an effect amplified for particular polynomials like those of Figure 1. However, the preimage problem for ZT hash is still believed to be hard if the input space is not restricted to some particular message sets. Thanks to the function σ, the bit string that is the input of the second instance of h^{vec}_{ZT} does not belong to a weak preimage set. Consequently, it cannot be inverted and neither can the whole function.

Now, consider a simple malleability issue present in both h_{ZT} and h^{vec}_{ZT}, that is a relation between the hash values of m and $m' = m||0$:

$$\text{if } h^{vec}_{ZT}(m) = h_1||h_2 \text{ then } h^{vec}_{ZT}(m') = (h_1X + h_2)||h_1.$$

Consider $H(m) = h^{vec}_{ZT}(m||\sigma(h^{vec}_{ZT}(m)))$. Although they are strongly correlated, the hash values $h^{vec}_{ZT}(m)$ and $h^{vec}_{ZT}(m')$ will differ in many bits in general, so $\sigma(h^{vec}_{ZT}(m))$ and $\sigma(h^{vec}_{ZT}(m'))$ will differ in many bits and $H(m)$ and $H(m')$ will be completely uncorrelated. Of course some particular values (m, m') such that $h^{vec}_{ZT}(m)$ and $h^{vec}_{ZT}(m')$ are very close, for example differ by only the last bit, but finding such a pair without inverting h^{vec}_{ZT} already seems a hard problem. Moreover, any such m and m' will differ in many bits, so again $H(m)$ and $H(m')$ will be completely uncorrelated.

Considering more elaborated malleability relations involving for example the hash values of more than one message, leads to the same conclusion: the hash function H does not suffer from apparent malleability properties.

4 Efficiency of our hash function

We now show how to efficiently compute our function. We point out that all the optimized parameters and algorithms we give are also applicable to the original ZT hash.

The computational time of our hash function can be characterized as follows

$$t(m) = t_0 * k_0(m) + t_1 * k_1(m)$$

n	Polynomial
127	$X^{127}+X+1$
251	$X^{251}+X^7+X^4+X^2+1$
509	$X^{509}+X^8+X^7+X^3+1$
1021	$X^{1021}+X^5+X^2+X+1$
2039	$X^{2039}+X^{10}+X^9+X^8+X^7+X^5+X^4+X^2+1$

Figure 1: Irreducible polynomials that allow cheap modular reductions

where t_i is the time needed to multiply by A_i and $k_i(m)$ is the number of bits i in $m||\sigma(h_{ZT}^{vec}(m))$. Clearly $t_1 > t_0$: we have

$$\begin{aligned}(a,b)A_0 &= (aX+b,a),\\ (a,b)A_1 &= (aX+b,aX+a+b),\end{aligned}$$

so multiplying by A_0 requires one multiplication by X and one addition in the field $\mathbb{F}_{2^n}$ while multiplying by A_1 requires one addition more.

The arithmetic is in a field of characteristic 2 and is thus very efficient. In our C implementation on a 32-bit processor, we represent an element $a = a_{n-1}X^{n-1} + a_{n-2}X^{n-2} + ...a_1X + a_0$ as an array A of $L := \lceil \frac{n}{32} \rceil$ integers ($L := \lceil \frac{n}{64} \rceil$ in 64-bit architectures). An addition requires only L XORs, and a multiplication a by X requires L SHIFTs and a polynomial modular reduction. The operations $aX+b$ and $aX+a$ can be performed with L SXORs and a modular reduction.

In general a polynomial reduction would cost one TEST and L XORs, but for the well-chosen parameters of Figure 1 we reduce this cost to a TEST instruction and two XORs. For example, reducing by $X^{1021}+X^5+X^2+X+1$ amounts to testing whether $A[31] \wedge 20000000h$ is equal to 0, and if not to compute $A[31] = A[31] \oplus 20000000h$ and $A[0] = A[0] \oplus 27h$. For long messages, the TEST instruction will return 1 half of the times. According to this analysis,

$$\begin{aligned}t_0 &= Lt_{SXOR} + t_{XOR} + t_{TEST} + C_0\\ &\approx Lt_{SXOR}\\ t_1 &= L(t_{XOR}+t_{SXOR}) + t_{XOR} + t_{TEST} + C_1\\ &\approx L(t_{XOR}+t_{SXOR})\end{aligned}$$

where C_0 and C_1 are constant times needed to call the functions.

We obtain additional speedup by grouping the computation of consecutive message bits. Indeed,

$$\begin{aligned}(a,b)A_0A_0 &= (aX^2+bX+a,aX+b),\\ (a,b)A_0A_1 &= (aX^2+bX+a,aX^2+aX+bX+a+b),\\ (a,b)A_1A_0 &= (aX^2+aX+bX+a+b,aX+b),\\ (a,b)A_1A_1 &= (aX^2+aX+bX+a+b,aX^2+bX+a).\end{aligned}$$

The last product can be computed by computing first $aX+b$, then $(aX+b)X+a = aX^2+bX+a$ and finally $(aX^2+bX+a)+(aX+b)$. The cost of this sequence of instructions is at most

$$\begin{aligned}t_{11} &= L(t_{XOR}+2t_{SXOR}) + 2t_{XOR} + 2t_{TEST} + C_{11}\\ &\approx L(t_{XOR}+2t_{SXOR})\end{aligned}$$

n	No grouping	2-bit grouping	4-bit grouping
127	122	114	121
251	159	153	156
509	251	244	238
1021	386	368	388
2039	699	655	686

Figure 2: Running time (seconds) of different algorithms for H with different parameters

SHA	Time
SHA-1	2.5
SHA-256	5.0
SHA-512	23.7

Figure 3: Running time (seconds) for the SHA family

which is about 25% smaller than $2t_1$. We generalized this approach to more matrix groupings with the help of a computer program. More specifically, we wrote a Maple program that

- Computes the vector-by-matrices product;
- Looks for the best data paths with respect to the operations $a \to aX$, $(a, b) \to aX + b$ and $(a, b) \to a + b$;
- Selects the very best data path according to an optimization function including the computation times of individual operations and the number of registers needed in a C implementation;
- Writes a C code computing the products.

The running time results for our function and SHA are shown at Figure 2 and 3. All tests were performed on an 32-bit intel Q6600 quad processor running at 2.4 GHz, with 2Go DDR2 Ram. The OS is Ubuntu running kernel 2.6.24. Test vectors for performance evaluation were 500Mo random files generated using `/dev/urandom`.

The performances of our function should be further improved by using the multimedia sets of instructions available on processors. The computation time is linear in the word size in the processor, so it should be roughly halved on a 64-bit word computer (SHA-512 should also run faster).

5 Pseudorandomness

We now show that our hash function can be used to generate good sequences of pseudorandom numbers. More specifically, we consider the bitstring sequences

$$\begin{aligned} s_1(\mathcal{H}) &= \mathcal{H}(0)||\mathcal{H}(1)||\mathcal{H}(2)||...\mathcal{H}(9999) \\ s_2(\mathcal{H}) &= \mathcal{H}(0)||\mathcal{H}(\mathcal{H}(0))||\mathcal{H}^{(3)}(0)||...\mathcal{H}^{(10000)}(0) \end{aligned}$$

for $\mathcal{H} = h_{ZT}^{vec}$, $\mathcal{H} = H$, and give them as inputs to the Dieharder battery of tests of pseudo-randomness [2].

Results for h_{ZT}^{vec} exhibit a high correlation in the outputs, with a Chi square distribution exceeded only 0.01% of the time by a random value, for all the polynomials of Figure 1.

H seems to behave undistinguishibly from a random stream, as it passes successfully all tests in the Dieharder suite, and exhibits good Chi square distribution results, with values exceeded between 25 and 90% of the time.

6 Conclusion and Further Work

In this paper, we present a new provable cryptographic hash function that is twice as fast as Zémor and Tillich hash function and has enhanced security properties. We proposed optimal parameters and algorithms for both ZT hash and our construction, and analyzed the pseudo-randomness properties of our function. A careful implementation of the hash function allows for performances within a factor 23 of SHA-256 for an equivalent level of security (127 bit polynomial), and only 6.5 for SHA-512 (251 bit polynomial). Our hashing scheme H passes all tests in the ent and Dieharder test suites without exhibiting any particular behavior.

Our function is very efficient (for a "provably secure" hash function) because it uses the arithmetic of a binary field, but it is still slow for large parameters as intermediary results have to be stored in large integer arrays. In a future work, we will investigate how MMX instructions can partially solve this problem in software. We will also implement the function on an FPGA, for which we expect a throughput of $800Mb/s$ for 1024-bit parameters, only one half of the very best SHA-2 implementation. Finally, the provable security aspects of our construction will be presented in an oncoming paper.

Acknowledgment

Part of this work is supported by Belgian Interuniversity Attraction Pole P6/26 BCRYPT. Christophe Petit is a research fellow F.R.S.-F.N.R.S. at UCL Crypto group, Université catholique de Louvain. Nicolas Veyrat-Charvillon is a post-doctoral fellow at UCL Crypto group, Université catholique de Louvain, supported by the Translogistic project from the Walloon region.

References

[1] `http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf`.

[2] Dieharder. http://www.phy.duke.edu/ rgb/General/dieharder.php.

[3] FIPS 180-2 secure hash standard.

[4] K. S. Abdukhalikov and C. Kim. On the security of the hashing scheme based on SL2. In *FSE '98: Proceedings of the 5th International Workshop on Fast Software Encryption*, pages 93–102, London, UK, 1998. Springer-Verlag.

[5] D. X. Charles, E. Z. Goren, and K. E. Lauter. Cryptographic hash functions from expander graphs. To appear in *Journal of Cryptology.*

[6] C. Charnes and J. Pieprzyk. Attacking the SL2 hashing scheme. In *ASIACRYPT '94: Proceedings of the 4th International Conference on the Theory and Applications of Cryptology*, pages 322–330, London, UK, 1995. Springer-Verlag.

[7] S. Contini, A. K. Lenstra, and R. Steinfeld. VSH, an efficient and provable collision-resistant hash function. In S. Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 165–182. Springer, 2006.

[8] W. Geiselmann. A note on the hash function of Tillich and Zémor. In *Fast Software Encryption*, pages 51–52, 1996.

[9] V. Lyubashevsky, D. Micciancio, C. Peikert, and A. Rosen. Provably secure FFT hashing. In *NIST 2nd Cryptogaphic Hash Workshop*, 2006.

[10] C. Petit, K. E. Lauter, and J.-J. Quisquater. Full cryptanalysis of LPS and Morgenstern hash functions. Preprint, 2008.

[11] R. Steinwandt, M. Grassl, W. Geiselmann, and T. Beth. Weaknesses in the $SL_2(\mathbb{F}_{2^n})$ hashing scheme. In *Proceedings of Advances in Cryptology - CRYPTO 2000: 20th Annual International Cryptology Conference*, 2000.

[12] J.-P. Tillich and G. Zémor. Hashing with SL_2. In Y. Desmedt, editor, *CRYPTO*, volume 839 of *Lecture Notes in Computer Science*, pages 40–49. Springer, 1994.

[13] X. Wang, Y. L. Yin, and H. Yu. Finding collisions in the full SHA-1. In V. Shoup, editor, *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 17–36. Springer, 2005.

[14] G. Zémor and J.-P. Tillich. Collisions for the LPS expander graph hash function. To appear in the proceedings of *Advances in Cryptology - EUROCRYPT 2008*, 2008.