Cayley Hashes: A Class of Efficient Graph-based Hash Functions

Christophe Petit[1⋆], Kristin Lauter[2] and Jean-Jacques Quisquater[1]

[1]UCL Crypto Group, [2]Microsoft Research.

e-mails: christophe.petit@uclouvain.be,klauter@microsoft.com,jjq@uclouvain.be

Abstract Hash functions are widely used in cryptography. Recent breakthroughs against the standard SHA-1 prompted NIST to launch a competition for a new secure hash algorithm, SHA-3 [1]. *Provably secure* hash functions, that is functions whose security reduces to a simply-stated, supposedly hard mathematical problem, are widely believed to be much too slow for the NIST competition.

In this paper, we discuss Cayley hashes, a class of efficient and provably secure hash functions constructed from the Cayley graphs of (projective) linear groups. We review two existing constructions, the ZT and LPS hash functions, and put a new one forward, the Morgenstern hash function. We show that Cayley hashes are "provable" *and* efficient: on one hand, their security reduces to a representation problem in (projective) linear groups; on the other hand, they are only 5 times slower than SHA-2 in FPGA hardware, and about 400 times slower in software (in our future implementations, many optimizations currently under investigation are expected to decrease these gaps even more). Last but not least, Cayley hash computation can be easily parallelized. We believe their nice properties as well as their elegant design make Cayley hashes very interesting hash functions.

1 Introduction

Hash functions are widely used in cryptographic applications such as commitment schemes, digital signatures schemes, message authentication codes or password encryption. Typically, a hash function is required to be preimage and collision resistant and have nearly uniform output distribution. Due to the importance of cryptographic hash functions, the SHA family was designed as a standard [2]. However, recently discovered vulnerabilities in SHA-1 [23] prompted NIST to launch a competition for a New Cryptographic Hash Algorithm [1].

Practical vs. provable hashes Most current hash functions are made of a compression function and of a transform. The compression function processes a fixed number of bits and returns a fixed smaller number of bits. The transform uses the compression function iteratively to produce a fixed-length output from a sequence of arbitrary length. A good transform is supposed to be at least preimage, collision and second-preimage resistant if the compression function has the same properties.

⋆ Research Fellow of the Belgian Fund for Scientific Research (F.R.S.-FNRS).

In practice, the compression function is often instantiated by a block cipher, that is, its security is simply assumed, not related to the hardness of some mathematical problem. Noticed exceptions to this practice are Lyubashevsky *et al.* FFT Hashing [17], Contini *et al.* VSH [9], and Charles *et al.* hashes based on Pizer's and LPS' Ramanujan graphs [5]. The security of these hashes respectively reduces to a lattice problem, a variant of the factorization problem, finding isogenies between elliptic curves, and a representation problem in a projective linear group.

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, the price for a mathematical proof has so far always been paid in efficiency. Indeed, constructing an efficient and provable hash function seems a big challenge nowadays.

Cayley hashes In this paper, we examine three satisfactory solutions to this problem: the ZT hash function, which was designed in the past by Zémor and Tillich [22], the LPS hash function, whose efficiency we greatly enhance, and the Morgenstern hash function, which is new. The advantage of LPS hash over ZT hash is that it is based on a Ramanujan graph family; on the other hand it involves operations in a large prime field rather than a field of even characteristic, hence an efficiency loss. Our new proposal, based on Morgenstern's Ramanujan graphs, combines the advantages of both. We group the three hash functions under the name "Cayley hashes" because they are based on Cayley graphs of certain (projective) special linear groups.

The output distributions of Cayley hashes tend to uniformity as the message length increases, while collision and preimage resistance are equivalent to the hardness of simply-stated representation problems. Cayley hashes return a fixed length digest from an arbitrary-length input message, that is, no additional transform is needed to process large messages. Like RSA for example, they can be easily defined for any security level. Constructing keyed Cayley hash functions can be done in a very simple and natural way. The main drawback of Cayley hashes (at least in some applications) is their "malleability" so in Section 6, we present a heuristic way to remove this property at the expense of at most half the efficiency.

We also show how to make Cayley hashes much faster than stated in Charles *et al.* paper [5]. Indeed, they are much faster than what would be expected from a provable hash. In our very first implementations Cayley hashes are only 5 times slower than SHA-256 in FPGA hardware, and only about 400 times slower than SHA-256 in software. We are currently investigating many ideas that together could even accelerate these functions by a factor 10. Finally, we point out that Cayley hashes are fully parallelizable, so their computation can be made even much faster, especially in hardware.

Outline of the paper This paper is organized as follows: in Section 2 we present general and particular constructions of Cayley hashes, Section 3 reviews their security properties and Section 4 deals with the representation problem. Section 5 sketches efficient algorithms for computing them as well as our first implementation results. Section 6 presents current research and open problems and Section 7 concludes the paper.

2 Cayley Hashes

2.1 General presentation

Cryptographic Hash Functions. A *hash function* or *hash* $H : \mathcal{M} \to \mathcal{H}$ transforms an input of a large size (or of arbitrary size) into an output of small, fixed size. The elements of the domain $\mathcal{M}$ are called *messages* and the elements of the codomain $\mathcal{H}$ are called *hashes*. Properties required for hash functions are as wide as their applications:

- The **preimage resistance** roughly requires that given the hash of a randomly chosen message, it is computationally hard to recover any message with that hash.
- The **second preimage resistance** requires that given a randomly chosen message, it is computationally hard to find a second message with the same hash.
- The **collision resistance** requires that it is computationally hard to find two messages with the same hash.
- A **universal hash function** has nearly uniformly distributed outputs.
- A **universal one-way hash function** roughly satisfies preimage resistance and uniform output distribution.
- A **random oracle** is an abstraction for a "perfectly random" function.

We refer to [12,14] for formal definitions of those properties.

Expander graphs. Let $\mathcal{G} = (V, E)$ be a graph with vertex set V and edge set $E \subset V \times V$. The *directed girth* g of $\mathcal{G}$ is the largest g such that given any two vertices v_1 and v_2 of V, any pair of distinct directed paths joining v_1 to v_2 will be such that one of those paths has length at least g. (If $\mathcal{G}$ is undirected, the *girth* is the smallest cycle in the graph.) The *expanding constant* h of $\mathcal{G}$ is defined as $h = \min_{1 \leq |S| \leq \frac{|V|}{2}} \frac{|\delta(S)|}{|S|}$ where S is any subset of V and $\delta(S)$ is the set of edges in E connecting S to $V \setminus S$.

In this paper, we are focussing on regular *expander* graphs, that is, regular graphs with a large expanding constant. The eigenvalues of the adjacency matrix of a k-regular graph with $|V|$ vertices satisfy $\lambda_0 = k \geq \lambda_1 \geq ... \geq \lambda_{|V|-1} \geq -k$, and the double bound $\frac{k-\lambda_1}{2} \leq h \leq \sqrt{2k(k-\lambda_1)}$ relates h to λ_1 [4,10]. A bound of Alon-Boppana says that while $|V|$ goes to infinity, λ_1 cannot be too small, more precisely, $\liminf_{|V| \to +\infty} \lambda_1 \geq 2\sqrt{k-1}$, which implies that h cannot be too large. A k-regular *Ramanujan family* of graphs is a family of k-regular graphs with increasing number of vertices, such that $\liminf_{|V| \to +\infty} \lambda_1 = 2\sqrt{k-1}$. In that sense, Ramanujan graphs are graphs with asymptotically optimal expansion properties. We refer to the recent survey [13] for more details on expander graphs.

From Cayley Graphs to Cayley Hashes. A *Cayley graph* $\mathcal{C}_{G,S} = (V, E)$ is a graph constructed from a group G and a subset S of G as follows: V contains a vertex v_g associated to each element $g \in G$, and E contains the directed edge (v_{g_1}, v_{g_2}) iff there is some $s \in S$ such that $g_2 = g_1 s$. For a set S of size k, the Cayley graph $\mathcal{C}_{G,S}$ is a k-regular graph. In this paper, we consider the Cayley graphs of (projective) special

linear groups, which are known to display good expanding properties [21,15,19], or even to be Ramanujan [16,8,18].

To construct a *Cayley hash* from a directed Cayley graph, let $\{1,...,k\}^*$ be the set of arbitrary-length sequences $(m_1, m_2, m_3, ...m_l)$ of elements of $\{1,...,k\}$. Fixing an initial value $g_0 = 1$ in G and an ordering $\sigma : \{1,...,k\} \to S$, determines a Cayley hash function $H : \{1,...,k\}^* \to G$ defined by $H() = g_0$, $H(m_1) = g_0\sigma(m_1)$ and $H(m_1, m_2, ..., m_l) = H(m_1, m_2, ..., m_{l-1})\sigma(m_l)$. (Note that the successive computations of $g_0\sigma(m_1)$, $g_0\sigma(m_1)\sigma(m_2)$, *etc.* correspond to a walk from v_{g_0} to $v_{g_0\sigma(m_1)}$, $g_0 v_{\sigma(m_1)\sigma(m_2)}$, *etc.* in the Cayley graph $\mathcal{C}_{G,S}$.) If S is stable under inversion, the corresponding Cayley graph $\mathcal{C}_{G,S}$ is undirected. To avoid backtracking (that would lead to trivial collisions), the messages are decomposed in $(k-1)$-its rather than k-its, and H is defined as $H(m_1, m_2, m_3, ..., m_l) = H(m_1, m_2, m_3, ..., m_{l-1})\sigma_l$ with $\sigma_1 = \sigma(m_1)$ and $\sigma_i = \Sigma(\sigma_{i-1}, m_i) := \sigma\left(\sigma^{-1}(\sigma_{i-1}^{-1}) + m_i \bmod k\right), i \in \{2,...,l\}$.

A hash function $H' : \{0,1\}^* \to \{0,1\}^{\log_2 |G|}$ sending bitstrings to bitstrings is derived from H as $H' = \pi_2 \circ H \circ \pi_1$, where $\pi_1 : \{0,1\}^* \to \{1,...,k\}^*$ and $\pi_2 : G \to \{0,1\}^{\log_2 |G|}$ are respectively initial and final mappings. A *keyed* Cayley hash is constructed similarly, by letting the element g_0 be a function of the key.

2.2 ZT and LPS hashes

ZT hash function. The ZT hash was designed in the past by Zémor and Tillich [21,22,24], but without much care to the parameters. Rapidly, a number of attacks were discovered [6,20,3] that exploited this lack of specificity and lead to a lack of confidence in the function even if it remains essentially unbroken.

The definition is as follows: Let $P_n(X)$ be an irreducible polynomial of degree n in $\mathbb{F}_2[X]$, and let $K = \mathbb{F}_2[X]/(P_n(X))$. Let $G = SL(2,K) \cong PSL(2,K)$, $S = \left\{A_0 = \begin{pmatrix} X & 1 \\ 1 & 0 \end{pmatrix}, A_1 = \begin{pmatrix} X & X+1 \\ 1 & 1 \end{pmatrix}\right\}$, and $g_0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$. ZT hash is the Cayley hash corresponding to $\mathcal{C}_{G,S}$ with starting point g_0.

Originally, Zémor and Tillich only imposed the degree n of the polynomial $P_n(X)$ to be in the range $130 - 170$. In our specifications, we will be much more restrictive in the light of existing attacks (see Section 4). We require n to be a prime close to 1024, such that the factorization of $2^{2n} - 1$ involves at least one prime of size larger than 80 bits (ideally $2^{2n} - 1 = 3p_1p_2$ for some primes p_1, p_2). Moreover, trapdoor attacks are avoided if $P_n(X)$ is defined in a deterministic and clearly honest way, for example as the smallest irreducible polynomial of degree n larger than the binary representation of π. The output of the new ZT hash function is about 3072 bits.

LPS hash function LPS hashes were introduced by Charles, Goren and Lauter [5]. They make use of undirected Ramanujan graphs independently discovered by Lubotzky, Phillips and Sarnak, and by Margulis [16].

Let p and l be primes, l small and p large, both p and l equal to 1 mod 4, and l being a quadratic residue modulo p. To l and p is associated an LPS graph $X_{l,p}$ as follows. Let $\mathbf{i}$ be an integer such that $\mathbf{i}^2 = -1 \bmod p$. The vertices of $X_{l,p}$ are

matrices in $G = PSL(2, \mathbb{F}_p)$. The set S is taken as $S = \{G_j, j = 1..., l+1\}$, where

$$G_j = \begin{pmatrix} g_{0,j} + \mathbf{i}g_{1,j} & g_{2,j} + \mathbf{i}g_{3,j} \\ -g_{2,j} + \mathbf{i}g_{3,j} & g_{0,j} - \mathbf{i}g_{1,j} \end{pmatrix}, \qquad j = 1, ..., l+1;$$

where $(g_{0,j}, g_{1,j}, g_{2,j}, g_{3,j})$ are all the solutions of $g_0^2 + g_1^2 + g_2^2 + g_3^2 = l$, with $g_0 > 0$ and g_1, g_2, g_3 even. Note that S is stable under inversion, so the Cayley graph $\mathcal{C}_{G,S}$ is undirected. The LPS hash function is the Cayley hash associated to $\mathcal{C}_{G,S}$, starting at the identity.

Charles, Goren and Lauter recommend the use of a prime p of 1024 bits, and a short prime l. As additional constraint, we stress the necessity that the factorization of p^2-1 involves at least one prime of size larger than 80 bits (ideally $p^2-1 = 24p_1p_2$ for large primes p_1 and p_2). The prime p should be fixed in some deterministic, clearly honest way. The output of the hash function is about 3072 bits.

2.3 Morgenstern hashes

Morgenstern's Ramanujan graphs [18] generalize LPS graphs from an odd prime $p \equiv 1 \bmod 4$ to any power of prime q. We suggest to use his graphs with $q = 2^k$ for a hash function.

Let q be a power of 2 and $f(X) = X^2 + X + \epsilon$ irreducible in $\mathbb{F}_q[X]$. Let $g(X) \in \mathbb{F}_q[X]$ be irreducible of even degree $n = 2d$ and let $\mathbb{F}_{q^n} \approx \mathbb{F}_q[X]/(g(X))$. The vertices of the Morgenstern graph Γ_q are elements of $G = PSL_2(\mathbb{F}_{q^n})$. Let $\mathbf{i} \in \mathbb{F}_{q^n}$ be a root of $f(X)$. The set S is taken as $S = \{G_j, j = 1..., q+1\}$, where

$$G_j = \begin{pmatrix} 1 & \gamma_j + \delta_j\mathbf{i} \\ (\gamma_j + \delta_j\mathbf{i} + \delta_j)X & 1 \end{pmatrix}, \qquad j = 1, ..., q+1;$$

where $\gamma_j, \delta_j \in \mathbb{F}_q$ are all the $q+1$ solutions in $\mathbb{F}_q$ for $\gamma_j^2 + \gamma_j\delta_j + \delta_j^2\epsilon = 1$. Note that, as each of the G_j has order 2, the Cayley graph $\mathcal{C}_{G,S}$ is undirected. Morgenstern hash function is the Cayley hash associated to $\mathcal{C}_{G,S}$, starting at the identity.

Taking $q = 2$ is particularly attractive for efficiency reasons. We also recommend to choose $n = 2d \approx 1024$, with d prime such that the factorization of $2^{4d} - 1$ involves at least one prime of size larger than 80 bits (ideally $2^d - 1 = 3p_1$, $2^d + 1 = p_2$ and $2^{2d} + 1 = p_3$, with p_1, p_2, p_3 primes). Again, the polynomial $g(X)$ should be fixed in a deterministic, clearly honest way. The output of the hash function is about 3072 bits.

3 Security of Cayley Hashes

Cayley hashes enjoy important security properties:

- Any two colliding messages differ by at least g bits (or k-its), where g is the (directed) girth of the corresponding graph. This results directly from the definitions. From [21,16,18], the girths of ZT, LPS and Morgenstern graphs (with parameter size of 1024 bits) are respectively larger than about 1024, 882 and 1024.

– The outputs tend to be uniformly distributed as the message set becomes larger, and the convergence rate is fast. This comes from the good expanding properties of ZT, LPS and Morgenstern graphs, and results easily from the mixing properties of random walks on expander graphs [13].
– Differential cryptanalysis (DC), which has been the most successful approach against SHA-1, does not seem to apply to Cayley hashes. Indeed, DC typically activates various portions of the message simultaneously, while in Cayley hashes the bits (or k-its) are processed one at the time.
– Collision resistance is equivalent to the hardness of a simply-stated representation problem in the corresponding linear groups: see Problems 1, 2 and Proposition 1 below.
– Preimage resistance and second preimage resistance follow from collision resistance and Proposition 2 below.

Problem 1 *Let H be any of the Cayley hash functions defined in Sections 2.2 or 2.3 with recommended parameters. Let $S = \{G_1, ...G_k\} \subset G$ the generators for H. Find a product (in* reduced form*)*

$$\prod_{1 \leq i \leq N} G_{\theta(i)}^{e_i} = 1$$

where e_i are integers, $\theta : \{1, ...N\} \to \{1...k\}$ and $\sum e_i$ is logarithmic in the size of G. By reduced form*, we mean that for each i, $G_{\theta(i+1)} \neq G_{\theta(i)}, G_{\theta(i)}^{-1}$.*

Problem 2 *Same as Problem 1, but the e_i are additionally required to be positive integers.*

Proposition 1 *A solution to Problem 2 where H is ZT, LPS or Morgenstern hash implies a collision under H, which in turn implies a solution to Problem 1 for H. For LPS and Morgenstern's hashes, Problem 1, Problem 2 and finding collisions are all as hard.*

PROOF: Similar as the proof in [5].

Proposition 2 *Any collision-resistant hash with arbitrary input size is (a) second preimage resistant and (b) preimage resistant.*

PROOF: This is a classical result. Let H be a collision-resistant hash.

(a) Suppose we have an algorithm A that, upon input of a message m, is able to produce a second message m' such that $H(m) = H(m')$ with non negligible probability. Define an algorithm B that generates a random input m, gives it to A, receives m' from A and returns (m, m'). Algorithm B produces collision with the same probability as A produces second-preimage.

(b) Suppose we have an algorithm A that, upon input the hash of a message $h = H(m)$, is able to produce a preimage m' such that $H(m') = h$ with non-negligible probability. Define an algorithm B that, until it finds a collision (m, m'), generates a random input m, gives $h = H(m)$ to A, and upon answer m' from A checks if $m = m'$. As the input set is larger than the output set, A will eventually produce $m' \neq m$, and B will return the collision (m, m').□

4 Hardness of the Representation Problem

In this section, we give arguments for the hardness of the representation problem Problem 2 in the light of known attacks against ZT hash function. These attacks may be classified as follows:

- Density attacks [24].
- Subgroup attacks [6,20].
- Geiselmann's attack involving discrete logarithm computations [11].
- Trapdoor attacks [20].

The representation problem has certainly not been studied as much as say, the discrete logarithm problem or the factorization problem. However, since the publication of ZT hash in CRYPTO'94, all the attacks against it concern particular weak parameters, or involve partial exhaustive searches or the computation of discrete logarithms. The only exception is the trapdoor attacks, but trapdoors can be avoided easily by standard methods.

4.1 Density attacks

An earlier version of the ZT scheme, proposed by Zémor in [24], was based on the Cayley graph of $SL(2,\mathbb{F}_p)$ with generators $S = \left\{ S_0 = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, S_1 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \right\}$. The scheme was shown to be subject to a so-called *density attack* and replaced by Zémor and Tillich for the ZT function of Section 2.2. The density attack lifts the representation problem form $\mathbb{Z}_p$ to the integers, where it can be solved using the Euclidean algorithm. The density attack chooses a matrix $U \in SL(2,\mathbb{Z})$ that reduces to the identity modulo p, and then find its factorization as products of S_0 and S_1. The attack worked well for this scheme because any positive matrix of $SL(2,\mathbb{Z})$ can be factorized as products of S_0 and S_1, and because U can be adequately chosen such that its factorization will be small [22].

A density attack on the ZT hash would require lifting the factorization problem onto $SL(2,\mathbb{F}_2[X])$, that is finding a matrix $U \in SL(2,\mathbb{F}_2[X])$ equal to the identity modulo $P_n(X)$, and if possible, expressing U as a product of Zémor-Tillich generators $A_0 = \begin{pmatrix} X & 1 \\ 1 & 0 \end{pmatrix}$ and $A_1 = \begin{pmatrix} X & X+1 \\ 1 & 1 \end{pmatrix}$. The attack does not work for ZT scheme because a random matrix $U \in SL(2,\mathbb{F}_2[X])$ has a very small probability to be a product of A_0 and A_1, that is, the set generated by A_0 and A_1 is not *dense* enough in $SL(2,\mathbb{F}_2[X])$ [22].

A density attack on LPS hash would require lifting the factorization problem onto $SL(2,\mathbb{Z})$. Each generator G_j of Section 2.2 would be divided by a square root of l (to have determinant 1), then lifted to the integers. *A priori*, each lifted matrix $\widetilde{G_j}$ has entries of size about the size of p, that is, they are large integers. If all the lifts chosen are positive matrices, then their (unique) factorizations with positive

exponents $\widetilde{G_j} = S_0^{q_{j,1}} S_1^{q_{j,2}} S_0^{q_{j,3}} ... S_1^{q_{j,m}}$ clearly show that the space generated by the $\widetilde{G_j}$ is a very small subset of the set of positive matrices of $SL(2, \mathbb{Z})$.

Similarly, a density attack on Morgenstern hash would require lifting the factorization problem onto $SL(2, \mathbb{F}_2[X])$. Each generator G_j of Section 2.3 would be divided by a square root of $X + 1$ (to have determinant 1), then lifted to matrices $\widetilde{G_j} \in SL(2, \mathbb{F}_2[X])$. Again, as the entries of the $\widetilde{G_j}$ are expected to have large degree, the subset of $SL(2, \mathbb{F}_2[X])$ generated by the lifts will be very small.

The arguments above do not totally discard density attacks on Cayley hashes. The matrix U reducing to identity could maybe be chosen better than randomly. Other lifts of the generators could be considered, including more than one lift per generator and (in LPS hash) lifts with negative coefficients. We have tried these approaches but the ZT, LPS and Morgenstern hash seem resistant to it.

4.2 Subgroup attacks

Two subgroup attacks Looking at the ZT hash function, it is easy to derive short relations like $A_1 A_0^{-1} A_1 = A_0$ involving A_0, A_1 and A_0^{-1}, which hold for any polynomial $P_n(X)$. In the first attack paper against ZT hash, Charnes and Pieprzyk [6] derive conditions for particular parameters $P_n(X)$ making the order r_0 of A_0 small, resulting in a short collision $A_1 A_0^{r_0 - 1} A_1 = A_0$.

At CRYPTO 2000, Steinwandt, Grassl, Geiselmann and Beth considerably restrict the set of parameters suitable for Zémor-Tillich hash [20]. They show that if n is not prime, and if $P_n(X)$ can be expressed as a functional decomposition $P_n(X) = g(h(X))$, $\deg(g) = n_1 < n$, $deg(h) = n_2 = n/n_1$, then a subgroup attack may be practical. The idea is that a bit sequence whose hash is a matrix M of small order, may always be repeated to give a collision with the void message. In particular, the matrices in the subgroup $SL(2, \mathbb{F}_{2^{n_1}})$ of $SL(2, \mathbb{F}_{2^n})$, or in one of its conjugate subgroups, have an order smaller than $2^{n_1} + 1$, and as shown by Steinwandt et al., these matrices are exactly the ones whose trace is an element of $\mathbb{F}_{2^{n_1}}$. The attack then works as follows: 2^{n_2} hash matrices are successively computed; the trace of each matrix is evaluated; it is checked whether the trace is in $\mathbb{F}_{2^{n_1}}$; and when such a matrix is found, a collision with the void message is derived.

Subgroup attacks show that the representation problem may be easy for particular parameters. The attacks exploit a lack of specification in Zémor-Tillich's paper, where no restriction was put on the irreducible polynomial $P_n(X)$ but on the approximate size of its degree. Indeed, Abdukhalinov and Kim [3] showed that the Charnes-Pieprzyk conditions are very unlikely (with probability less than 10^{-27}) to be satisfied for randomly chosen parameter $P_n(X)$, so this attack will be discarded by fixing the polynomial in some way. As Steinwandt *et al.* attack comprises an (exponentially long) exhaustive search, the increase of n from $130 - 170$ to about 1024 protects ZT hash against it.

Still, we can worry about other, more sophisticated subgroup attacks. Indeed, suppose that there is a sequence of subgroups $\{Id\} \subset G_1 \subset G_2 ... \subset G_k = G$ where

G is the group associated to ZT, LPS or Morgenstern hash and that the membership of a matrix in one of the G_i can be easily tested. An attack could try to recursively construct a small set S_{k-1} of elements of G_{k-1} from the elements of S, a small subset S_{k-2} of G_{k-2} from the elements of S_{k-1}, etc. If all the ratios $|G_i|/|G_{i-1}|$ are small enough the sets S_i may be constructed by exhaustive searches, and a representation of identity of length smaller than $k \max_i |G_i|/|G_{i-1}|$ can be derived.

The order of $|PSL(2, \mathbb{F}_{p^n})|$ is $\frac{p^n(p^{2n}-1)}{\gcd(p-1,2)}$ and 80 bits is currently believed to be out of reach by exhaustive search methods. The conditions we put in Section 2 on the parameters of ZT, LPS and Morgenstern hashes therefore prevent possible subgroups attacks.

4.3 Geiselmann's attack

The key of Geiselmann's attack is the following proposition, applied at $A_0 = \begin{pmatrix} X & 1 \\ 1 & 0 \end{pmatrix}$ and $A_1 = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} X+1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$.

Proposition 3 [11] *Let* $A = \begin{pmatrix} \alpha & 1 \\ 1 & 0 \end{pmatrix} \in SL(2, \mathbb{F}_{2^n})$ *and* $\widehat{A}(X) := X^2 + \alpha X + 1 \in \mathbb{F}_{2^n}[X]$.

- *If* $\widehat{A}(X)$ *is irreducible, a matrix* $M \in \mathbb{F}_{2^n}^{2\times 2}$ *is a power of* A *if and only if* $M = \lambda A + \mu I$ *for some* $\lambda, \mu \in \mathbb{F}_{2^n}$, $\det(M) = 1$ *and* $M^{ord(A)} = I$.
- *If* $\widehat{A}(X)$ *factorizes into* $\widehat{A}(X) = (X+\beta)(X+\beta^{-1})$, *a matrix* $M \in \mathbb{F}_{2^n}^{2\times 2}$ *is a power of* A *if and only if* $M = \lambda A + \mu I$ *for some* $\lambda, \mu \in \mathbb{F}_{2^n}$, $\det(M) = 1$ *and* $(\lambda\beta + \mu)^{ord(A)} = 1$.

The attack works in three steps:

1. A matrix $C \in SL_2(\mathbb{F}_{2^n})$ is chosen and a matrix equation of the form $A_0^{e_1} A_1^{e_2} ... A_0^{e_l} = (\lambda_1 A_0 + \mu_1 I)(\lambda_2 A_1 + \mu_2 I)...(\lambda_l A_0 + \mu_l I) = C$ is considered, with $l = 3$ or 4.
2. The matrix equation gives four polynomial equations with unknown $\mu_i, \lambda_i \in \mathbb{F}_{2^n}, i = 1...l$. After adding three or four equations for the conditions on the determinants the system is solved. In general, it has solutions.
3. Conditions with the orders are checked. If they are not fulfilled, another matrix C is selected, otherwise the exponents e_i are recovered as the discrete logarithms of the matrices $\lambda_1 A_0 + \mu_1 I$, $\lambda_2 A_1 + \mu_2 I$, *etc.*

Similar attacks on LPS and Morgenstern hashes Geiselman's attack can easily be adapted for LPS and Morgenstern hash. Here, we develop the attack on LPS hash with $l = 5$. Let $\omega = \frac{1-2\mathbf{i}}{1+2\mathbf{i}}$ and $\Lambda = \begin{pmatrix} 1 & 0 \\ 0 & \omega \end{pmatrix} \in PSL(2, \mathbb{F}_p)$. The matrix equation

$G_{i_1}^{e_1} G_{i_2}^{e_2} ... G_{i_l}^{e_l} = C$ translates to a system of four equations with unknowns x_0, $x_1 = \omega^{e_1}$, $x_2 = \omega^{e_2}$,... $x_l = \omega^{e_l}$. Indeed, the generators G_j factorize as

$$
\begin{aligned}
G_1 &= \begin{pmatrix} 1+2\mathbf{i} & 0 \\ 0 & 1-2\mathbf{i} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & \omega \end{pmatrix} = \Lambda, \\
G_{-1} &= \begin{pmatrix} 1-2\mathbf{i} & 0 \\ 0 & 1+2\mathbf{i} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & \omega^{-1} \end{pmatrix} = \Lambda^{-1}, \\
G_2 &= \begin{pmatrix} 1 & 2 \\ -2 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ \mathbf{i} & -\mathbf{i} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \omega \end{pmatrix} \begin{pmatrix} 1 & -\mathbf{i} \\ 1 & \mathbf{i} \end{pmatrix} := P_2^{-1} \Lambda P_2, \\
G_{-2} &= \begin{pmatrix} 1 & -2 \\ 2 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ \mathbf{i} & -\mathbf{i} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \omega^{-1} \end{pmatrix} \begin{pmatrix} 1 & -\mathbf{i} \\ 1 & \mathbf{i} \end{pmatrix} := P_2^{-1} \Lambda^{-1} P_2, \\
G_3 &= \begin{pmatrix} 1 & 2\mathbf{i} \\ 2\mathbf{i} & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \omega \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} := P_3^{-1} \Lambda P_3, \\
G_{-3} &= \begin{pmatrix} 1 & -2\mathbf{i} \\ -2\mathbf{i} & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \omega^{-1} \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} := P_3^{-1} \Lambda^{-1} P_3.
\end{aligned}
$$

As a consequence, any power $G_{i_j}^{e_i}$ writes down as $P_{i_j}^{-1} \Lambda^{e_i} P_{i_j}$ and the entries of $M = G_{i_1}^{e_1} G_{i_2}^{e_2} ... G_{i_l}^{e_l}$ are easily written as polynomials in x_1, x_2,... x_l. Finally, four equations $M_{a,b} = x_0 C_{a,b}$ with $a, b = 1, 2$ are deduced from the system $G_{i_1}^{e_1} G_{i_2}^{e_2} ... G_{i_l}^{e_l} = C$.

Practicability The size of a collision produced by Geiselmann's attack is about lE where E is an upper bound on the exponents e_i. Assuming in a first approximation that the products $A_0^{e_1} A_1^{e_2} ... A_0^{e_l}$ with $e_1, e_2, ...e_l \leq E$ are all distinct in $SL(2, \mathbb{F}_{2^n})$, the expected number of matrices C to be tested before getting one preimage is about $2^{3n}/E^l$ so E must be about $2^{3n/l}$ and the collision size will be about $l2^{3n/l}$. With the new parameters $n \approx 1024$ we recommend for the ZT hash, not only the collision would be much too large but the attack would require solving discrete logarithm in the field $\mathbb{F}_{2^{2048}}$, which may be assumed to be a hard problem from Cryptographic practice.

To avoid the discrete logarithm computation, the following strategy can be considered: recover e_i only if it is smaller than a bound E, which can be done by an exhaustive precomputation of matrices A_0^e and A_1^e, $e = 1, 2, ...E$. In accordance with the above arguments, l should be chosen as $l \approx \frac{3n}{\log_2 E}$. Considering the limit of precomputation power to be $E = 2^{40}$, $n \approx 1024$ gives $l \approx 25$. The drawback with this approach is that it increases the number of unknowns without adding equations to the system to be solved. As a consequence, the second step of Geiselmann attack would give a lot of useless solutions and become very difficult. Moreover, the best methods known to solve general polynomial systems are Groëbner basis algorithms, which are exponentially slow in the number of variables.

4.4 Trapdoor attacks

A trapdoor is a weakness that is intentionally introduced on a system, which allows the person who knows it to break the algorithm much faster. Steinwandt et al. [20] showed how to choose Zémor-Tillich parameters to make any sufficiently long given message $m = (m_1, m_2 ... m_l)$ collide with the void message. Namely, the product $H(m) = A_{m_1} A_{m_2} ... A_{m_l} := \begin{pmatrix} p_1(X)\ p_2(X) \\ p_3(X)\ p_4(X) \end{pmatrix}$ is computed in $SL(2, \mathbb{Z})$ and $P_n(X)$ is chosen as an irreducible polynomial dividing $\gcd(p_1(X)-1, p_2(X), p_3(X), p_4(X)-1)$.

Similar trapdoors can be constructed for LPS and Morgenstern hashes. The possible existence of trapdoors is an actual concern in cryptographic applications because people would need to trust the person who chooses the parameters that she did not cheat. However, as we already said trapdoors can be avoided by standard techniques, for example by including a universal constant like π in the definition of the parameters.

5 Efficiency

Speeding up the computation The main part of the Cayley hash computations are the matrix products (one per step), especially as the message lengths increase. A matrix product usually costs 8 field multiplications and 4 field additions, but the special form of our Cayley generators allows removing all the field multiplications. In ZT hash, multiplying a generic matrix $M = \begin{pmatrix} a\ b \\ c\ d \end{pmatrix}$ by $A_0 = \begin{pmatrix} X\ 1 \\ 1\ 0 \end{pmatrix}$ or $A_1 = \begin{pmatrix} X\ X+1 \\ 1\ \ 1 \end{pmatrix}$ only requires 2 one-bit shifts and at most 6 XOR operations on 1024 bits words. In LPS hash, the product of a matrix

$$M = \begin{pmatrix} a_0 + a_1\mathbf{i}\ b_0 + b_1\mathbf{i} \\ c_0 + c_1\mathbf{i}\ d_0 + d_1\mathbf{i} \end{pmatrix}$$

with a generator G_j is, by using $\mathbf{i}^2 + 1 \equiv 0 \bmod p$,

$$MG_j = \left(\begin{array}{c|c} (a_0g_{0,j} - a_1g_{1,j} - b_0g_{2,j} - b_1g_{3,j}) & (a_0g_{2,j} - a_1g_{3,j} + b_0g_{0,j} + b_1g_{1,j}) \\ +\mathbf{i}(a_0g_{1,j} + a_1g_{o,j} + b_0g_{3,j} - b_1g_{2,j}) & +\mathbf{i}(a_0g_{3,j} + a_1g_{2,j} - b_0g_{1,j} + b_1g_{0,j}) \\ \hline (c_0g_{0,j} - c_1g_{1,j} - d_0g_{2,j} - d_1g_{3,j}) & (c_0g_{2,j} - c_1g_{3,j} + d_0g_{0,j} + d_1g_{1,j}) \\ +\mathbf{i}(c_0g_{1,j} + c_1g_{o,j} + d_0g_{3,j} - d_1g_{2,j}) & +\mathbf{i}(c_0g_{3,j} + c_1g_{2,j} - d_0g_{1,j} + d_1g_{0,j}) \end{array}\right).$$

As all $g_{i,j}$ are smaller than $\sqrt{l}$ (for $l = 5$, $g_0 = 1$ and one of g_1, g_2, g_3 is ± 2), this product can be computed with a few field additions and shifts. The LPS step computation is significantly sped up with these formulae, at the only expense of doubling the internal memory usage and a few additional field operations in the final mapping. The same trick can be used to compute a Morgenstern hash step with only XORs and one-bit shifts, that is, very efficient operations.

Implementation results We implemented ZT, LPS and Morgenstern hash functions on a 32-bit 3.20GHz Pentium 4 using the GMP C library. The parameters we used are : a random irreducible polynomial $P_n(X)$ of degree 1024 for ZT, $l = 5$ and a random prime p of 1024 bits for LPS, $q = 2$ and a random polynomial $g(X)$ of degree 1024 for Morgenstern. We also implemented LPS with field multiplications (that is, without the trick above) to compare fairly with [5]. The bandwidths we obtain are 1.4MbHz for ZT, 733kbHz for LPS (83kbHz when we do field multiplications), and 613kbHz for Morgenstern hash function. An evaluation of the Morgenstern hash function (with $q = 2$) was also achieved for a XILINX VIRTEX4 FPGA, with a bandwidth of 360MbHz.

Cayley hashes are exceptionally fast in hardware and comparable to other provable hashes in software. Indeed, the best SHA-2 implementations on FPGA are currently below 1.8GbHz [7], that is only 5 times faster than Morgenstern hashes. The exceptional speed of Morgenstern (and ZT) hash in hardware comes from the simplicity of the step formulae that involve only XORs and one-bit shifts. In software, 1024-bit operations are sequentially decomposed into 32− or 64−bit operations, hence an efficiency lost. With the above trick, the computation of LPS hash is performed between 5 and 9 times faster than in [5] (145kbHz on a 64−bit platform). Still, Cayley hashes are currently a bit slower in software than other provable hashes, and far from the 500MbHz rates easily obtained for the SHA family. The VSH function [9] achieves rates of about 3.2MbHz while a fast variant of it runs a bit faster than 8MbHz (but involve a large amount of precomputation and storage). The fastest (but less secure) version of the FFT hashing scheme [17] runs at about 5.4MbHz.

Optimizations We are currently working on in many ways of accelerating Cayley hashes computation. Special parameters like generalized Mersenne's primes for LPS hashes or trinomials for ZT and Morgenstern hashes will speed up the computation by more than 50% as the field reductions would then be done for free. The two rows of each matrix can be computed in parallel and using associativity of the matrix product, long messages can also be decomposed in blocks treated separately. We believe that many improvements remain to be done and could together lead to a speed-up of a factor 10.

6 Current and Further Work

Cayley hashes raise many problems of different types, that include:

- Finding the best parameter sets for security (making subgroup attacks as unlikely as possible) as well as efficiency (making some parts of the computation easier).
- Considering the malleability problem which is the main drawback in our approach: roughly, from the hashes of two messages m_1 and m_2 it is easy to compute the hash of a related message, namely $m_1||m_2$. To remove this property, we are considering the hash function $\widetilde{H}(m) = \pi_2 \circ H \circ \pi_1 \circ \pi_2 \circ H \circ \pi_1(m)$. The

function $\widetilde{H}$ enjoys the security properties of H and can be computed in at most twice the same time; we want to provide arguments for its non-malleability.
– Replacing matrices by vectors: the idea is to compute only one row of the matrices that are the outputs of the Cayley hashes, starting from $(1, 0)$ or $(0, 1)$. For LPS hash, this approach corresponds to using the graphs $Y_{l,p}$ rather than the graphs $X_{l,p}$ of [16]. The new graphs enjoy expansion properties deduced from the previous ones. Whether the corresponding representation problem becomes easier or not is an open question.

7 Conclusion

Cayley hashes are elegant cryptographic hash functions constructed from Cayley graphs. In this paper, we have reviewed two previous instances (the ZT and LPS hashes) and proposed a new one based on Morgenstern's Ramanujan graphs. The expanding properties of Cayley graphs guaranty the uniform distribution of outputs, while the collision and preimage resistance follows from a hardness hypothesis on a representation problem in simple linear groups. We have given arguments for the hardness of these problems, based on existing attacks against the ZT hash function.

In this paper, we also showed that Cayley hashes can be made very efficient in hardware and comparable to other provable hashes in software. Indeed, our new Morgenstern hash is currently only 5 times slower than the best hardware implementations of SHA-256, and we are aware of methods and tricks to develop much faster implementations. Most importantly, Cayley hashes can be parallelized easily which will allow even better performances.

The drawback of the parallel property is that it comes with some malleability in the hash function. In Section 6, we proposed a heuristical way to avoid malleability. We hope the very nice properties of Cayley hashes will stimulate research on this question as well as on the hardness of the representation problem.

References

1. `http://www.csrc.nist.gov/groups/ST/hash/sha-3/index.html`.
2. Fips 180-2 secure hash standard.
3. 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.
4. N. Alon and V. Milman. λ_1, isoperimetric inequalities for graphs, and superconcentrators. *Journal of Combinatorial Theory, series B*, 38:73–88, 1985.
5. D. X. Charles, E. Z. Goren, and K. E. Lauter. Cryptographic hash functions from expander graphs. Cryptology ePrint Archive: Report 2006/021, available from `http://eprint.iacr.org/2006/021.pdf`.
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. R. Chaves, G. Kuzmanov, L. Sousa, and S. Vassiliadis. Improving SHA-2 hardware implementations. In L. Goubin and M. Matsui, editors, *CHES*, volume 4249 of *Lecture Notes in Computer Science*, pages 298–310. Springer, 2006.
8. P. Chiu. Cubic Ramanujan graphs. *Combinatorica*, 12:275–285, 1992.
9. 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.
10. J. Dodziuk. Difference equations, isoperimetric inequality, and transience of certain random walks. *Transactions of the American Mathematical Society*, 284:787–794, 1984.
11. W. Geiselmann. A note on the hash function of Tillich and Zémor. In D. Gollmann, editor, *Fast Software Encryption*, volume 1039 of *Lecture Notes in Computer Science*, pages 51–52. Springer, 1996.
12. O. Goldreich. *Fundations of Cryptography, Volume II Basic Applications*. Cambridge University Press, 2004.
13. S. Hoory, N. Linial, and A. Wigderson. Expander graphs and their applications. *Bull. Amer. Math. Soc.*, 43:439–561, 2006.
14. J. Katz and Y. Lindell. *Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series)*. Chapman & Hall/CRC, 2007.
15. J. Lafferty and D. Rockmore. Numerical investigation of the spectrum of certain families of Cayley graphs, 1993.
16. A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan graphs. *Combinatorica*, 8:261–277, 1988.
17. V. Lyubashevsky, D. Micciancio, C. Peikert, and A. Rosen. Provably secure FFT hashing. In *NIST 2nd Cryptogaphic Hash Workshop*, 2006.
18. M. Morgenstern. Existence and explicit construction of $q+1$ regular Ramanujan graphs for every prime power q. *Journal of Combinatorial Theory*, B 62:44–62, 1994.
19. P. Sarnak. *Some Applications of Modular Forms*. Cambridge University Press, 1990.
20. 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.
21. 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.
22. J.-P. Tillich and G. Zémor. Group-theoretic hash functions. In *Proceedings of the First French-Israeli Workshop on Algebraic Coding*, pages 90–110, London, UK, 1993. Springer-Verlag.
23. 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.
24. G. Zémor. Hash functions and Cayley graphs. *Des. Codes Cryptography*, 4(4):381–394, 1994.