Post-quantum hash functions using $\mathrm{SL}_n(\mathbb{F}_p)$

Corentin Le Coz[*], Christopher Battarbee, Ramón Flores[†],
Thomas Koberda[‡], and Delaram Kahrobaei[§]

Department of Mathematics: Algebra and Geometry (WE01),
Universiteit Gent, Belgium, corentinlecoz@outlook.com
Department of Computer Science, University of York, UK,
cb2036@york.ac.uk
Department of Geometry and Topology, University of Seville,
Spain, ramonjflores@us.es
Department of Mathematics, University of Virginia,
thomas.koberda@gmail.com
Computer Science and Mathematics Departments, Queens College
CUNY, Department of Computer Science, University of York, UK,
dkahrobaei@gc.cuny.edu

September 19, 2023

Abstract

We define new families of Tillich-Zémor hash functions, using higher dimensional special linear groups over finite fields as platforms. The Cayley graphs of these groups combine fast mixing properties and high girth, which together give rise to good preimage and collision resistance of the corresponding hash functions. We justify the claim that the resulting hash functions are post-quantum secure.

1 Introduction

Hash functions obtained from families of expander graphs were introduced by Charles-Lauter-Goren in [CLG08], where they were in turn inspired by a scheme of Tillich-Zémor [TZ94a]. Charles-Lauter-Goren

1

considered specific families of expander graphs discovered by Lubotzky-Phillips-Sarnak [LPS88] and Pizer [Piz90]. The Charles-Lauter-Goren construction is quite general, and can be applied to any expander family in which finding cycles is hard, and thereby furnishes collision resistant hash functions. Similar schemes have been proposed by several authors; see [SS16, BSV17, GM18], and [Sos18] for a survey on this topic.

Interest in hash functions based on novel platforms fits into the context of recent efforts to modernize existing hash functions, and to adapt them to the design and security of hash-based consensus mechanisms, most notably with respect to blockchains [Bor20], and especially in light of the recently proved practicality of finding collisions in the SHA-1 hashing algorithm [SBK$^+$17].

The general idea behind Tillich-Zémor hash functions is the following. Fixing a base vertex, the input of the hash function is interpreted as a sequence of instructions, resulting in a non-backtracking path in a d-regular graph. The output of the hash function is the endpoint vertex of the path. More precisely, the input is a string of numbers in

$$[d-1] := \{1, 2, \ldots, d-1\}$$

of arbitrary length, and the output is the vertex obtained by performing a simple walk starting at a base vertex, using the elements of $[d-1]$ as transition data for subsequent steps in the walk. See Definition 2.4 below for details.

A well-constructed hash function is an efficiently computable function which enjoys two main features. The first is *preimage resistance*, which means that given a point in the hash value, it is computationally hard to find an input that maps to that hash value. The second, is *collision resistance*, which requires the problem of finding distinct inputs with the same output to be computationally difficult.

The main goal of this paper is to propose a new Goren-Lauter-Charles–type scheme, where the hash functions use Cayley graphs of the special linear groups $\mathrm{SL}_n(\mathbb{F}_p)$ as platforms, where here p is prime and $n \geq 3$. The restriction to the fields $\mathbb{F}_p$ comes from the fact that the corresponding groups will be obtained as quotients of $\mathrm{SL}_n(\mathbb{Z})$. A crucial observation is that, in schemes using these groups as a platform, the problem of finding a preimage or a collision corresponds to finding factorizations of the identity matrix with prescribed factors. With this observation in hand, and by taking into account recent work of Arzhantseva-Biswas [AB22] concerning the expanding properties of the Cayley graphs of these groups, we offer a detailed study of the security of our protocol. In particular, we have the following:

- *Preimage resistance.* In Proposition 2.3, we collect the expansion properties the family of Cayley graphs $\{G_{n,p}\}_p$ of the groups $\mathrm{SL}_n(\mathbb{F}_p)$, where $n \geq 3$ is fixed and where p tends to infinity. Expansion in this family of graphs guarantees good mixing properties, because under these conditions the random walk gives a good approximation to the uniform distribution after $O(\log p)$ steps. In particular, the likelihood of success of an indistinguishability attack is strictly bounded,

2

and tends to $1/2$ as the order of the group tends to infinity; see Proposition 3.3.

- *Collision resistance.* The strength of our hash function with respect to collision resistance is mainly based on the absence of small cycles in the Cayley graphs of the underlying groups. In fact, Proposition 3.1 provides a lower bound on the girth of the graphs $\{G_{n,p}\}_p$ on the order of $\log p$. It follows that a factorization of the identity, which is easily seen to be equivalent to finding a collision for the hash function, is in turn equivalent to solving over a system of n^2 equations in a number of variables that is $O(\log p)$, over the field $\mathbb{F}_p$. In full generality, solving such systems of equations is NP-hard.

  Replacing the problem of factoring the identity with the problem of factoring an arbitrary group element yields a similar system of equations, lending further evidence of resistance of the hash function to finding preimages; see Section 3.2.

For $n = 2$, certain Cayley graphs of the groups $\mathrm{PSL}_2(\mathbb{F}_p)$ give rise to the celebrated Lubotzky–Phillips–Sarnak expander graphs [LPS88], which were then used to build hash functions in [CLG08]. A successful collision attack (i.e. an efficient computation of a collision) was found in [TZ08], by taking coefficients in $\mathbb{Z}[i]$ and then reducing to a system of equations of degree two. More recently, Sardari [Sar21] attacked preimage resistance by designing a polynomial-time algorithm that represents a number as a sum of four squares with some restricted congruence conditions. The essentially different nature of higher dimensional special linear groups gives evidence of additional security, and makes it likely that these attacks are far more difficult to execute for the hash functions proposed here.

Considering symmetric generating sets enables us to employ results from the theory of simple random walks in simplicial graphs. Nevertheless, the fact that we restrict ourselves to non backtracking random walks precludes the use of multiplicativity of the hash function, and thus complicates parallel computation. We discuss these issues in Section 3.4.

Structure of the paper: In Section 2 we give the relevant group theoretic background, define the hash functions, prove the expansion property of the Cayley graphs, and exhibit concrete examples. In Section 3, we describe the various properties of our scheme: namely, we relate free generation with a lower bound on the girth; we then describe the role of polynomial equations in preimage finding and in collision resistance, as well as the role of mixing in indistinguishability attacks. Finally, we discuss multiplicativity and parallel computing, showing that the collision attack from Grassl et. al. [GIMS11] using palindromes does not break the scheme presented here. Section 4 concludes the paper.

2 Definition of the hash functions

This section defines our hash functions and exhibits concrete instances. We start by recalling some relevant background material which will be essential in our construction and in the sequel.

2.1 Background about special linear groups

For general results about special linear groups over finite fields, we refer the reader to Hall's book [Hal15]. In this section we concentrate on a number of properties established by Arzhantseva-Biswas in their article [AB22]. We summarize their results in the following theorem:

Theorem 2.1 (Arzhantseva-Biswas [AB22])**.** *Let* $n \geq 2$ *and let* p *a prime. Write* $\pi_p : \mathrm{SL}_n(\mathbb{Z}) \to \mathrm{SL}_n(\mathbb{F}_p)$ *for the canonical projection given by reduction modulo* p. *There exist matrices* $\tilde{A}, \tilde{B} \in \mathrm{SL}_n(\mathbb{Z})$ *such that:*

(i) *There exists a prime* p_0 *such that for all* $p \geq p_0$, *the matrices* $\tilde{A}_p := \pi_p(\tilde{A})$ *and* $\tilde{B}_p := \pi_p(\tilde{B})$ *generate* $\mathrm{SL}_n(\mathbb{F}_p)$.

(ii) *If* $\langle \tilde{A}, \tilde{B} \rangle$ *is the subgroup generated by* $\tilde{A}$ *and* $\tilde{B}$ *inside of* $\mathrm{SL}_n(\mathbb{Z})$, *then* $\langle \tilde{A}, \tilde{B} \rangle$ *is isomorphic to* F_2, *the free group of rank two.*

(iii) *The diameter of the Cayley graph* $G_{n,p}$ *of* $\mathrm{SL}_n(\mathbb{F}_p)$ *with respect to* $\{\tilde{A}_p^{\pm 1}, \tilde{B}_p^{\pm 1}\}$ *is* $O(\log p)$.

Observe that for $n \geq 3$, items (i) and (ii) reflect the fact that the subgroup of $\mathrm{SL}_n(\mathbb{Z})$ generated by $\tilde{A}$ and $\tilde{B}$ is usually a thin subgroup of $\mathrm{SL}_n(\mathbb{R})$. The fact that $\tilde{A}_p$ and $\tilde{B}_p$ generated the corresponding finite quotients for all but finitely many values of p is a reflection of strong/superstrong approximation. In turn, item (iii) implies that the girth of $G_{n,p}$ is optimal, because the graphs $\{G_{n,p}\}_p$ form a family of expander graphs (see Proposition 2.3 below).

Remark 2.2. When using the Cayley graphs $G_{n,p}$ as a platform, we think of n as being fixed and p as modulating the level of security, with the trade-off being that the hash functions become more expensive to compute for large p.

Possible choices for $\tilde{A}$ and $\tilde{B}$ are given by:

$$\tilde{A} = \begin{pmatrix} 1 & a & 0 & 0 & \ldots & 0 \\ 0 & 1 & a & 0 & \ldots & 0 \\ 0 & 0 & 1 & a & \ldots & 0 \\ \vdots & & & & & \vdots \\ & & & & \ldots & a \\ 0 & 0 & 0 & 0 & \ldots & 1 \end{pmatrix}, \; \tilde{B} = \begin{pmatrix} 1 & 0 & 0 & \ldots & & 0 \\ b & 1 & 0 & \ldots & & 0 \\ 0 & b & 1 & \ldots & & 0 \\ 0 & 0 & b & \ldots & & 0 \\ \vdots & & & & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & b & 1 \end{pmatrix} \in \mathrm{SL}_n(\mathbb{F}_p),$$

with $a, b \geq 2$. These matrices will be crucial in the description of our hash function.

2.2 Expansion

For us to implement the Charles-Lauter-Goren approach, we must take advantage of the good mixing properties of the expander graphs.

Proposition 2.3. *For $n \geq 3$ fixed and $p \to \infty$, the sequence $\{G_{n,p}\}_p$ is a family of expander graphs.*

Sketch of proof. By item (i) of Theorem 2.1, there exists a prime p_0 such that for all $p \geq p_0$, the matrices $\tilde{A}_p := \pi_p(\tilde{A})$ and $\tilde{B}_p := \pi_p(\tilde{B})$ generate $\mathrm{SL}_n(\mathbb{F}_p)$. Now, since $\mathrm{SL}_n(\mathbb{Z})$ has property (T) for $n \geq 3$ [BdlHV08], and $\mathrm{SL}_2(\mathbb{Z})$ has property (τ) with respect to the family of congruence subgroups [Lub05], the proposition follows. $\square$

An immediate consequence of this proposition is that the random walk approximates the uniform distribution after $O(\log p)$ steps in the corresponding graph $G_{n,p}$. As we will elaborate in Section 3.3, this result is relevant for the purpose of analyzing preimage resistance to indistinguishability attacks; see Proposition 3.3, in particular. We note that in [BSV17], random walks are conducted on Cayley graphs with respect to non-symmetric generating sets, and thus their asymptotic behavior is less clear. Similar issues arise in [TNS20], since then hash values could be restricted to a proper subgroup. As stated in [AB22], we note that one can effectively compute the bound lower p_0. No explicit bound on p_0 has been given, though by combining existing results one can probably prove that p need not be very large, likely on the order of magnitude of n; see for instance [GV12, Appendix] and [Gur99, Theorem D]. Note that the larger the value of the prime p, the more secure the hash function.

2.3 The general construction

We now use matrices given in [AB22] to define an explicit family of hash functions.

Definition 2.4 (Special linear group based hash functions)**.** Let $n \geq 3$ and let p be a prime number. Let $a, b, \ell \geq 2$ that satisfy:

- If $n = 3$, $a \equiv 1 \pmod{3}$, $b \equiv -1 \pmod{3}$ and $\ell = 4^k$ for some positive integer k.

- If $n \geq 4$, there exists a prime q such that $n \equiv a \equiv b \equiv 1 \pmod{q}$ and ℓ is at least $3(n-1)$ and is of the form $q^{k+1} + 1$ for some integer k.

Consider the matrices $\tilde{A}$ and $\tilde{B}$ from the previous section. In the following we will denote $A \equiv \tilde{A}^\ell$ and $B \equiv \tilde{B}^\ell$.

We use the notation $[k]$ to denote the set of integers in $[k]$, and $[k]^*$ to denote the set of finite strings of integers in $[k]$. We now define the hash function $\varphi : [3]^* \to \mathrm{SL}_n(\mathbb{F}_p)$. We start by choosing bijections

$$s : [4] \to \{A^{\pm 1}, B^{\pm 1}\}, \quad s_\lambda : [3] \to \{A^{\pm 1}, B^{\pm 1}\} \setminus s(\lambda)$$

for each $\lambda \in [4]$.

Then, given

$$x = (x_i)_{1 \leq i \leq k} \in [3]^k,$$

we have the following inductive definition:

- $B_1 = s(x_1)$,
- $B_i = s_\lambda(x_i)$ with $\lambda = s^{-1}(B_{i-1}^{-1})$, for $2 \leq i \leq k$.

Finally, we set $\varphi(x) := B_1 \cdots B_k \in \mathrm{SL}_n(\mathbb{F}_p)$.

Remark 2.5. Note that $G_{n,p}$ is 4–regular, so that after the first bit x_1 of the input x, there are exactly three non-backtracking edges in the graph by which to proceed. The input x can thus be viewed as encoding a reduced word in the free group F_2. The lack of backtracking in the resulting walk on $G_{n,p}$ is crucial for the avoidance of collisions, as well as for the reduction of mixing time.

As stated in [AB22], the elements $\{A, B\}$ generate a free subgroup of $\mathrm{SL}_n(\mathbb{Z})$ and generate $\mathrm{SL}_n(\mathbb{F}_p)$ for all but finitely many values of p, and these facts give rise to strong preimage and collision resistance of the resulting hash functions.

2.4 A concrete example

We finish this section by describing a family of concrete examples of hash functions, which are constructed for the specific values $a = 4$, $b = 2$ and $\ell = 4$. We do not know what minimal value of n would ensure security.

Definition 2.6. Let p be a prime, and let

$$
A = \begin{pmatrix} 1 & 4 & 0 & 0 & \ldots & 0 \\ 0 & 1 & 4 & 0 & \ldots & 0 \\ 0 & 0 & 1 & 4 & \ldots & 0 \\ \vdots & & & & & \vdots \\ & & & & \ldots & 4 \\ 0 & 0 & 0 & 0 & \ldots & 1 \end{pmatrix}^4, \ B = \begin{pmatrix} 1 & 0 & 0 & \ldots & & 0 \\ 2 & 1 & 0 & \ldots & & 0 \\ 0 & 2 & 1 & \ldots & & 0 \\ 0 & 0 & 2 & \ldots & & 0 \\ \vdots & & & & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 2 & 1 \end{pmatrix}^4 \in \mathrm{SL}_n(\mathbb{F}_p),
$$

Let $s(1) = A$, $s(2) = B$, $s(3) = A^{-1}$, $s(4) = B^{-1}$. We define the functions $\{s_\lambda\}_{\lambda \in [4]}$ as follows:

- $s_1(1) = B$, $s_1(2) = A^{-1}$, $s_1(3) = B^{-1}$,
- $s_2(1) = A$, $s_2(2) = A^{-1}$, $s_2(3) = B^{-1}$,
- $s_3(1) = A$, $s_3(2) = B^{-1}$, $s_3(3) = B$,
- $s_4(1) = A$, $s_4(2) = A^{-1}$, $s_4(3) = B$,

Given an input sequence $x = \{x_i\}_{i \in [1,k]} \in [3]^k$, we inductively define:

- $B_1 = s_1(x_1)$
- $B_i = s_\lambda(x_i)$, with $\lambda = s^{-1}(B_{i-1}^{-1})$, for each $k \in [2, k]$.

Then, the sequence x is hashed to the matrix:

$$
\varphi(x) = B_1 \cdots B_k.
$$

Thus, we obtain a hash function for every $n \geq 3$.

Example 2.7. With $n = 3$ we have:

$$
A = \begin{pmatrix} 1 & 16 & 96 \\ 0 & 1 & 16 \\ 0 & 0 & 1 \end{pmatrix}, \ B = \begin{pmatrix} 1 & 0 & 0 \\ 8 & 1 & 0 \\ 24 & 8 & 1 \end{pmatrix} \in \mathrm{SL}_3(\mathbb{F}_p),
$$

For example, the sequence 2232221 will be mapped to

$$A^{-2}B^{-1}A^{-3}B = \begin{pmatrix} 694190977 & 233260720 & 29297952 \\ -38379648 & -12896255 & -1619792 \\ 1191936 & 400512 & 50305 \end{pmatrix} \in \mathrm{SL}(\mathbb{F}_p).$$

3 Properties of the constructed hash functions

In this section we use graph and group-theoretic machinery to describe the security of the hash functions defined above. We center our analysis on resistance to preimage and collision breaking. The exposition is divided into three parts: first, we establish a lower bound in the girth of the Cayley graphs of the group $\mathrm{SL}_n(\mathbb{F}_p)$ with respect to the generating system $\{A_p^{\pm 1}, B_p^{\pm 1}\}$; second, we describe the consequences of girth bounds for collision resistance; third, we investigate resistance of the protocol against an indistinguishability attack.

3.1 Free groups and girth

The following proposition is in the spirit of [BSV17].

Proposition 3.1. *Let $A, B \in \mathrm{SL}_n(\mathbb{Z})$ such that the entries of $A^{\pm 1}$ and $B^{\pm 1}$ are bounded in absolute value by a positive constant c. If A and B generate a free subgroup of $\mathrm{SL}_n(\mathbb{Z})$, then the girth of the Cayley graph of $\langle A_p, B_p \rangle \leq \mathrm{SL}_n(\mathbb{F}_p)$, with respect to $\{A_p^{\pm 1}, B_p^{\pm 1}\}$ is at least*

$$\left\lfloor \frac{\log(p-1)}{\log nc} \right\rfloor.$$

Proof. For any reduced word w in $A^{\pm 1}$ and $B^{\pm 1}$, we write $w_{\mathbb{Z}}$ (resp. $w_{\mathbb{F}_p}$)) for the projection of w to $\mathrm{SL}_n(\mathbb{Z})$ (resp. $\mathrm{SL}_n(\mathbb{F}_p)$). It follows by a straightforward induction on k that, if w has length k, then the entries of $w_{\mathbb{Z}}$ cannot exceed $(nc)^k$ in absolute value. Now, let ℓ be the girth of the corresponding Cayley graph. Then, there exists a non trivial reduced word w of length ℓ such that $w_{\mathbb{F}_p} = 1$. It follows that $w_{\mathbb{Z}}$ is of the form $1 + pM$, where M is an integer matrix. Since w is non trivial and since $\{A, B\}$ generate a rank two free subgroup of $\mathrm{SL}_n(\mathbb{Z})$, the matrix M is nonzero. We conclude that $w_{\mathbb{Z}}$ has an entry of absolute value at least $p - 1$. Since the entries of $w_{\mathbb{Z}}$ cannot exceed $(nc)^k$ in absolute value, we have that the length ℓ of w is bounded below by $\lfloor \frac{\log(p-1)}{\log nc} \rfloor$, the desired conclusion. $\square$

3.2 Preimage and collision resistance, and post-quantum heuristics

We now analyze the resistance of our model to finding preimages and to collisions. Observe that finding a preimage of a particular hash value (resp. finding a collision of hash values) is equivalent to finding a factorization of a given group element (resp. of the identity) in $\mathrm{SL}_n(\mathbb{F}_p)$ with

respect to the generating set. We note that in general, Tillich–Zémor hash functions seem to have robust collision resistance; see [Tin23].

The matrices A_p, B_p involved have order p, so a factorization can be seen as a family of equations $\{(E_m)\}_{m \geq 0}$ with variables

$$k_1, \ldots, k_m, \ell_1, \ldots, \ell_m \in \mathbb{F}_p$$

satisfying:

$$(E_m) \qquad\qquad A^{k_1} B^{\ell_1} \ldots A^{k_m} B^{\ell_m} = M,$$

for a given challenge $M \in \mathrm{SL}_n(\mathbb{F}_p)$. Note that there are trivial solutions to preimage and collision breaking of the hash function, since A_p^p is the identity. Since the girth of $G_{n,p}$ is $O(\log p)$, we consider nontrivial solutions to preimage or collision breaking to be ones where

$$C_1 \log p \leq \sum_{i=1}^{m} (k_i + \ell_i) \leq C_2 \log p,$$

where C_1 and C_2 are positive constants depending on n but not on p. Note that estimates for C_1 and C_2 can be produced, and that Proposition 3.1 furnishes an estimate for C_1, for instance. Sharp values for C_1 and C_2 are of relatively minor consequence for us.

Each entry of the left-hand side matrix in equation (E_m) is polynomial in

$$k_1, \ldots, k_m, \ell_1, \ldots, \ell_m.$$

Thus, the equation (E_m) corresponds to a system of n^2 multivariate polynomial equations over $\mathbb{F}_p$.

Solving multivariate polynomial equations over a finite field is known to be NP-hard [GJ79], which suggests a good level of security. Moreover, the reduction to solving multivariate polynomials, a class of hardness problems considered for standardization by NIST, provides a certain degree of confidence that the hash function is post-quantum. We contrast this approach with schemes based on isogeny graphs, which reduce to a more well-defined problem, albeit one not know to be NP-hard.

NP-hardness of a class of problems is a worst case complexity property, and for certain NP-hard classes of problems, relatively simple and efficient algorithms can find solutions in the vast majority of cases. Thus, NP-hardness of the underlying problem is not a guarantee of post-quantum behavior of the hash function.

A more compelling case for the hash function to be post-quantum arises from empirical difficulty of factoring in special linear groups over finite fields. For instance, in [FPPR11], subexponential factorization algorithms were found for $\mathrm{SL}_2(\mathbb{F}_{2^k})$, and these were only found in 2011. These algorithms rely essentially on the fact that the matrices are 2×2, and on the fact that the underlying field has characteristic two. Thus, the methods do not generalize in any straightforward way to larger dimensional special linear groups nor to fields with odd characteristic. In practice, factoring matrices over finite fields is quite difficult, and implemented algorithms are inefficient. Hardness appears to be optimized when the system of equations resulting from (E_m) is neither underdetermined

nor overdetermined, i.e. when the number of equations and variables is comparable. Thus, the larger the value of p the more secure the hash function, at the expense of computational time and space, and the balance of degrees of freedom and constraints occurs when $n^2 \sim \log p$, or in other words when n is approximately the square root of the logarithm of p. This is precisely the balance to required so that the number of equations and number of variables are comparable, as per the foregoing discussion. We may then expect the factorization problem to take exponential time in the number of variables in this case, and by resistant to the speedup resulting from quantum attacks.

3.3 The mixing property and indistinguishability attacks

In this section we study how to formalize the relationship between mixing properties and indistinguishability, and how expansion weakens indistinguishability attacks and thus gives heuristic confidence in preimage breaking resistance.

3.3.1 The mixing property

By the mixing property, we mean that the output vertex of a random input — in our case a random walk — approaches the uniform distribution on the hash space. When the random walk approaches the uniform distribution quickly, mixing is observed even when the input messages have relatively small length, say polynomial in $\log p$. More precisely, we have the following corollary of result of Alon-Benjamini-Lubetzky-Sodin [ABLS07], which characterizes the rate at which a random walk on a graph converges to the uniform distribution in terms of the spectral properties of its adjacency matrix:

Theorem 3.2. *[ABLS07, Theorem 1.1, cf. proof of Theorem 1.3] Suppose $d > 2$. Let $X_0, X_1, \ldots, X_\ell$ be a non backtracking random walk on a d–regular connected graph G with N vertices. There is a constant $C > 0$ such that whenever $\ell \geq C \cdot \log N$ we have*

$$|\Pr(X_\ell = v) - 1/N| \leq 1/N^2,$$

for every vertex v of G.

Examining the proof given in [ABLS07], one finds that the rate of mixing depends not so much on the graph G, as much as on eigenvalues of the adjacency matrix of G. Thus, if G is a member of a sequence $\{G_i\}_{i \in \mathbb{N}}$ of expander graphs, we may take the constant C in Theorem 3.2 to depend only on the expansion constant of the family.

It is well-known that mixing properties are desirable in Tillich-Zémor hashing schemes; see [CLG08, TZ08]. As explained in [TZ08], mixing is particularly relevant when the hash functions are used in protocols whose security relies on the random oracle model; see [BR93] for example of such protocols. The relevance of this approach depends on the distribution of possible messages and in particular on how they are encoded, a question we do not address in the present paper.

Surprisingly few mathematical statements addressing the relationship between mixing and attacks are present in the literature; an example can be found in [Ste21, Theorem 3], in the context of commitment schemes. The following proposition, which is the main goal of this section, fits in this context. We will give precise definitions of all the terms involved below.

Proposition 3.3. *Let $\varphi : [3]^k \to \mathrm{SL}_n(\mathbb{F}_p)$ the hash function of Definition 2.4, and let $N = |\mathrm{SL}_n(\mathbb{F}_p)|$. There is a positive constant C such that if $k \geq C \cdot \log N$, then a generic indistinguishability attack wins with probability at most $1/2 + 1/N$.*

In particular we have the following immediate consequence. Note that probability $1/2$ is the best we can hope for if an adversary is asked to distinguish between two situations, because such an adversary can always choose at random with success probability $1/2$.

Corollary 3.4. *With the previous notation, as p tends to infinity, the probability of winning for a generic indistinguishability attack tends to $1/2$.*

The complexity of the so-called generic indistinguishability attack mentioned here is linear in the number of vertices of the graph involved. Thus, in the specific case of Proposition 3.3 it has exponential complexity, since the size of $\mathrm{SL}_n(\mathbb{F}_p)$ is polynomial in p, and our reference is $\log p$. Thus, we obtain a more general notion of indistinguishability than in [KL21, §6.8], restricted to polynomial-time algorithms.

The proof of Proposition 3.3 will be obtained as a corollary of the more general Proposition 3.5, and of the mixing properties of the random walk on $\mathrm{SL}_3(\mathbb{F}_p)$. We elaborate on these ideas presently.

3.3.2 Generic attack of indistinguishability challenge

The goal of this subsection is to define indistinguishability challenges, investigate possible attacks, and to justify to what extent the hash functions defined in this paper are resistant to these attacks.

Indistinguishability challenge. Let H be a finite set. The challenger $\mathcal{C}$ has access to two public probability distributions μ_X and μ_Y, which take values in H. These distributions are a priori distinct, and we think of them as being "close". $\mathcal{C}$ picks $X \in H$ (resp. $Y \in H$) at random, according to the distribution μ_X (resp. μ_Y). $\mathcal{C}$ then flips a balanced coin $P \in \{0, 1\}$ and sends to their adversary $\mathcal{A}$ the element $Z \in H$, which satisfies:

- $Z = X$ if $P = 0$,
- $Z = Y$ if $P = 1$.

Given Z, the goal of $\mathcal{A}$ is to guess P. In other words, $\mathcal{A}$ needs to decide if $\mathcal{C}$ sent an element of H according to the distribution of μ_X or of μ_Y.

Generic indistinguishability attack. The adversary $\mathcal{A}$ knows the public distributions μ_X and μ_Y. For each binary relation

$$* \in \{<, =, >\},$$

we denote by H_* the set of $h \in H$ such that $\Pr(Y = h) * \Pr(X = h)$. The best strategy available to $\mathcal{A}$ is the following:

- If $Z \in H_>$, then $\mathcal{A}$ guesses $P = 1$,
- If $Z \in H_=$, then $\mathcal{A}$ guesses a random $P \in \{0, 1\}$,
- If $Z \in H_<$, then $\mathcal{A}$ guesses $P = 0$.

In the following proposition, we relate the ℓ^∞-distance between the distributions μ_X and μ_Y on one hand, and the chances of success of $\mathcal{A}$ following the strategy above on the other hand.

Proposition 3.5. *Let $\varepsilon > 0$ be such that*

$$|\Pr(X = h) - \Pr(Y = h)| \leq \varepsilon,$$

for every $h \in H$. Then, $\mathcal{A}$ guesses the correct P with probability at most

$$\frac{1}{2} + \varepsilon \frac{\#H}{4}.$$

Remark 3.6. The probability that $\mathcal{A}$ guesses correctly is $1/2$ when the strategy is to guess a random $P \in \{0, 1\}$, so Proposition 3.5 above essentially says that $\mathcal{A}$ cannot do much better, provided the distributions are sufficiently close.

Proof of Proposition 3.5. Let $P_\mathcal{A} := \Pr(\mathcal{A} \text{ is right})$. We have

$$P_\mathcal{A} = \sum_{h \in H} \Pr(\mathcal{A} \text{ is right} \mid Z = h) \Pr(Z = h)$$

For each binary relation $* \in \{<, =, >\}$ let

$$P_* = \sum_{h \in H_*} \Pr(\mathcal{A} \text{ is right} \mid Z = h) \Pr(Z = h).$$

Then, we have $P_\mathcal{A} = P_< + P_= + P_>$. We analyze each of these terms independently.

- If $Z \in H_=$, then $\mathcal{A}$ guesses a random $P \in \{0, 1\}$. Thus,

$$P_= = \sum_{h \in H_=} \frac{\Pr(Z = h)}{2}$$
$$= \frac{\Pr(X = h) + \Pr(Y = h)}{4}.$$

- If $Z \in H_<$, then $\mathcal{A}$ guesses $P = 0$. Thus,

$$P_< = \sum_{h \in H_<} \Pr(P = 0 \mid Z = h) \Pr(Z = h)$$
$$= \sum_{h \in H_<} \Pr(Z = h \mid P = 0) \Pr(P = 0)$$
$$= \sum_{h \in H_<} \frac{\Pr(X = h)}{2}.$$

- If $Z \in H_>$, then $\mathcal{A}$ guesses $P = 1$. Thus,

$$P_> = \sum_{h \in H_>} \Pr(P = 1 \mid Z = h)\Pr(Z = h)$$

$$= \sum_{h \in H_>} \Pr(Z = h \mid P = 1)\Pr(P = 1)$$

$$= \sum_{h \in H_>} \frac{\Pr(Y = h)}{2}.$$

So, we obtain

$$P_\mathcal{A} = \frac{1}{2}\sum_{h \in H} \max\{\Pr(X = h), \Pr(Y = h)\}.$$

We can now give an upper bound on $P_\mathcal{A}$. We can assume without any loss of generality that $\#H_< \leq \frac{\#H}{2}$. Then,

$$P_\mathcal{A} = \frac{1}{2}\sum_{h \notin H_<} \Pr(Y = h) + \frac{1}{2}\sum_{h \in H_<} \Pr(X = h)$$

$$\leq \frac{1}{2}\sum_{h \notin H_<} \Pr(Y = h) + \frac{1}{2}\sum_{h \in H_<} (\Pr(Y = h) + \varepsilon)$$

$$\leq \frac{1}{2}\sum_{h \in H} \Pr(Y = h) + \varepsilon\frac{\#H_<}{2}$$

$$\leq \frac{1}{2} + \varepsilon\frac{\#H}{4}. \qquad \square$$

Generic indistinguishability attack in our framework. In the framework of hash functions, we consider a function $\varphi : S \to H$, where S and H are finite sets. The challenger $\mathcal{C}$ has two variables in hand, namely $X' \in S$ and $Y \in H$, both chosen uniformly at random. The indistinguishablility challenge is now adapted to the two variables $X := \varphi(X')$ and Y. We can now prove Proposition 3.3.

Proof of Proposition 3.3. From Theorem 3.2, there are constants

$$C, C' > 0$$

such that after

$$C \log N = C' \log p$$

steps, the non backtracking random walk on $\mathrm{SL}_n(\mathbb{F}_p)$ is at ℓ^∞-distance at most $1/N^2$ from the uniform distribution. From Proposition 3.5, the generic indistinguishability attack wins with probability at most $1/2 + 1/4N$. $\qquad \square$

3.4 Multiplicativity and parallel computing

The hash functions considered in this article take as input a string of integers in [3], converting each integer into a matrix of the form $\{A^{\pm 1}, B^{\pm 1}\}$, and finally outputs the product of these matrices.

The fact that we require the underlying walk to be non-backtracking implies that this attribution is not locally determined: a given bit in the string is mapped to a matrix that depends on the previous bits. This dependency can be dramatic: for example, according to Definition 2.6 a sequence of the form $133\cdots3$ will be mapped to the product $B\cdot B\cdot B\cdots B$, while a sequence of the form $333\ldots3$ will be mapped to the product $B^{-1}\cdot B^{-1}\cdot B^{-1}\cdots B^{-1}$. In particular, the last bit 3 of these two strings can be mapped to different matrices, depending on the first bit in the string. The endpoints of the corresponding walk in in the Cayley graph may be far away from each other.

As a consequence, the function φ need not be multiplicative under concatenation of strings. This lack of multiplicativity makes it difficult to perform parallel computations with the given hash functions, as we now investigate in more detail.

3.4.1 Good and bad tails

Recall that for a finite set X, the notation X^* is used for the set of finite length strings of elements of X. As before, the notation [3] denotes the set $\{1,2,3\}$.

Definition 3.7. Let G be a finite group, generated by two elements A and B. Let $\tilde{\varphi} : [3]^* \to \{A^{\pm1}, B^{\pm1}\}^*$.

A string $s \in [3]^*$ is called a *good tail* with respect to $\tilde{\varphi}$ if there exists $S \in \{A^{\pm1}, B^{\pm1}\}$ such that for every $s' \in [3]^*$, the last letter of $\tilde{\varphi}(s's)$ is S, where here $s's$ is the string obtained from the concatenation of s' and s. A string which is not a good tail is called a *bad tail*.

Local constraints in Definition 2.6, can be obtained by the following fact:

Fact 3.8. The function

$$\tilde{\varphi} : \{x_i\} \in [3]^* \mapsto \{B_i\} \in \{A^{\pm1}, B^{\pm1}\}^*$$

constructed in Definition 2.6 has the following good tails: 11, 31, 22, 32, 13 and 23.

Proof. It is straightforward to check that:

- any string ending in 11 or 31 outputs a string ending in A;
- any string ending in 22 or 32 outputs a string ending in A^{-1};
- any string ending in 13 outputs a string ending in B;
- any string ending in 23 outputs a string ending in B^{-1}. $\square$

The following proposition shows that the attribution above is optimal.

Proposition 3.9. *Special linear group based hash functions (Definition 2.4) admit at most six good tails of length two.*

The bound in Proposition 3.9 is sharp, as shown via the attributions of Definition 2.6. The proof of Proposition 3.9 will follow from the following lemma:

Figure 1: Description of the maps s_i in Definition 2.6

last step matrix	1	2	3
A^{-1}	B	A^{-1}	B^{-1}
B^{-1}	A	A^{-1}	B^{-1}
A	A	B^{-1}	B
B	A	A^{-1}	B

Lemma 3.10. *Let*

$$\tilde{\varphi} : \{x_i\} \in [3]^* \mapsto \{B_i\} \in \{A^{\pm 1}, B^{\pm 1}\}^*$$

be a special linear group based hash function (Definition 2.4), and let $b \in [3]$. There exists $b' \in [3]$ such that $b'b$ is a bad tail with respect to $\tilde{\varphi}$.

Proof. The only freedom that we have in the construction of Definition 2.4 is how we define the maps s_i. We call elements of $\{A^{\pm 1}, B^{\pm 1}\}$ *step matrices*. Using $\tilde{\varphi}$, we say that the elements of $[3]$ are *encoded* by step matrices. We summarize the definitions of the maps s_i in a table, with one row for each step matrix, and one column for each element of $[3]$. Each cell from this tabular contains a step matrix.

To use this table, start from a string $\{x_i\} \in [3]^*$. Say that for some $i > 1$, we want to find the step matrix associated with x_i. Let S be the step matrix encoding x_{i-1}. Then, the step matrix encoding x_i is the step matrix in the cell located in the row labelled by S and in the column labelled by x_i.

It follows from the definition of the maps s_i that for each step matrix S, the row labelled by S contains exactly the three matrices in the set $\{A^{\pm 1}, B^{\pm 1}\} \setminus S^{-1}$. The attributions of Definition 2.6 can be described as in Figure 1.

Moreover, since each matrix can actually be the last step matrix used, every cell of the table can potentially be used. Fix an element $b \in [3]$, and assume for a contradiction that every integer $b' \in [3]$ has the property that $b'b$ is a good tail. The column corresponding to each b' has to contain at least two different step matrices. This implies that, in the row labelled by b, at least two cells contain the same step matrix. Since this is true for each $b' \in [3]$, this implies in particular that there is a step matrix S that is contained three times in the column labelled by b. The fourth cell of this column cannot be part of the row labelled by S since this would give rise to another S in a different column. This implies that the label S' of this row appears in a cell of another column. Additionally, this column contains a cell with step matrix not equal to S'. Then, the label $b' \in [3]$ of this column gives us an integer having the property that inputs ending by $b'b$ can have either S or another matrix as a final matrix. This is a contradiction and concludes the proof of the lemma. □

Proof of Proposition 3.9. From Lemma 3.10, to each integer of $[3]$ corresponds at least on bad tail, giving three different bad tails. □

14

As remarked previously, Definition 2.6 shows that the estimate in Proposition 3.9 is sharp, and so that in some sense, we have found an optimal way of defining the maps s_i.

3.4.2   Multiplicativity

It follows from the discussion of good and bad tails above that multiplicativity of the hash function can be obtained by restricting to sequences whose product ends with the matrix $s(1)$.

Fact 3.11. In Definition 2.6, we have $\varphi(s_1 * s_2) = \varphi(s_1) \cdot \varphi(s_2)$, provided s_1 ends with 11 or 31.

3.4.3   Parallel computing

Multiplicativity of the hash function under suitable conditions can be leveraged to compute its values by parallel computation. First, look for good tail substrings, namely: 11, 31, 22, 32, 13 or 23. For generic messages, one would expect such substrings to be quite common. Next, split the input immediately following one of these strings, and apply a slightly modified hash function (i.e. using the relevant s_i instead of s_1 in the first matrix attribution). Finally, compute the product of the hash outputs.

Example 3.12. Say we want to hash the string 1321321323. Observe that: 13213**21323**. We thus compute: $M_1 = \varphi(13213)$ and $M_2 = \varphi'(21323)$, where φ' is defined analogously to φ in Definition 2.6, apart from the fact that B_1 is set to be $s_4(x_1)$ instead of $s_1(x_1)$, since $\varphi(13213)$ is a product ending by B. Finally, $\varphi(1321221323) = M_1 \cdot M_2$.

3.5   Palindromic attacks

One of several proposals of hashing by walks on Cayley graphs can be found in [TZ94b], wherein the Cayley graph is that of $SL_2(\mathbb{F}_{2^n})$. A method for finding collisions for this hash function is presented in [GIMS11] (cf. [FPPR11]); we argue that the attack does not apply in our case, though our evidence for this claim is primarily empirical.

The idea of [GIMS11] is to find collisions on *palindromes*; that is, bitstring entries that are invariant under reversing the order. To begin, one conjugates generators of $SL_2(\mathbb{F}_{2^n})$ to obtain new generators which give rise to an isomorphic graph, but which are symmetric matrices. That is, if the original generators are $\{A^{\pm 1}, B^{\pm 1}\}$, one finds a matrix C such that $\hat{A} = CAC^{-1}$ and $\hat{B} = CBC^{-1}$ are both symmetric matrices.

We first note that in our case, finding C is not easy; for $SL_3(5)$ and $SL_3(7)$, about 0.02% of the elements satisfy this criterion. Moreover, there is no obvious way to compute C; attempts to calculate the entries of such a matrix directly have proved resistant to equation solving methods in standard computer algebra systems - indeed, this approach is actually less efficient than just checking all possible matrices. Therefore, we do not have much data for larger primes, since the naive method used to find a suitable matrix C quickly becomes computationally infeasible.

Provided one can find a matrix C, it follows that collisions in the hash function with respect to $\hat{A}, \hat{B}$ as generators are exactly the collisions

with A, B as generators; one can therefore rename the matrices $\hat{A}, \hat{B}$ as A, B. One then proceeds according [GIMS11, Lemma 1]: upon input of a palindromic string v, the output of the product of conjugated generators in $SL_2(\mathbb{F}_{2^n})$ will always be a symmetric matrix.

Since our hash function requires avoidance of backtracking in the walk, we are not guaranteed a palindromic matrix product from a palindromic input string; however, since one could reverse-engineer the necessary input to obtain a palindromic matrix product, we proceed to discuss palindromic matrix products without reference to hash function.

It turns out, as one may check easily by induction, that a palindromic product in symmetric generators will itself be symmetric. The ultimate goal of [GIMS11] is to use this fact to demonstrate that the function

$$\rho : M \mapsto AMA + BMB,$$

where M is a palindromic product, outputs a matrix populated with either zeroes or the square of a field element appearing as an entry in M. One then employs number theoretic tricks to force the nonzero elements to 0 in M and thus to obtain $\rho(M) = 0$. One thus builds distinct but equal palindromic matrix products.

Consider the generators from Definition 2.6 over $SL_3(\mathbb{F}_{11})$. Transforming these generators with respect to the matrix

$$C = \begin{pmatrix} 2 & 6 & 10 \\ 5 & 3 & 10 \\ 2 & 3 & 3 \end{pmatrix},$$

one checks that the palindrome $M = ABABA$ is such that

$$M = \begin{pmatrix} 7 & 4 & 2 \\ 4 & 0 & 6 \\ 2 & 6 & 6 \end{pmatrix}, \quad \rho(M) = \begin{pmatrix} 2 & 1 & 5 \\ 1 & 1 & 7 \\ 5 & 7 & 7 \end{pmatrix}.$$

In particular, for each $i \in [10]$, the matrix $\rho(M)$ contains an entry that is not the ith power of any entry of M. This furnishes evidence that for $p = 11$, there is little hope of extending [GIMS11, Corollary 1] to our context; we argue that the lack of closed form of transformed generators in general, the difficulty of finding them for larger parameters, and this example with a small value of p, conspire to provide strong evidence that the approach will fail in general.

4   Conclusions

We have presented new Tillich-Zémor hash functions, with platforms Cayley graphs of $SL_n(\mathbb{F}_p)$ for $n \geq 3$. We show that choosing appropriate generating matrices produces graphs without small cycles, and having a quick mixing property, both of which are highly desirable for preimage and collision resistance. Moreover, flexibility of choice of generating matrices and of the dimension n gives the option of increasing the complexity of attacks. Future work includes the exact computation of the spectral gap and the prime p_0 (cf. item (i) of Theorem 2.1). Moreover, simulations should be carried out in order to compare with other existing schemes and determine the optimal values of p and n to be taken in implementations.

References

[AB22]   Goulnara Arzhantseva and Arindam Biswas. Logarithmic girth expander graphs of $SL_n(\mathbb{F}_p)$. *Journal of Algebraic Combinatorics*, 2022.

[ABLS07]   Noga Alon, Itai Benjamini, Eyal Lubetzky, and Sasha Sodin. Non-backtracking random walks mix faster. *Commun. Contemp. Math.*, 9(4):585–603, 2007.

[BdlHV08] Bachir Bekka, Pierre de la Harpe, and Alain Valette. *Kazhdan's Property (T)*. New Mathematical Monographs. Cambridge University Press, 2008.

[Bor20]   Tom Borthwick. Applications of homomorphic cryptographic primitives in blockchain and the internet of things, Undergraduate Thesis, University of York, 2020.

[BR93]   Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, CCS '93, page 62–73, New York, NY, USA, 1993. Association for Computing Machinery.

[BSV17]   Lisa Bromberg, Vladimir Shpilrain, and Alina Vdovina. Navigating in the Cayley graph of $SL_2(\mathbb{F}_p)$ and applications to hashing. *Semigroup Forum*, 94(2):314–324, 2017.

[CLG08]   Denis Charles, Kristin Lauter, and Eyal Goren. Cryptographic hash functions from expander graphs. *Journal of Cryptology*, 22:93–113, 12 2008.

[FPPR11]   Jean-Charles Faugère, Ludovic Perret, Christophe Petit, and Guénaël Renault. New subexponential algorithms for factoring in $SL_2(\mathbb{F}_{2^n})$. *IACR Cryptol. ePrint Arch.*, page 598, 2011.

[GIMS11]   Markus Grassl, Ivana Ilić, Spyros Magliveras, and Rainer Steinwandt. Cryptanalysis of the Tillich–Zémor hash function. *Journal of cryptology*, 24(1):148–156, 2011.

[GJ79]   Michael R. Garey and David S. Johnson. *Computers and intractability*. A Series of Books in the Mathematical Sciences. W. H. Freeman and Co., San Francisco, Calif., 1979. A guide to the theory of NP-completeness.

[GM18]   Mohammad Hossein Ghaffari and Zohreh Mostaghim. More secure version of a Cayley hash function. *Groups Complex. Cryptol.*, 10(1):29–32, 2018.

[Gur99]   Robert M. Guralnick. Small representations are completely reducible. *J. Algebra*, 220(2):531–541, 1999.

[GV12]   Alireza Salehi Golsefidy and Péter P. Varjú. Expansion in perfect groups. *Geom. Funct. Anal.*, 22(6):1832–1891, 2012.

[Hal15]   Brian Hall. *Lie groups, Lie algebras, and representations*, volume 222 of *Graduate Texts in Mathematics*. Springer, Cham, second edition, 2015. An elementary introduction.

[KL21]   Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. Chapman & Hall/CRC Cryptography and Network Security. CRC Press, Boca Raton, FL, 2021. Third edition [of 2371431].

[LPS88]   Alex Lubotzky, Ralph S. Phillips, and Peter C. Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, 1988.

[Lub05]   Alex Lubotzky. What is. . . property (τ)? *Notices Amer. Math. Soc.*, 52(6):626–627, 2005.

[Piz90]   Arnold K. Pizer. Ramanujan graphs and Hecke operators. *Bull. Amer. Math. Soc. (N.S.)*, 23(1):127–137, 1990.

[Sar21]   Naser T. Sardari. Complexity of strong approximation on the sphere. *Int. Math. Res. Not. IMRN*, (18):13839–13866, 2021.

[SBK+17]   Marc Stevens, Elie Bursztein, Pierre Karpman, Ange Albertini, and Yarik Markov. The first collision for full SHA-1. In *Advances in cryptology—CRYPTO 2017. Part I*, volume 10401 of *Lecture Notes in Comput. Sci.*, pages 570–596. Springer, Cham, 2017.

[Sos18]   Bianca Sosnovski. Recent developments in Cayley hash functions. In *Mathematical software—ICMS 2018*, volume 10931 of *Lecture Notes in Comput. Sci.*, pages 438–447. Springer, Cham, 2018.

[SS16]   Vladimir Shpilrain and Bianca Sosnovski. Compositions of linear functions and applications to hashing. *Groups Complex. Cryptol.*, 8(2):155–161, 2016.

[Ste21]   Bruno Sterner. Commitment schemes from supersingular elliptic curve isogeny graphs. Cryptology ePrint Archive, Report 2021/1031, 2021. `https://ia.cr/2021/1031`.

[Tin23]   Simran Tinani. Methods for collisions in some algebraic hash functions. 2023.

[TNS20]   Hayley Tomkins, Monica Nevins, and Hadi Salmasian. New Zémor-Tillich type hash functions over $\mathrm{GL}_2(\mathbb{F}_{p^n})$. *J. Math. Cryptol.*, 14(1):236–253, 2020.

[TZ94a]   Jean-Pierre Tillich and Gilles Zémor. Group-theoretic hash functions. In *Algebraic coding (Paris, 1993)*, volume 781 of *Lecture Notes in Comput. Sci.*, pages 90–110. Springer, Berlin, 1994.

[TZ94b]   Jean-Pierre Tillich and Gilles Zémor. Hashing with SL_2. In *Annual International Cryptology Conference*, pages 40–49. Springer, 1994.

[TZ08]   Jean-Pierre Tillich and Gilles Zémor. Collisions for the LPS expander graph hash function. In *Proceedings of the Theory and Applications of Cryptographic Techniques 27th Annual International Conference on Advances in Cryptology*, EUROCRYPT'08, page 254–269, Berlin, Heidelberg, 2008. Springer-Verlag.