SHORT COLLISION SEARCH IN ARBITRARY SL_2 HOMOMORPHIC HASH FUNCTIONS

CIARAN MULLAN AND BOAZ TSABAN

Abstract. We study homomorphic hash functions into $\mathrm{SL}_2(q)$, the 2×2 matrices with determinant 1 over the field with q elements. Modulo a well supported number theoretic hypothesis, which holds in particular for all concrete homomorphisms proposed thus far, we prove that a random homomorphism is at least as secure as any concrete homomorphism. For a family of homomorphisms containing several concrete proposals in the literature, we prove that collisions of length $O(\log q)$ can be found in running time $O(\sqrt{q})$. For general homomorphisms we offer an algorithm that, heuristically and according to experiments, in running time $O(\sqrt{q})$ finds collisions of length $O(\log q)$ for q even, and length $O(\log^2 q/\log\log q)$ for arbitrary q. For any conceivable practical scenario, our algorithms are substantially faster than all earlier algorithms and produce much shorter collisions.

1. Introduction

Let $\{0,1\}^*$ be the monoid of all finite bitstrings with string concatenation as monoid multiplication and the empty string as identity element. Let $\mathrm{SL}_2(q)$ be the group of 2×2 matrices of determinant 1 with entries in the finite field $\mathbb{F}_q$ with $q = p^n$ elements. Over 20 years ago, Zémor [17] proposed a general hash function construction employing homomorphisms $h\colon \{0,1\}^* \to \mathrm{SL}_2(q)$, that is, functions h with the property that $h(uv) = h(u)h(v)$ for all $u, v \in \{0,1\}^*$. For a pair of elements $A = (A_0, A_1)$ of $\mathrm{SL}_2(q)$, denote by h_A the unique homomorphism such that $h_A(0) = A_0$ and $h_A(1) = A_1$. A bitstring $b_1 \dots b_m \in \{0,1\}^*$ is hashed to the matrix

$$h_A(b_1 \dots b_m) = h_A(b_1) \cdots h_A(b_m) = A_{b_1} \cdots A_{b_m} \in \mathrm{SL}_2(q).$$

At present, feasible cryptanalyses on this construction apply for only very special instances of A and q. An elegant cryptanalysis for the case where q is a power of 2 and A is a specific, natural pair of matrices was recently provided by Grassl et al. [7]. We refer to the survey of Petit and Quisquater [12] for an introduction and details of known cryptanalytic results.

In this paper we study Zémor's construction in its full generality. Based on a well supported conjecture concerning expander graphs, in Section 2 we prove that $\mathrm{SL}_2(q)$ homomorphic hash functions do not become less secure if we switch to random homomorphisms. In Section 3 we provide an algorithm producing, modulo the same well-known conjecture, collisions of length $O(\log q)$ in time $O(\sqrt{q})$, for arbitrary q and a class of homomorphisms including those in [15] (end of §6, $i = 2$), [17], [16], and [1]. In Section 4, for random (A_0, A_1) and arbitrary q we provide a collision search algorithm, and show, heuristically, that it finds collisions of length $O(\log^2 q/\log\log q)$ in running time $O(\sqrt{q})$.

In Section 5 we show that, for messages of all practical sizes, our algorithm is faster and produces much shorter collisions than the best known subexponential time algorithm due to Faugère et al. [5]. Moreover, it is shown that the heuristic methods of Petit [14] and Faugère et al. can be used, for q a power of 2, to reduce an *arbitrary* pair of generators (A_0, A_1) into a form in which our algorithm of Section 3 applies. Consequently, we obtain collisions of linear length for arbitrary homomorphisms into $\mathrm{SL}_2(2^n)$.

The theory employed in Sections 2 and 3 may be used to obtain, in a rigorous manner, estimations for the first phase of an earlier algorithm of Petit et al. [13]. We survey this algorithm in Appendix A. For an optimal choice of parameters we estimate its performance, which turns out to be not as good as our new algorithms. Furthermore, our algorithms are conceptually simpler: unlike Petit et al. we do not appeal to discrete logarithm solving or use of the LLL algorithm. We remark that Petit et al.'s algorithm produces bistrings hashing to the identity matrix, of length linear in p. While the same can be done with our first algorithm of Section 3, apparently this cannot be achieved with our second, more general algorithm of Section 4.

Finally, in Appendix B we prove that palindromic collisions, as exploited by Grassl et al. [7] in their efficient attack for q even, do not exist for arbitrary q, based on the same natural generating sets.

The running time of all algorithms studied in this paper is measured by the number of multiplications of elements of $\mathrm{SL}_2(q)$. We mention, here only, that the memory required by our algorithm can be made negligible, using distinguished points as in [13, §6]. All of our estimations are supported by extensive computer experiments. When we are interested in estimating the involved constants, we use lg, the logarithm in base 2, instead of log. The operator $|\ |$ means: absolute value when applied to a real number, cardinality when applied to a set, and bitlength when applied to a bitstring.

2. Hashing with random elements is at least as secure

In earlier papers on $\mathrm{SL}_2(q)$ hash functions (see [12] and references therein), much effort has been put on selecting the pair $(A_0, A_1) = (h(0), h(1))$ carefully. Here, we show that hashing with a random homomorphism—that is, with a pair of random elements (A_0, A_1)—is not less secure than hashing with any prescribed, carefully chosen homomorphism.

In this paper, by *graph* we always mean a directed one. Let G be a group. For a generating subset S of G, the *Cayley graph* of (G, S) is the graph Γ with G as set of vertices, and an edge from g to ga for each $g \in G, a \in S$. This is a regular graph of degree $|S|$. A regular graph $\Gamma = (V, E)$ is an *ϵ-expander* if, for each set of vertices $U \subseteq V$ with $|U| \leq |V|/2$, the set $N(U)$—of neighbours of elements of U—satisfies $|N(U) \setminus U| \geq \epsilon|U|$. (Necessarily, $\epsilon \leq 1$ in this case.) Surveys on expander graphs are available in [8, 6, 9].

For a d-regular graph Γ with adjacency matrix A, let

$$\lambda(\Gamma) = \max\{|\lambda| \ : \ \lambda \text{ is an eigenvalue of } A,\ |\lambda| \neq d\}.$$

Throughout this section, $|G|$ should be thought of as tending to infinity, whereas $|S|$ (and thus d) and ϵ should be considered constant. We will use the following known facts.[1]

Theorem 2.1. *Let $\Gamma = (V, E)$ be a finite d-regular graph.*

(1) *If Γ has loops on each vertex and Γ is an ϵ-expander, then $d - \lambda(\Gamma) \geq \epsilon^2/(4 + 2\epsilon^2)$* [6, Theorem E.7].

[1] The references given are to the surveys, where the primary references can be found.

(2) *Let $\alpha = \lambda(\Gamma)/d$ and $\hat{A} = \frac{1}{d}A$. Let* $\mathbf{u}$ *be the uniform distribution on V, and let* $\mathbf{p}$ *be an arbitrary distribution on V. Then, for each event B:*

$$\left|\Pr_{\hat{A}^m\mathbf{p}}[B] - \Pr_{\mathbf{u}}[B]\right| \leq \frac{1}{2}\|\hat{A}^m\mathbf{p} - \mathbf{u}\|_1 \leq \frac{1}{2}\sqrt{|V|}\cdot\alpha^m$$

for all m [8, Theorem 3.2].

In Item (2) of Theorem 2.1, $\hat{A}^m\mathbf{p}$ is the distribution on V corresponding to choosing a vertex according to the distribution $\mathbf{p}$, and then performing m steps of random walk on the graph, where in each step one moves to a uniformly chosen neighbour of the present vertex.[2]

Let G be a finite group, and let $g = (g_0, \dots, g_{k-1})$ be a k-tuple of generators of G. The homomorphic hash function $h_g\colon \{0,\dots,k-1\}^* \to G$ is defined by

$$h_g(b_1b_2\dots b_m) := g_{b_1}g_{b_2}\cdots g_{b_m} \in G$$

for all $b_1b_2\dots b_m \in \{0,\dots,k-1\}^*$.

The first item of the following proposition was pointed out to us by E. Breuillard.[3]

Proposition 2.2. *Let G be a finite group, and let $S = \{g_0,\dots,g_{k-1}\}$ be generators of G such that the Cayley graph of $(G, S^{\pm 1})$ is an ϵ-expander. Then:*

(1) *The Cayley graph of (G,S) is an $\epsilon/(k+1)$-expander.*

(2) *Let $m = (c(k+1)^3/\epsilon^2)\log|G|$, $c > 5/2$. Let* $\mathbf{u}$ *be the uniform distribution on G. If $v \in \{0,\dots,k-1\}^m$ is chosen uniformly at random, then for each event B:*

$$\left|\Pr[h_g(v)\in B] - \Pr_{\mathbf{u}}[B]\right| \leq \frac{1}{2}\|h_g(v) - \mathbf{u}\|_1 < \frac{1}{2|G|^{c/5-1/2}}.$$

Proof. (1) Let $\delta = \epsilon/(k+1)$. Assume that there is $U \subseteq G$ such that $|U| \leq |G|/2$ and $|US \setminus U| < \delta|U|$. Fix $s \in S$. In particular, $|Us \setminus U| < \delta|U|$, and thus

$$|Us^{-1}\cap U| = |U \cap Us^{-1}| = |(Us\cap U)s^{-1}| = |Us \cap U| \geq (1-\delta)|U|.$$

Thus, $|Us^{-1}\setminus U| < \delta|U|$, and therefore

$$\epsilon|U| \leq |US^{\pm1}\setminus U| \leq |US^{-1}\setminus U| + |US\setminus U| < k\delta|U| + \delta|U| = (k+1)\delta|U| = \epsilon|U|;$$

a contradiction.

(2) Let $\delta = \epsilon/(k+1)$. By (1), the Cayley graph of (G,S) is a δ-expander. Consider the Cayley graph of $(G, S\cup\{e\})$, where e is the neutral element of G. This is the Cayley graph of (G,S), with a loop added at each vertex. As $N(U)\setminus U$ does not become larger when adding loops, the Cayley graph of $(G,S\cup\{e\})$ is a δ-expander, too.

As the Cayley graph Γ of $(G,S\cup\{e\})$ has loops on all vertices, Theorem 2.1 applies. As $\delta \leq 1/2$,

$$k+1-\lambda(\Gamma) \geq \frac{\delta^2}{4+2\delta^2} > \frac{\delta^2}{5}.$$

Thus,

$$\frac{\lambda(\Gamma)}{k+1} < 1 - \frac{\delta^2}{5d} = 1 - \frac{\epsilon^2}{5d^3}.$$

[2] As there are loops on the vertices, one may remain at the same vertex after the step.

[3] We state and prove this observation in a slightly more general setting than the one provided by Breuillard, but the argument is identical to Breuillard's.

Let $v = b_1b_2 \dots b_m \in \{0,\dots,k-1\}^m$ be chosen uniformly at random. Then

$$h_g(v) = g_{b_1} \cdots g_{b_m}.$$

Take a uniform random walk of length m in the Cayley graph Γ of $(G, S \cup \{e\})$, starting at e, as follows:

(1) Start at e, and set $i = 1$.
(2) Repeat m times: In probability $1/(k+1)$, stay where you are, and otherwise, multiply on the right by g_{b_i} and increment i.

Let a be where we end up the above walk, and let $r = g_{b_i}g_{b_{i+1}} \cdots g_{b_m}$. Then

$$h_g(v) = g_{b_1} \cdots g_{b_m} = ag_{b_i}g_{b_{i+1}} \cdots g_{b_m} = ar.$$

By Theorem 2.1, for $\alpha = 1 - \epsilon^2/5(k+1)^3$:

$$\left|\Pr[a \in B] - \Pr_{\mathbf{u}}[B]\right| \leq \frac{1}{2}\|a - \mathbf{u}\|_1 < \frac{1}{2}\sqrt{|G|} \cdot \alpha^m$$

for all m. Multiplying both distributions by r on the right, $\mathbf{u}$ and the estimations remain unchanged, and a changes to $h_g(v)$.

Let $m = c/5 \cdot \log_{1/\alpha}|G|$. Then

$$\sqrt{|G|} \cdot \alpha^m = \sqrt{|G|} \cdot \alpha^{c/5 \cdot \log_{1/\alpha}|G|} = \sqrt{|G|}(|G|^{\log_{1/\alpha}\alpha})^{c/5} = \sqrt{|G|} \cdot |G|^{-c/5} = 1/|G|^{c/5-1/2}.$$

As

$$\log\alpha = \log(1 - \epsilon^2/5(k+1)^3) < -\epsilon^2/5(k+1)^3,$$

we have that

$$\log_{1/\alpha}|G| = \frac{\log|G|}{\log\frac{1}{\alpha}} = \frac{\log|G|}{-\log\alpha} < \frac{\log|G|}{\epsilon^2/5(k+1)^3} = \frac{5(k+1)^3}{\epsilon^2}\log|G|,$$

and m is as required. □

As its proof indicates, the following theorem can be generalized to arbitrary, not necessarily equal, numbers of given generators and random elements. We state it, though, in the form needed here.

Theorem 2.3. *Let G be a finite group, and let $g = (g_0, g_1)$ be a pair of generators of G such that the Cayley graph of $(G, \{g_0^{\pm 1}, g_1^{\pm 1}\})$ is an ϵ-expander. Assume that, for some m, if $r = (r_0, r_1) \in G^2$ is chosen uniformly at random, one can find, in a nonnegligible probability, collisions of length $O(l)$ in h_r. Then one can find, in the same probability and similar time, collisions in the original hash function h_g, of length $O(l/\epsilon^2 \cdot \log|G|)$.*

Proof. Let $m = (c \cdot 3^3/\epsilon^2)\log|G|$, with c large enough (say, 10). Take uniformly random, independent $v_0, v_1 \in \{0,1\}^m$. By Proposition 2.2, $r_0 := h_g(v_0)$ and $r_1 := h_g(v_1)$ are statistically indistinguishable from independent, uniformly random elements of G. A collision

$$h_r(b_1b_2 \cdots b_{l_1}) = h_r(c_1c_2 \cdots c_{l_2})$$

of length $l := \max\{l_1, l_2\}$ yields the collision

$$h_g(v_{b_1}v_{b_2} \cdots v_{b_{l_1}}) = h_g(v_{c_1}v_{c_2} \cdots v_{c_{l_2}})$$

of length $O(ml) = O(l/\epsilon^2 \cdot \log|G|)$. □

Let $\epsilon > 0$. Let $\mathbb{P}$ be a family of prime powers. For each $q \in \mathbb{P}$, assume that $A_0^{(q)}, A_1^{(q)} \in \mathrm{SL}_2(q)$ are generators such that the Cayley graph of $(\mathrm{SL}_2(q), \{A_0^{(q)}, A_1^{(q)}\}^{\pm 1})$ is an ϵ-expander. Then, by Theorem 2.3, the associated hash functions $h_{A^{(q)}}$ are not more secure than random hash functions $h\colon \{0,1\}^* \to \mathrm{SL}_2(q)$. In other words, the hash functions h_R with $R = (R_0, R_1) \in \mathrm{SL}_2(q)^2$ a uniformly random pair of matrices are the strongest in terms of collision resistance.

This observation is applicable in our setting for two reasons. The first is that, in all concrete proposals made thus far (e.g., [15, 16, 17]) the corresponding Cayley graph was proved to be an expander. The second, more general, is the following well known and well supported conjecture (cf. Conjecture 2.9 in [9]).

Conjecture 2.4 (Lubotzky)**.** *There is a constant $\epsilon > 0$ such that, for all prime powers q, and all generators A_0, A_1 of $\mathrm{SL}_2(q)$, the Cayley graph of $(\mathrm{SL}_2(q), \{A_0^{\pm 1}, A_1^{\pm 1}\})$ is an ϵ-expander.*

In the case where the generators A_0, A_1 are chosen at random and q is prime, this conjecture was proved to hold for randomly chosen matrices, with probability going to 1 as q increases, by Bourgain and Gamburd [2]. Breuillard, Green, Guralnick and Tao [4] have recently extended this result to q an arbitrary prime power. From another direction, Breuillard and Gamburd [3] proved that there is a set of primes q, of density 1 in the primes, for which the conjecture holds regardless of the choice of generators.

3. Collisions of linear length

The following theorem provides an algorithm for finding collisions of length $O(\log q)$ in time $O(\sqrt{q})$, for a special class of generators. This class includes a substantial portion of the concrete pairs of generators proposed in the literature, including the ones in [15] (end of §6, $i = 2$), [17], [16], and [1]. According to Lubotzky's above-mentioned Conjecture 2.4 and the discussion following it, ϵ may be viewed as a constant in the following theorem.

Theorem 3.1. *Let $A = (A_0, A_1)$ be a pair of generators of $\mathrm{SL}_2(q)$ such that $|A_0 - A_1| = 0$. If the Cayley graph of $(\mathrm{SL}_2(q), \{A_0^{\pm 1}, A_1^{\pm 1}\})$ is an ϵ-expander, then a collision on h_A of length $O(\log q/\epsilon^2)$ can be found in time $O(\sqrt{q})$.*

The remainder of this section details the proof of Theorem 3.1. Let

$$\mathcal{T} := \left\{ \begin{pmatrix} \alpha & \beta \\ 0 & \alpha^{-1} \end{pmatrix} \ : \ 0 \neq \alpha \in \mathbb{F}_q,\ \beta \in \mathbb{F}_q \right\}$$

be the subgroup of $\mathrm{SL}_2(q)$ consisting of all upper triangular matrices.

Lemma 3.2. *For generators A_0, A_1 of $\mathrm{SL}_2(q)$, the following conditions are equivalent:*

(1) $|A_0 - A_1| = 0$.

(2) *There exists $P \in \mathrm{SL}_2(q)$ and $\xi_0, \xi_1 \in \mathbb{F}_q$ such that*

$$P^{-1} A_i P = \begin{pmatrix} \xi_i & -1 \\ 1 & 0 \end{pmatrix}$$

for $i = 0, 1$.

Proof. (1) ⇒ (2): Let v be a nontrivial vector with $A_0v - A_1v = (A_0 - A_1)v = \vec{0}$. Let

$$u := A_0v = A_1v.$$

Assume that $u = \alpha v$ for some $\alpha \in \mathbb{F}_q$. Let $P \in \mathrm{SL}_2(q)$ be a matrix whose first column is v. Then

$$P^{-1}A_iP = \begin{pmatrix} \alpha & * \\ 0 & * \end{pmatrix} \in \mathcal{T}$$

for $i = 0, 1$, and thus A_0, A_1 do not generate $\mathrm{SL}_2(q)$; a contradiction.

Thus, u is linearly independent of v. Let Q be the matrix whose columns are $(-u, v)$ and let $P = |Q|^{-1}Q$. Then

$$P^{-1}A_iP = \begin{pmatrix} * & -1 \\ * & 0 \end{pmatrix},$$

and having determinant 1, we arrive at (2).

(2) ⇒ (1):

$$|A_0 - A_1| = |P^{-1}(A_0 - A_1)P| = |P^{-1}A_0P - P^{-1}A_1P| = \left|\begin{pmatrix} \xi_0 - \xi_1 & 0 \\ 0 & 0 \end{pmatrix}\right| = 0. \quad \square$$

By Lemma 3.2, we may assume that

$$A_i = \begin{pmatrix} \xi_i & -1 \\ 1 & 0 \end{pmatrix}$$

for $i = 0, 1$.

Definition 3.3. For a bitstring $v = b_1b_2 \dots b_m \in V$, we define $v^{\mathrm{r}} := b_m \dots b_2b_1$ as the reversal bitstring.

Lemma 3.4. *Let $A = (A_0, A_1)$ be a pair of elements of $\mathrm{SL}_2(q)$ with*

$$A_i = \begin{pmatrix} \xi_i & -1 \\ 1 & 0 \end{pmatrix}$$

for $i = 0, 1$. For a bitstring v let

$$\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} := h_A(v).$$

Then

$$h_A(v^{\mathrm{r}}) = \begin{pmatrix} \alpha & -\gamma \\ -\beta & \delta \end{pmatrix}.$$

Proof. By induction on $|v|$. If $|v| = 1$ then $h_A(v)$ is A_0 or A_1, both of the desired form. Assume the result holds for v. Then for each $i \in \{0, 1\}$, we have by the induction hypothesis that

$$\begin{aligned} h_A(vi) &= \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}\begin{pmatrix} \xi_i & -1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} \alpha\xi_i + \beta & -\alpha \\ \gamma\xi_i + \delta & -\gamma \end{pmatrix}, \\ h_A(iv^{\mathrm{r}}) &= \begin{pmatrix} \xi_i & -1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} \alpha & -\gamma \\ -\beta & \delta \end{pmatrix} = \begin{pmatrix} \alpha\xi_i + \beta & -\gamma\xi_i - \delta \\ \alpha & -\gamma \end{pmatrix}. \end{aligned}$$

Thus, $h_A((vi)^{\mathrm{r}}) = h_A(iv^{\mathrm{r}})$ has the desired form. □

Let

$$\mathcal{K} := \left\{ \begin{pmatrix} 1 & \beta \\ 0 & 1 \end{pmatrix} \;:\; \beta \in \mathbb{F}_q \right\}.$$

$\mathcal{K}$ is a subgroup of $\mathcal{T}$. Since $\mathcal{K}$ is abelian, hashing into $\mathcal{K}$ with two noncommuting bitstrings u, v (i.e., such that $uv \neq vu$) yields the collision

$$h_A(uv) = h_A(u)h_A(v) = h_A(v)h_A(u) = h_A(vu).$$

Proposition 3.5. *Let $A = (A_0, A_1)$ be a pair of elements of* $\mathrm{SL}_2(q)$*, with*

$$A_i = \begin{pmatrix} \xi_i & -1 \\ 1 & 0 \end{pmatrix}$$

for $i = 0, 1$. Let $b_1b_2 \dots b_m$ be a bitstring such that $h_A(b_1b_2 \dots b_m) \in \mathcal{T}$. Then for all $i \in \{0,1\}$

$$h_A(ib_m \dots b_2) \in \mathcal{T},$$

and

$$h_A(b_1b_2 \dots b_m ib_m \dots b_2), h_A(ib_m \dots b_2b_1b_2 \dots b_m) \in \mathcal{K}.$$

Proof. Let

$$h_A(v) = \begin{pmatrix} \alpha & \beta \\ 0 & \alpha^{-1} \end{pmatrix}.$$

By Lemma 3.4,

$$\begin{aligned} h_A(ib_m \dots b_2) &= h_A(i)h_A(b_m \dots b_2)h_A(b_1)h_A(b_1)^{-1} \\ &= h_A(i)h_A(b_m \dots b_2b_1)h_A(b_1)^{-1} \\ &= \begin{pmatrix} \xi_i & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \alpha & 0 \\ -\beta & \alpha^{-1} \end{pmatrix} \begin{pmatrix} \xi_i & -1 \\ 1 & 0 \end{pmatrix}^{-1} \\ &= \begin{pmatrix} * & -\alpha^{-1} \\ \alpha & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ -1 & \xi_i \end{pmatrix} \\ &= \begin{pmatrix} \alpha^{-1} & * \\ 0 & \alpha \end{pmatrix} \in \mathcal{T}. \end{aligned}$$

Moreover, we have that

$$\begin{aligned} h_A(b_1b_2 \dots b_m ib_m \dots b_2) &= h_A(b_1b_2 \dots b_m)h_A(ib_m \dots b_2) \\ &= \begin{pmatrix} \alpha & * \\ 0 & \alpha^{-1} \end{pmatrix} \begin{pmatrix} \alpha^{-1} & * \\ 0 & \alpha \end{pmatrix} \\ &= \begin{pmatrix} 1 & * \\ 0 & 1 \end{pmatrix} \in \mathcal{K}, \end{aligned}$$

and similarly for $h_A(ib_m \dots b_2b_1b_2 \dots b_m)$. □

Corollary 3.6. *Let $A = (A_0, A_1)$ be a pair of elements of* $\mathrm{SL}_2(q)$*, with*

$$A_i = \begin{pmatrix} \xi_i & -1 \\ 1 & 0 \end{pmatrix}$$

for $i = 0, 1$. Let $v = b_1 b_2 \dots b_m$ be a bitstring such that $h_A(v) \in \mathcal{T}$. Let $u = b_m \dots b_2$. Then the palindromic bitstring $uv := b_m \dots b_1 \dots b_m$ of length $2m-1$ satisfies $h_A(0uv1) = h_A(1uv0)$, a collision of length $2m+1$.

Proof. By Proposition 3.5, we have that

$$\begin{aligned} h_A(v)h_A(0uv1)h_A(u) &= h_A(v0u)h_A(v1u) \\ &= h_A(v1u)h_A(v0u) \\ &= h_A(v)h_A(0uv1)h_A(u). \end{aligned}$$

Multiplying on the right by $h_A(u)^{-1}$ and on the left by $h_A(v)$, the assertion follows. □

We can now describe our algorithm. First conjugate the given generators to matrices $B = (B_0, B_1)$ which have the form as in Lemma 3.2. As conjugation is a group isomorphism, the Cayley graph is unchanged, which thus remains an ϵ-expander. Note that the order of $\mathrm{SL}_2(q)$ is $(q-1)q(q+1) \approx q^3$. By Proposition 2.2, there exists $m \in O(\log q/\epsilon^2)$ such that the statistical distance between $h_A(v)$ (v uniformly random in $\{0,1\}^m$) and a uniformly random element of $\mathrm{SL}_2(q)$ is smaller than $1/q^2$.

Next, hash on h_B into the subgroup $\mathcal{T}$ using a meet-in-the-middle approach as done by Petit et al. [13]. We describe this approach using different, but equivalent terminology. In order to effectively hash into $\mathcal{T}$, we need an efficient encoding of the cosets of $\mathcal{T}$ in $\mathrm{SL}_2(q)$. This is given by the following proposition.

Definition 3.7. Extend the definition of the quotient $\alpha\beta^{-1}$ to the case $\beta = 0$ by declaring $\alpha \cdot 0^{-1} = \infty$ for all $\alpha \in \mathbb{F}_q$.

Proposition 3.8. *The map*

$$\begin{aligned} \mathrm{SL}_2(q)/\mathcal{T} &\longrightarrow \mathbb{F}_q \cup \{\infty\} \\ \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}\mathcal{T} &\longmapsto \alpha\gamma^{-1} \end{aligned}$$

is well defined and bijective.

Proof. Assume that

$$\begin{pmatrix} \alpha_1 & * \\ \gamma_1 & * \end{pmatrix}\mathcal{T} = \begin{pmatrix} \alpha_2 & * \\ \gamma_2 & * \end{pmatrix}\mathcal{T}.$$

Then

$$\begin{pmatrix} * & * \\ -\gamma_2 & \alpha_2 \end{pmatrix}\begin{pmatrix} \alpha_1 & * \\ \gamma_1 & * \end{pmatrix} = \begin{pmatrix} \alpha_2 & * \\ \gamma_2 & * \end{pmatrix}^{-1}\begin{pmatrix} \alpha_1 & * \\ \gamma_1 & * \end{pmatrix} \in \mathcal{T},$$

that is, $-\gamma_2\alpha_1 + \alpha_2\gamma_1 = 0$. Thus, $\alpha_1\gamma_2 = \alpha_2\gamma_1$, and the claim follows by considering the possible cases: if any of γ_1, γ_2 is 0, say, $\gamma_1 = 0$, then $\alpha_1 \neq 0$ (since the matrices are invertible), and thus $\gamma_2 = 0$, and the code of both cosets is ∞. If none of γ_1, γ_2 is 0, then the codes are $\alpha_1\gamma_1^{-1} = \alpha_2\gamma_2^{-1}$. This proves that the map is well defined.

It is clear that the map is onto. As $|\,\mathrm{SL}_2(q)/\mathcal{T}| = q + 1 = |\mathbb{F}_q \cup \{\infty\}|$, the map is bijective. □

So, to hash into $\mathcal{T}$ start generating bitstrings v in order of increasing length, together with their hash values $h_B(v)$ and store v and the code of the coset $h_B(v)\mathcal{T}$, as given by Proposition 3.8. That is, if $C = h_B(v)$ then in terms of the entries of C the code of $C\mathcal{T}$ is given by $c_{11}c_{21}^{-1}$. Search for the code of $C^{-1}\mathcal{T}$ in the set of stored codes. The code of $C^{-1}\mathcal{T}$, in terms of the entries of C, is $-c_{22}c_{21}^{-1}$. If one is found, say of $h_B(u)$, then $h_B(u)\mathcal{T} = h_B(v)^{-1}\mathcal{T}$, and therefore

$$h_B(vu)\mathcal{T} = h_B(v)h_B(u)\mathcal{T} = \mathcal{T},$$

so that we can terminate with

$$h_B(vu) \in \mathcal{T}.$$

By Proposition 2.2, for each pair u, v of bitstrings the probability that the codes of $h_B(u)\mathcal{T}$ and $h_B(v)^{-1}\mathcal{T}$ are equal is, up to an additive $O(1/q^2)$ error, the same as the probability that the codes of $r_0\mathcal{T}$ and $r_1\mathcal{T}$ are equal, for uniformly random elements r_0, r_1 of $\mathrm{SL}_2(q)$. As $|\mathrm{SL}_2(q)/\mathcal{T}| = q + 1$, this probability is $1/(q + 1)$. The additive error of $O(1/q^2)$ is negligible compared to that, thus $O(\sqrt{q})$ bitstrings suffice for the above procedure to terminate.

Suppose we have found a bitstring $b_1 \ldots b_{2m} = uv$ whose hash value lies in $\mathcal{T}$. By Corollary 3.6, the palindromic bitstring

$$w := b_{2m} \ldots b_1 \ldots b_{2m}$$

satisfies

$$h_A(0w1) = h_A(1w0);$$

a collision of length $4m + 1$, which is $O(\log q/\epsilon^2)$. This completes the proof of Theorem 3.1. □

Remark 3.9. Heuristically, there is no need to assume in Theorem 3.1 that A_0 and A_1 generate $\mathrm{SL}_2(q)$. Indeed, if they do not, then as shown in Lemma 3.2, they are simultaneously conjugate to elements of $\mathcal{T}$, and thus we can find a collision of length $\lg q$ as in Section 4.2. Thus, in any case we end up with collisions of length roughly $\lg q$ if $|A_0 - A_1| = 0$.

Remark 3.10. Note that once a string v is found that hashes into $\mathcal{K}$ (as in Proposition 3.5) one can construct preimages to the identity element by concatenting v with itself p times.

Heuristic estimations and computer experiments. Throughout this paper, in our heuristic estimations we assume that for our purposes hashes of distinct bitstrings behave as if they are independent, uniformly distributed elements of the group in question. (Unless there is an obvious obstruction, cf. Section 4.2.1.)

For the algorithm presented above, one needs that, for two of our generated matrices, C_1, C_2, the codes of $C_1\mathcal{T}$ and $C_2^{-1}\mathcal{T}$ are identical. This happens, heuristically, with probability $1/(q+1) \approx 1/q$. Thus, we need to generate about $\sqrt{q}$ matrices. To this end, it suffices to hash all bistrings of length up to $\lg \sqrt{q} \approx \lg q/2$. Having achieved that, the length of the bitstring hashing to $\mathcal{T}$ is twice that, $\lg q$, and the length of the final collision is roughly $2 \lg q$. Our experimental results suggest that this heuristic is quite precisely correct.

We have tested our algorithms for a variety of pairs p, n such that $q = p^n \approx 2^{16}, 2^{32}$. For each $N = 16, 32$, we first chose a random p in a prescribed interval $\{2^k, 2^k+1, 2^k+2, \ldots, 2^{k+1}\}$ indicated in the tables below, and then took n to be the rounded value of $N/\lg p$, so that $p^n \approx 2^N$. For each choice of N and an interval for p, we conducted 10,000 experiments where, in each experiment, we took a random $\xi_0, \xi_1 \in \mathbb{F}_q$, and applied our algorithm to the pair

$$A_0 = \begin{pmatrix} \xi_0 & -1 \\ 1 & 0 \end{pmatrix}, \quad A_1 = \begin{pmatrix} \xi_1 & -1 \\ 1 & 0 \end{pmatrix}.$$

The output of these sets of 10,000 experiments is the minimum, median, average (and standard deviation), and maximum values encountered for each of the measured quantities (work and length). For $N = 16$, we have also computed, for the same instances, the work needed to find the shortest collision (by breadth-first search enumeration) and its length.

Table 1. Results for $q \approx 2^{16}$: Minimum, median, **average** (and standard deviation), and maximum values encountered. 10,000 experiments for each range of p.

	shortest collision		**our algorithm**	
$p \in$	work	length	work	length
$\{2^1, \dots, 2^2\}$	$0.00q$	$0.38 \lg q$	$0.04\sqrt{q}$	$0.70 \lg q$
	$6.29q$	$1.12 \lg q$	$2.42\sqrt{q}$	$2.20 \lg q$
	8.75q ($9.41q$)	**1.10** $\lg q$ ($0.12 \lg q$)	**2.50**$\sqrt{q}$ ($1.30\sqrt{q}$)	**2.16** $\lg q$ ($0.24 \lg q$)
	$100.90q$	$1.38 \lg q$	$8.28\sqrt{q}$	$2.68 \lg q$
$\{2^3, \dots, 2^4\}$	$0.00q$	$0.20 \lg q$	$0.01\sqrt{q}$	$0.40 \lg q$
	$3.67q$	$1.08 \lg q$	$2.23\sqrt{q}$	$2.22 \lg q$
	5.22q ($5.14q$)	**1.05** $\lg q$ ($0.12 \lg q$)	**2.44**$\sqrt{q}$ ($1.29\sqrt{q}$)	**2.16** $\lg q$ ($0.26 \lg q$)
	$61.65q$	$1.35 \lg q$	$8.02\sqrt{q}$	$2.78 \lg q$
$\{2^7, \dots, 2^8\}$	$0.00q$	$0.20 \lg q$	$0.01\sqrt{q}$	$0.32 \lg q$
	$3.87q$	$1.08 \lg q$	$2.30\sqrt{q}$	$2.20 \lg q$
	5.40q ($5.33q$)	**1.06** $\lg q$ ($0.12 \lg q$)	**2.45**$\sqrt{q}$ ($1.30\sqrt{q}$)	**2.16** $\lg q$ ($0.26 \lg q$)
	$56.84q$	$1.34 \lg q$	$7.99\sqrt{q}$	$2.84 \lg q$
$\{2^{15}, \dots, 2^{16}\}$	$0.00q$	$0.26 \lg q$	$0.02\sqrt{q}$	$0.46 \lg q$
	$3.77q$	$1.07 \lg q$	$2.34\sqrt{q}$	$2.20 \lg q$
	5.34q ($5.29q$)	**1.05** $\lg q$ ($0.12 \lg q$)	**2.46**$\sqrt{q}$ ($1.32\sqrt{q}$)	**2.16** $\lg q$ ($0.26 \lg q$)
	$69.47q$	$1.35 \lg q$	$10.73\sqrt{q}$	$2.74 \lg q$

The results of our experiments are displayed in Tables 1 and 2. The striking observation is that, for all of these sets of parameters, and for the total 80,000 experiments conducted, none deviated substantially from our optimistic heuristic estimations. Moreover, it is clearly visible that our algorithm is not sensitive to the field characteristic p.

4. A generic short collision search algorithm

We now present a generic collision finding algorithm for $\mathrm{SL}_2(q)$ homomorphic hash functions for arbitrary q and arbitrary pairs $A = (A_0, A_1)$. Heuristically, and according to

Table 2. Results of the new algorithm for $q \approx 2^{32}$: Minimum, median, **average** (and standard deviation), and maximum values encountered. 10,000 experiments for each range of p.

$p \in$	**work**	**length**
$\{2^1, \dots, 2^2\}$	$0.04\sqrt{q}$ $2.39\sqrt{q}$ $\mathbf{2.48}\sqrt{q}$ $(1.31\sqrt{q})$ $9.03\sqrt{q}$	$1.34 \lg q$ $2.10 \lg q$ $\mathbf{2.08} \lg q$ $(0.12 \lg q)$ $2.40 \lg q$
$\{2^7, \dots, 2^8\}$	$0.02\sqrt{q}$ $2.33\sqrt{q}$ $\mathbf{2.48}\sqrt{q}$ $(1.30\sqrt{q})$ $8.51\sqrt{q}$	$1.22 \lg q$ $2.10 \lg q$ $\mathbf{2.08} \lg q$ $(0.12 \lg q)$ $2.40 \lg q$
$\{2^{15}, \dots, 2^{16}\}$	$0.03\sqrt{q}$ $2.33\sqrt{q}$ $\mathbf{2.48}\sqrt{q}$ $(1.31\sqrt{q})$ $8.27\sqrt{q}$	$1.28 \lg q$ $2.10 \lg q$ $\mathbf{2.08} \lg q$ $(0.12 \lg q)$ $2.40 \lg q$
$\{2^{31}, \dots, 2^{32}\}$	$0.03\sqrt{q}$ $2.37\sqrt{q}$ $\mathbf{2.48}\sqrt{q}$ $(1.30\sqrt{q})$ $8.50\sqrt{q}$	$1.26 \lg q$ $2.10 \lg q$ $\mathbf{2.08} \lg q$ $(0.12 \lg q)$ $2.36 \lg q$

experiments, our algorithm finds collisions of length roughly $2 \lg q / \lg \lg q$ in running time $O(\sqrt{q})$. This algorithm improves upon an algorithm of Petit et al. [13] for q a power of 2. Petit et al. demonstrate, heuristically, that their algorithm is expected to find collisions of length about $12 \mathrm{lg}^2 q$ in running time $O(\sqrt{q} \log q)$. A straightforward generalization of their algorithm to an arbitrary field size $q = p^n$ yields collisions of length about $12p\mathrm{lg}^2 q$, and a slight modification of their approach yields p times shorter collisions. We detail this approach and its mentioned refinement in Appendix A.

The basic idea of our approach is to hash with $A = (A_0, A_1)$ until we find two elements that commute. For suppose we find two distinct strings u, v whose hash values commute. Then a collision is given by $h_A(uv) = h_A(vu)$. An obvious approach would be to hash into a commutative subgroup.

Roughly speaking, our algorithm is as follows. The first step is to hash twice on h_A into the subgroup $\mathcal{T}$. In fact, we show, heuristically, that we may assume that one of the matrices A_0, A_1 is already in $\mathcal{T}$, and it suffices to hash just once into $\mathcal{T}$. This halves the amount of work, and makes it possible to reduce the length of the final collision by a factor of $\lg \lg q$. We then use the obtained matrices $C_0, C_1 \in \mathcal{T}$, to reduce the problem to hashing on h_C to find two commuting elements. As we will see, aiming for the above-mentioned subgroup $\mathcal{K}$

(this was the approach taken by Petit et al. [13]) is problematic for our approach, whereas the subgroup $\mathcal{D}$ of diagonal matrices is a good choice. In fact, we have a slightly better method, hashing directly to commuting elements, not necessarily diagonal ones.

We describe our algorithm in two phases: the first phase describes how to reduce the problem into one where A_0, A_1 are in $\mathcal{T}$, and the second phase describes how to hash on $\mathcal{T}$ to find commuting elements.

4.1. First phase: moving into $\mathcal{T}$.

In this phase we find two short bitstrings hashing into $\mathcal{T}$. Finding the first string is easy. Since conjugation is a group automorphism, collisions are preserved under conjugation. The probability that a matrix in $\mathrm{SL}_2(q)$ is diagonalizable is $1/2 - \Theta(1/q)$ [10]. Thus heuristically, A_0, A_1 or short combination thereof, call it A_2, may be assumed to be diagonalizable. In other words, there is a bitstring u_0 of constant length such that $A_2 := h_A(u_0)$ is diagonalizable.

Let $P \in \mathrm{SL}_2(q)$ be such that $P^{-1}A_2P$ is diagonal. In particular, $P^{-1}A_2P \in \mathcal{T}$. Conjugating A_0, A_1 by P, let

$$\begin{aligned} B_0 &:= P^{-1}A_0P, \\ B_1 &:= P^{-1}A_1P. \end{aligned}$$

Setting $B = (B_0, B_1)$, we have that

$$C_0 := h_B(u_0) = P^{-1}A_2P \in \mathcal{T}.$$

It remains to find a second string whose hash value on h_B lies in $\mathcal{T}$, which we can do using the meet-in-the-middle method used in the proof of Theorem 3.1. We expect the need to generate roughly $\sqrt{q}$ bitstrings in order to find strings u, v with the same code (as given by Proposition 3.8), so that the string vu hashes into $\mathcal{T}$ and

$$|vu| = |u| + |v| \approx 2\lg\sqrt{q} = \lg q.$$

Setting $u_1 := vu$ we arrive at two strings u_0, u_1 of lengths l_0 constant and $l_1 \approx \lg q$, respectively, hashing to $C_0, C_1 \in \mathcal{T}$.

4.2. Second phase: finding commuting elements in $\mathcal{T}$.

After finding strings u_0, u_1 hashing to $C_0, C_1 \in \mathcal{T}$, the next and final step is to find two strings whose hash values commute on h_C.

4.2.1. *An obstruction.* It is tempting to repeat the same procedure for h_C and the subgroup $\mathcal{K}$ of $\mathcal{T}$ of index $q-1$. Unfortunately, we encounter the following obstruction, stemming from $\mathcal{K}$ being abelian. Let

$$T_0 = \begin{pmatrix} \alpha_0 & * \\ 0 & \alpha_0^{-1} \end{pmatrix}, \quad T_1 = \begin{pmatrix} \alpha_1 & * \\ 0 & \alpha_1^{-1} \end{pmatrix}.$$

For each bitstring $b_1 \dots b_m$, the upper left entry of $T_{b_1} \dots T_{b_m}$ is

$$\alpha_{b_0} \dots \alpha_{b_m} = \alpha_0^{\nu_0(b_1 \dots b_m)} \alpha_1^{\nu_1(b_1 \dots b_m)},$$

where $\nu_0(\cdot), \nu_1(\cdot)$ denote, respectively, the number of 0-bits and the number of 1-bits in a bitstring.

On average, to have $\alpha_0^{k_0}\alpha_1^{k_1} = 1$, we need k_0 and k_1 to be roughly $\sqrt{q}$, which would increase the length of the final collision by $\sqrt{q}$, i.e. exponentially in $\lg q$.

This problem is circumvented by Petit et al. [13] by hashing roughly $\lg q$ times into $\mathcal{T}$, and then using an algorithm based on the LLL algorithm and computing discrete logarithms in $\mathbb{F}_q$ (see Appendix A). However, this has a price, both in terms of running time and the length of resulting collisions.

We propose two simpler and more efficient approaches.

4.2.2. *First solution: hashing into $\mathcal{D}$.* Instead of hashing into $\mathcal{K}$, consider the diagonal subgroup $\mathcal{D}$. To construct a collision, we need to find two strings that hash on h_C into $\mathcal{D}$. We already have one such string, namely u_0 with hash value $h_B(u_0) := h_C(0)$.

We can employ a similar meet-in-the-middle approach as in the previous phase to find a bitstring w of length roughly $\lg q$ such that $h_C(w) \in \mathcal{D}$. Note that to avoid trivialities w must not be a sequence of concatenations of u_0.

Again, to employ a meet-in-the-middle approach we need an efficient encoding of the cosets of $\mathcal{D}$ in $\mathcal{T}$, which is given by the following.

Proposition 4.1. *The map*

$$\begin{array}{rcl} \mathcal{T}/\mathcal{D} & \longrightarrow & \mathbb{F}_q \\ \begin{pmatrix} \alpha & \beta \\ 0 & \alpha^{-1} \end{pmatrix} \mathcal{D} & \longmapsto & \alpha\beta \end{array}$$

is well defined and bijective.

Proof. Assume that

$$\begin{pmatrix} \alpha_1 & \beta_1 \\ 0 & \alpha_1^{-1} \end{pmatrix} \mathcal{D} = \begin{pmatrix} \alpha_2 & \beta_2 \\ 0 & \alpha_2^{-1} \end{pmatrix} \mathcal{D}.$$

Then

$$\begin{pmatrix} \alpha_2^{-1} & -\beta_2 \\ 0 & \alpha_2 \end{pmatrix} \begin{pmatrix} \alpha_1 & \beta_1 \\ 0 & \alpha_1^{-1} \end{pmatrix} = \begin{pmatrix} \alpha_2 & \beta_2 \\ 0 & \alpha_2^{-1} \end{pmatrix}^{-1} \begin{pmatrix} \alpha_1 & \beta_1 \\ 0 & \alpha_1^{-1} \end{pmatrix} \in \mathcal{D},$$

and therefore $\alpha_2^{-1}\beta_1 - \beta_2\alpha_1^{-1} = 0$, that is, $\alpha_1\beta_1 = \alpha_2\beta_2$, and the codes are equal.

The map is onto. As $|\mathcal{T}/\mathcal{D}| = q(q-1)/(q-1) = q = |\mathbb{F}_q|$, the map is bijective. □

4.2.3. *Second solution: hashing to commuting elements of $\mathcal{T}$.* This solution, which seeks for more balanced strings whose hashes commute, turns out slightly better than the previous approach of hashing into $\mathcal{D}$. We need a code to test when two elements of $\mathcal{T}$ commute.

Proposition 4.2. *Matrices*

$$\begin{pmatrix} \alpha & \beta \\ 0 & \alpha^{-1} \end{pmatrix}, \begin{pmatrix} \gamma & \delta \\ 0 & \gamma^{-1} \end{pmatrix}$$

not equal to $\pm I$ commute if and only if $(\alpha - \alpha^{-1})\beta^{-1} = (\gamma - \gamma^{-1})\delta^{-1}$.

Proof. By direct calculation, all entries of

$$\begin{pmatrix} \alpha & \beta \\ 0 & \alpha^{-1} \end{pmatrix} \begin{pmatrix} \gamma & \delta \\ 0 & \gamma^{-1} \end{pmatrix} - \begin{pmatrix} \gamma & \delta \\ 0 & \gamma^{-1} \end{pmatrix} \begin{pmatrix} \alpha & \beta \\ 0 & \alpha^{-1} \end{pmatrix}$$

are 0, except perhaps the upper right one

$$\alpha\delta - \beta\gamma^{-1} - \gamma\beta - \delta\alpha^{-1} = \delta(\alpha - \alpha^{-1}) - \beta(\gamma - \gamma^{-1}),$$

which is 0 if and only if

$$\delta(\alpha-\alpha^{-1})=\beta(\gamma-\gamma^{-1}).$$

If β and δ are both nonzero then we can rewrite the above equation as

$$(\alpha-\alpha^{-1})\beta^{-1}=(\gamma-\gamma^{-1})\delta^{-1},$$

and the claim is proved.

If $\beta=0$ then, since $\alpha\neq\pm1$ we have that $\delta(\alpha-\alpha^{-1})=0$ implies $\delta=0$. It follows that the matrices are diagonal, and thus commute, and we have that (in the notation of Definition 3.7)

$$(\alpha-\alpha^{-1})\beta^{-1}=\infty=(\gamma-\gamma^{-1})\delta^{-1}.$$

The case $\delta=0$ is identical. □

Thus, to find two strings whose hashes on h_C commute do the following. For roughly $\sqrt{q}$ bitstrings v (that are not a power of u_0) compute

$$h_C(v)=\begin{pmatrix}\alpha & \beta\\ 0 & \alpha^{-1}\end{pmatrix}$$

and store v and the code $(\alpha-\alpha^{-1})\beta^{-1}$. If we ever encounter the code 0 or ∞ then we are done, since this matrix commutes with C_0. Assuming this rare event does not occur, find two strings u,v such that the codes of $h_C(u)$ and $h_C(v)$ are equal. We expect

$$|uv|,|vu|\approx2|v|\approx2\lg\sqrt{q}=\lg q,$$

and the overall length of the collision

$$h_C(uv)=h_C(u)h_C(v)=h_C(v)h_C(u)=h_C(vu)$$

is on average, in terms of the original hash function h_A,

$$|uv|\approx\frac{l_0+\lg q}{2}\cdot\lg q\approx\frac{1}{2}\lg^2 q.$$

The factor 1/2 comes from expecting a roughly equal number of zeros and ones.

4.3. **Compressed collisions.** In the first phase, we arrived at two strings u_0,u_1 of lengths l_0 constant and $l_1\approx\lg q$, respectively, hashing to $C_0,C_1\in\mathcal{T}$. For both the first and second solutions above, we can reduce the total collision length by exploiting the fact that u_0 is roughly $\lg q/l_0$ times shorter than u_1.

Let $C=(C_0,C_1)$. For each bitstring $b_1\dots b_k\in\{0,1\}^*$,

$$h_C(b_1\dots b_k)=C_{b_1}\cdots C_{b_k}=h_A(u_{b_1})\cdots h_A(u_{b_k})=h_A(u_{b_1}\dots u_{b_k}).$$

Define

$$\begin{aligned}\nu_0(b_1\dots b_k) &= |\{i=1,\dots,k\ :\ b_i=0\}|\\ \nu_1(b_1\dots b_k) &= |\{i=1,\dots,k\ :\ b_i=1\}|\\ \|b_1\dots b_k\|_{l_0,l_1} &= \nu_0(b_1\dots b_k)\cdot l_0+\nu_1(b_1\dots b_k)\cdot l_1.\end{aligned}$$

Then, in terms of h_A, the length of a collision $h_C(w_1)=h_C(w_2)$ is $\max\{\|w_1\|_{l_0,l_1},\|w_2\|_{l_0,l_1}\}$.

Following is an algorithm for producing finite bitstrings v such that the length $\|v\|_{l_0,l_1}$ is monotonically increasing, for $l_0<l_1$.

Algorithm 4.3.

(1) $g := \gcd(l_0, l_1)$; $k_0 := l_0/g$; $k_1 := l_1/g$.
(2) For $n = 1, \dots, k_1$:

$$S_n := \begin{cases} \{0^{n/k_0}\} & \text{if } k_0 \mid n, \\ \emptyset & \text{otherwise.} \end{cases}$$

(3) $S_{k_1} := S_{k_1} \cup \{1\}$.
(4) For $n = k_1 + 1, k_1 + 2, \dots$:

$$S_n := \{v0 \ : \ v \in S_{n-k_0}\} \cup \{v1 \ : \ v \in S_{n-k_1}\}.$$

Proposition 4.4. *Let $l_0 < l_1$ be natural numbers. In the notation of Algorithm 4.3:*

(1) *For each $v \in \{0,1\}^*$, $\|v\|_{l_0,l_1}$ is divisible by g.*
(2) *For each n, $S_n = \{v \in \{0,1\}^* \ : \ \|v\|_{l_0,l_1}/g = n\}$.*
(3) *$|S_n| = |S_{n-k_0}| + |S_{n-k_1}|$, a generalized Fibonacci sequence.*
(4) *$|S_1 \cup S_2 \cup \ldots \cup S_n| \geq \lfloor k_1/k_0 \rfloor^{\lfloor n/2k_1 \rfloor} = \lfloor l_1/l_0 \rfloor^{\lfloor gn/2l_1 \rfloor}$.*

Proof. (1) Obvious.

(2,3) By induction on n, observing that the bitstrings of length gn split into those terminating with 0 and those terminating with 1.

(4) Let $m = \lfloor n/2k_1 \rfloor$. The map

$$\begin{array}{rcl} \{1, \dots, \lfloor k_1/k_0 \rfloor\}^m & \longrightarrow & S_1 \cup S_2 \cup \ldots \cup S_n \\ (i_1, \dots, i_m) & \longmapsto & 0^{i_1} 1 0^{i_2} 1 \ldots 0^{i_m} 1 \end{array}$$

is injective. Its range is as claimed. Indeed,

$$\|0^{i_1}10^{i_2}1\ldots0^{i_m}1\|_{l_0,l_1} \leq m(l_0\lfloor k_1/k_0\rfloor + l_1) = m(l_0\lfloor l_1/l_0\rfloor + l_1) \leq m \cdot 2l_1 \leq (n/2k_1) \cdot 2l_1 = gn.$$

Apply (2). □

To find shorter collisions we use the same algorithms as before, but generate the bitstrings according to Algorithm 4.3. By item (4) of Proposition 4.4, we need that

$$\sqrt{q} \approx (l_1/l_0)^{gn/2l_1},$$

and since l_0 is constant, we have

$$\frac{1}{2}\lg q \approx \frac{gn}{2l_1}\lg\frac{l_1}{l_0} \approx \frac{gn}{2}\frac{\lg l_1}{l_1},$$

that is,

$$gn \approx \frac{l_1 \lg q}{\lg l_1} \approx \frac{\lg^2 q}{\lg\lg q}.$$

The length of the obtained collision is twice that.

Remark 4.5. The diagonalization trick in the first phase, that reduces the running time by a constant factor, is in charge of the $\lg\lg q$ factor reduction of the resulting length. It may be that the constant estimation for the minimal length of a diagonal element is not provable, even using that Cayley graph of $(\mathrm{SL}_2(q), \{A_0, A_1\})$ is an expander. The reason is that a random walk in an expander graph may miss a subset of probability $1/2$ for a logarithmic number of steps. If we aim, instead, at collisions of length $O(\log^2 q)$, then the first phase of our algorithm would be to hash twice into $\mathcal{T}$, and the estimations for running time and bitstring lengths are provable as in the previous section. We do not know whether estimations in the second (noncompressed) phase are provable. If, for two random elements A_0, A_1 of $\mathcal{T}$, the Cayley graph of $(\mathcal{T}, \{A_0^{\pm 1}, A_1^{\pm 1}\})$ is (with high probability) an expander, then they are.

4.4. **Computer experiments.**

Computer experiments are reported in Tables 3 and 4. Here too, our optimistic estimations are all validated. Indeed, our estimation $2\lg^2 q/\lg\lg q$ turns out slightly more generous than needed.

5. Linear collisions for $q = 2^n$

Faugère et al. [5], building on [14], devised a heuristic subexponential time algorithm in the case where q is a power of 2. Heuristically, for $n_0 \leq n$, their time complexity and collision length are

$$2^{\frac{\omega n \log n \log n_0}{n_0 \log(n/n_0)}} \text{ and } \frac{32n^3 3^{n_0}}{n_0},$$

respectively, where $\omega \approx 2.8$ is the matrix multiplication constant. For the collisions to have polynomial length, n_0 must be $O(\log n)$. To minimize time complexity, n_0 should be $\Theta(\log n)$. Let $n_0 = c\log n$. Then the time complexity and collision length are, very roughly,

$$2^{\frac{\omega}{c} \cdot \frac{n \log\log n}{\log n}} \text{ and } \frac{32}{c} \cdot \frac{n^{3+c\log 3}}{\log n}.$$

To compare the performance of our algorithm to that of the subexponential algorithm from a practical point of view, we have limited the length of the collision to 2^{80} bits (one terra terra bits), a generous upper bound for an acceptable message length. Then, for each $n = 64, 128, 256, \ldots, 16384$, we have computed the maximal value of n_0 for which the collision length of the subexponential algorithm is not greater than 2^{80}. For this value of n_0, the running time of the subexponential algorithm is minimal. Table 5 lists, for each of these n, the running time and collision length (rounded) for our algorithm and the subexponential one. One sees clearly that, limiting the collision length to 2^{80}, our generic algorithm is much faster in all cases, and produces much shorter collisions.

But this is not the end of the story. Petit has realized that, for q a power of 2, some of the methods of [14] and [5] can be combined with our methods from Section 3. Following is a heuristic algorithm obtaining, heuristically, collisions of linear length for *arbitrary* generators A_0, A_1 of $\mathrm{SL}_2(2^n)$. This algorithm grew out of a proposal of Petit. A matrix $E \in \mathrm{SL}_2(q)$ is *orthogonal* if $EE^{\mathrm{t}} = I$. The orthogonal matrices in $\mathrm{SL}_2(2^n)$ are precisely matrices of the form

$$E = \begin{pmatrix} \alpha+1 & \alpha \\ \alpha & \alpha+1 \end{pmatrix},$$

where $\alpha \in \mathbb{F}_{2^n}$ [14]. In particular, these matrices are symmetric and satisfy $E^2 = I$.

Let A_0, A_1 be generators of $\mathrm{SL}_2(2^n)$. Let

$$B_0 := A_0A_1;\ B_1 := A_1A_0.$$

It suffices to find a collision for (B_0, B_1). The traces of B_0 and B_1 are equal. By the proof of [14, Lemma 2], there are several possibilities:

TABLE 3. Results for $q \approx 2^{16}$: Minimum, median, **average** (and standard deviation), and maximum values encountered. 10,000 experiments for each range of p. $L := \lg^2 q / \lg\lg q$.

	shortest collision		**diagonalizable**	**shortest triangular**		**compressed search**	
$p \in$	work	length	length	work	length	work	length
$\{2^1, \ldots, 2^2\}$	$0.00q$	$0.31 \lg q$	1.00	$0.03\sqrt{q}$	$0.31 \lg q$	$0.00\sqrt{q}$	$0.08L$
	$3.66q$	$1.07 \lg q$	1.00	$2.41\sqrt{q}$	$1.06 \lg q$	$2.68\sqrt{q}$	$1.34L$
	$\mathbf{5.38}q$ ($5.32q$)	$\mathbf{1.05} \lg q$ ($0.11 \lg q$)	**1.40** (0.79)	$\mathbf{2.66}\sqrt{q}$ ($1.51\sqrt{q}$)	$\mathbf{1.04}L$ ($0.13 \lg q$)	$\mathbf{2.89}\sqrt{q}$ ($1.54\sqrt{q}$)	$\mathbf{1.36}L$ ($0.30L$)
	$54.98q$	$1.32 \lg q$	5.00	$10.79\sqrt{q}$	$1.32 \lg q$	$12.13\sqrt{q}$	$2.98L$
$\{2^3, \ldots, 2^4\}$	$0.00q$	$0.27 \lg q$	1.00	$0.03\sqrt{q}$	$0.20 \lg q$	$0.00\sqrt{q}$	$0.08L$
	$3.71q$	$1.08 \lg q$	1.00	$2.45\sqrt{q}$	$1.04 \lg q$	$2.60\sqrt{q}$	$1.32L$
	$\mathbf{5.35}q$ ($5.34q$)	$\mathbf{1.05} \lg q$ ($0.11 \lg q$)	**1.41** (0.79)	$\mathbf{2.68}\sqrt{q}$ ($1.54\sqrt{q}$)	$\mathbf{1.04}L$ ($0.13 \lg q$)	$\mathbf{2.81}\sqrt{q}$ ($1.51\sqrt{q}$)	$\mathbf{1.34}L$ ($0.32L$)
	$58.57q$	$1.35 \lg q$	6.00	$10.34\sqrt{q}$	$1.35 \lg q$	$11.56\sqrt{q}$	$3.02L$
$\{2^7, \ldots, 2^8\}$	$0.00q$	$0.20 \lg q$	1.00	$0.03\sqrt{q}$	$0.21 \lg q$	$0.00\sqrt{q}$	$0.06L$
	$3.96q$	$1.08 \lg q$	1.00	$2.40\sqrt{q}$	$1.06 \lg q$	$2.64\sqrt{q}$	$1.34L$
	$\mathbf{5.54}q$ ($5.41q$)	$\mathbf{1.06} \lg q$ ($0.12 \lg q$)	**1.41** (0.79)	$\mathbf{2.67}\sqrt{q}$ ($1.57\sqrt{q}$)	$\mathbf{1.04}L$ ($0.14 \lg q$)	$\mathbf{2.81}\sqrt{q}$ ($1.48\sqrt{q}$)	$\mathbf{1.34}L$ ($0.32L$)
	$49.42q$	$1.34 \lg q$	5.00	$33.43\sqrt{q}$	$1.58 \lg q$	$9.88\sqrt{q}$	$2.86L$
$\{2^{15}, \ldots, 2^{16}\}$	$0.00q$	$0.26 \lg q$	1.00	$0.01\sqrt{q}$	$0.19 \lg q$	$0.00\sqrt{q}$	$0.06L$
	$3.82q$	$1.07 \lg q$	1.00	$2.43\sqrt{q}$	$1.06 \lg q$	$2.62\sqrt{q}$	$1.32L$
	$\mathbf{5.41}q$ ($5.32q$)	$\mathbf{1.06} \lg q$ ($0.11 \lg q$)	**1.41** (0.80)	$\mathbf{2.66}\sqrt{q}$ ($1.51\sqrt{q}$)	$\mathbf{1.04}L$ ($0.13 \lg q$)	$\mathbf{2.82}\sqrt{q}$ ($1.52\sqrt{q}$)	$\mathbf{1.34}L$ ($0.32L$)
	$51.93q$	$1.32 \lg q$	5.00	$10.13\sqrt{q}$	$1.32 \lg q$	$10.52\sqrt{q}$	$2.88L$

TABLE 4. Results for $q \approx 2^{32}$: Minimum, median, **average** (and standard deviation), and maximum values encountered. 10,000 experiments for each range of p. $L := \lg^2 q / \lg \lg q$.

	diagonalizable	**shortest triangular**		**compressed search**	
$p \in$	length	work	length	work	length
$\{2^1, \ldots, 2^2\}$	1.00	$0.04\sqrt{q}$	$0.66 \lg q$	$0.03\sqrt{q}$	$0.58L$
	1.00	$2.44\sqrt{q}$	$1.03 \lg q$	$2.98\sqrt{q}$	$1.34L$
	1.41 (0.80)	$\mathbf{2.69}\sqrt{q}$ $(1.54\sqrt{q})$	$\mathbf{1.02} \lg q$ $(0.06 \lg q)$	$\mathbf{3.23}\sqrt{q}$ $(1.78\sqrt{q})$	$\mathbf{1.38}L$ $(0.22L)$
	6.00	$10.38\sqrt{q}$	$1.17 \lg q$	$10.66\sqrt{q}$	$2.44L$
$\{2^7, \ldots, 2^8\}$	1.00	$0.02\sqrt{q}$	$0.56 \lg q$	$0.00\sqrt{q}$	$0.18L$
	1.00	$2.39\sqrt{q}$	$1.02 \lg q$	$2.92\sqrt{q}$	$1.34L$
	1.41 (0.81)	$\mathbf{2.63}\sqrt{q}$ $(1.51\sqrt{q})$	$\mathbf{1.02} \lg q$ $(0.07 \lg q)$	$\mathbf{3.19}\sqrt{q}$ $(1.74\sqrt{q})$	$\mathbf{1.38}L$ $(0.22L)$
	5.00	$10.07\sqrt{q}$	$1.21 \lg q$	$11.19\sqrt{q}$	$2.42L$
$\{2^{15}, \ldots, 2^{16}\}$	1.00	$0.04\sqrt{q}$	$0.65 \lg q$	$0.00\sqrt{q}$	$0.18L$
	1.00	$2.43\sqrt{q}$	$1.03 \lg q$	$2.92\sqrt{q}$	$1.34L$
	1.40 (0.79)	$\mathbf{2.69}\sqrt{q}$ $(1.55\sqrt{q})$	$\mathbf{1.02} \lg q$ $(0.07 \lg q)$	$\mathbf{3.19}\sqrt{q}$ $(1.77\sqrt{q})$	$\mathbf{1.38}L$ $(0.22L)$
	5.00	$10.26\sqrt{q}$	$1.19 \lg q$	$10.52\sqrt{q}$	$2.46L$
$\{2^{31}, \ldots, 2^{32}\}$	1.00	$0.01\sqrt{q}$	$0.51 \lg q$	$0.00\sqrt{q}$	$0.14L$
	1.00	$2.43\sqrt{q}$	$1.03 \lg q$	$2.95\sqrt{q}$	$1.34L$
	1.42 (0.80)	$\mathbf{2.67}\sqrt{q}$ $(1.54\sqrt{q})$	$\mathbf{1.02} \lg q$ $(0.06 \lg q)$	$\mathbf{3.21}\sqrt{q}$ $(1.75\sqrt{q})$	$\mathbf{1.38}L$ $(0.22L)$
	6.00	$10.61\sqrt{q}$	$1.17 \lg q$	$11.17\sqrt{q}$	$2.40L$

Table 5. Generic collision search versus subexponential collision search.

	subexponential algorithm		our algorithm	
q	work	length	work	length
2^{64}	2^{143}	2^{80}	2^{32}	2^{10}
2^{128}	2^{137}	2^{80}	2^{64}	2^{12}
2^{256}	2^{202}	2^{80}	2^{128}	2^{14}
2^{512}	2^{344}	2^{80}	2^{256}	2^{16}
2^{1024}	2^{625}	2^{80}	2^{512}	2^{18}
2^{2048}	2^{1181}	2^{80}	2^{1024}	2^{20}
2^{4096}	2^{2292}	2^{80}	2^{2048}	2^{21}
2^{8192}	2^{4532}	2^{80}	2^{4096}	2^{23}
2^{16384}	2^{9093}	2^{80}	2^{8192}	2^{25}

(1) Certain (rare) pathologies happen,[4] in which there are collisions of length 2, and we are done.
(2) B_0, B_1 are simultaneously conjugate to upper triangular matrices, so by Section 4.2 we can find a collision of length $\lg q$ in time $\sqrt{q}$. This case is also rare for random generators.
(3) In the remaining, main case, B_0, B_1 can be simultaneously conjugated to a pair of the form
$$C, C^{\mathrm{t}},$$
i.e., such that the second matrix is the transpose of the first. It suffices to find a collision for (C, C^{t}). This is the only case remaining to be dealt with.

By [14, Lemma 8], we can find an *orthogonal* matrix E such that $ECE = C^{\mathrm{t}}$. Thus,

$$\begin{aligned} CE &= EC^{\mathrm{t}}, \text{ and} \\ C^{\mathrm{t}}E &= EC. \end{aligned}$$

Consider the pair (CE, C). By the above-mentioned special form of orthogonal matrices, $|E - I| = 0$, and thus

$$|CE - C| = |C(E - I)| = |C| \cdot |E - I| = 0.$$

Transforming a collision for (CE, C) to one for (C, C^t) is possible if the number of 0's is either even in both strings or odd in both strings. In this case, using that $CE = EC^{\mathrm{t}}$, $C^{\mathrm{t}}E = EC$, and $E^2 = I$, the E's can be pushed to the left, transposing the matrices C, C^{t} on their way, and vanishing when meeting other E's. If an E remains (necessarily, on both sides), it can be cancelled from both sides. Thus, heuristically, we need two collisions for (CE, C) to conclude.

[4] Rare pathologies are possible if A_0, A_1 are chosen in very special form, see the proof of [14, Lemma 2].

By the proof of Lemma 3.2, there are two cases to consider: CE and C are simultaneously conjugate to either upper triangular matrices or to matrices of the form

$$\begin{pmatrix} \xi_i & 1 \\ 1 & 0 \end{pmatrix}.$$

In the former case, by Section 4.2, we can find collisions of the prescribed form of length $\lg q$ in time $\sqrt{q}$. In the latter, main case, by Corollary 3.6 it suffices to hash with $h = h_{(CE,C)}$ once (in time $\sqrt{q}$ and string length $\lg q$) into an upper triangular matrix, say $h(b_1 \ldots b_m) \in \mathcal{T}$. By Corollary 3.6,

$$h(0b_m \ldots b_1 \ldots b_m 1) = h(1b_m \ldots b_1 \ldots b_m 0).$$

As the number of 0 bits in both strings of this collision is equal, this collision can be transformed into one for (C, C^{t}), and we are done.

To illustrate this algorithm in the main case, assume that A_0, A_1 are given. Then:

(1) Set $B_0 := A_0A_1,\ B_1 := A_1A_0$.
(2) Find a matrix P such that $P^{-1}B_0P = C,\ P^{-1}B_1P = C^{\mathrm{t}}$, for some matrix C.
(3) Find an orthogonal matrix E such that $CE = EC^{\mathrm{t}}$.
(4) Find a matrix Q such that

$$D_0 := Q^{-1}(CE)Q = \begin{pmatrix} \xi_0 & 1 \\ 1 & 0 \end{pmatrix},\quad D_1 := Q^{-1}CQ = \begin{pmatrix} \xi_1 & 1 \\ 1 & 0 \end{pmatrix}.$$

(5) Find a bitstring $b_1 \ldots b_m$ such that $D_{b_1} \cdots D_{b_m} \in \mathcal{T}$, so that

$$D_0D_{b_m} \cdots D_{b_1} \cdots D_{b_m}D_1 = D_1D_{b_m} \cdots D_{b_1} \cdots D_{b_m}D_0.$$

For example, assume that $m = 3$ and $b_1 \ldots b_m = 011$. Then

$$D_0D_0D_1D_1D_1D_0D_1 = D_1D_0D_1D_1D_1D_0D_0,$$

and in terms of CE and C,

$$CECECCCCEC = CCECCCCECE.$$

Moving the E's to the left, using $CE = EC^{\mathrm{t}}$, $C^{\mathrm{t}}E = EC$, and $E^2 = I$, we have that

$$EC^{\mathrm{t}}CC^{\mathrm{t}}C^{\mathrm{t}}C^{\mathrm{t}}C^{\mathrm{t}}C = EC^{\mathrm{t}}C^{\mathrm{t}}CCCCC^{\mathrm{t}},$$

and thus

$$C^{\mathrm{t}}CC^{\mathrm{t}}C^{\mathrm{t}}C^{\mathrm{t}}C^{\mathrm{t}}C = C^{\mathrm{t}}C^{\mathrm{t}}CCCCC^{\mathrm{t}}.$$

In terms of B_0 and B_1, we have that

$$B_1B_0B_1B_1B_1B_1B_0 = B_1B_1B_0B_0B_0B_0B_1,$$

and in terms of A_0 and A_1,

$$A_1A_0A_0A_1A_1A_0A_1A_0A_1A_0A_1A_0A_0A_1 = A_1A_0A_1A_0A_0A_1A_0A_1A_0A_1A_0A_1A_1A_0.$$

The first reduction doubles the collision length. All other reductions preserve the collision length. Thus, we expect collision lengths of the algorithm to be roughly

$$2 \cdot 2\lg q = 4\lg q.$$

5.1. **Computer experiments.** The results for $q = 2^{16}, 2^{32}$ are very similar to those in Tables 1 and 2, with the only difference that, as expected, the collision length is doubled. Results of experiments for $q = 2^{40}$ are provided in Table 6. Here too, our heuristic estimations are confirmed, and even generous. The standard deviation of the collision length is very small, and is expected to converge to 0 as q increases.

Table 6. Results of the new algorithm for $q = 2^{40}$, 10,000 experiments.

$q = 2^{40}$	**work**	**length**
Minimum	$0.02\sqrt{q}$	$2.76 \lg q$
Median	$2.39\sqrt{q}$	$4.16 \lg q$
Average (and standard deviation)	$\mathbf{2.48}\sqrt{q}$ $(1.31\sqrt{q})$	$\mathbf{4.12} \lg q$ $(0.2 \lg q)$
Maximum	$8.32\sqrt{q}$	$4.54 \lg q$

Acknowledgments. We thank Alexei Belov, Alex Lubotzky and Terrence Tao for useful information about expander Cayley graphs. We also thank Emmanuel Breuillard for pointing out Proposition 2.2(1) to us, and for his permission to include it here. We owe special thanks to Christophe Petit, for numerous discussions about the problem studied here and the known methods, for useful advice that helped improve the presentation of this paper, and for his proposals that led to the generic linear collision search algorithm for even q. This research was initiated when the second named author visited Simon Blackburn, Carlos Cid, and the first named author at Royal Holloway, University of London. This author thanks his hosts for their kind hospitality.

References

[1] K. Abdukhalikov, C. Kim, On the security of the hashing scheme based on SL_2, FSE '98, Lecture Notes in Computer Science 1372 (1998), 93–102.

[2] J. Bourgain, A. Gamburd, Uniform expansion bounds for Cayley graphs of $\mathrm{SL}_2(\mathbb{F}_p)$, Annals of Mathematics 167 (2008), 625–642.

[3] E. Breuillard, A. Gamburd, Strong uniform expansion in $\mathrm{SL}(2,p)$, Geometric Functional Analysis 20 (2010), 1201–1209.

[4] E. Breuillard, B. Green, R. Guralnick, T. Tao, Expansion in finite sumple groups of Lie type, preprint.

[5] J. Faugère, L. Perret, C. Petit, and G. Renault, New subexponential algorithms for factoring in $\mathrm{SL}_2(\mathbb{F}_{2^n})$, preprint.
`http://eprint.iacr.org/2011/598.pdf`

[6] O. Goldreich, Computational Complexity: A Conceptual Perspective, Cambridge University Press, 2008.

[7] M. Grassl, I. Ilić, S. Magliveras, R. Steinwandt, Cryptanalysis of the Tillich-Zémor hash function, Journal of Cryptolgy 24 (2011), 148–156.

[8] S. Hoory, N. Linial and A. Wigderson, Expander graphs and their applications, Bulletin of the American Mathematical Society 43 (2006), 439–561.

[9] A. Lubotzky, Expander graphs in pure and applied mathematics, Bulletin of the American Mathematical Society 49 (2012), 113–162.

[10] V. Naik (moderator), Element structure of special linear group of degree two over a finite field, Groupprops, The Group Properties Wiki.
`http://groupprops.subwiki.org/wiki/Element_structure_of_special_linear_group_of_degree_two_over_a_finite_field`

[11] C. Petit, J. Quisquater, Preimages for the Tillich–Zémor hash function, SAC '10, Lecture Notes in Computer Science 6544 (2010), 282–301.

[12] C. Petit, J. Quisquater, *Rubik's for cryptographers*, Notices of the American Mathematical Society 61 (2013), 703–709.

[13] C. Petit, J. Quisquater, J. Tillich, G. Zémor, Hard and easy components of collision search in the Zémor-Tillich hash function: new attacks and reduced variants with equivalent security, CT-RSA '09, Lecture Notes in Computer Science 5473 (2009), 182–194.

[14] C. Petit, Towards Factoring Towards factoring in $\mathrm{SL}_2(2^n)$, Design, Codes and Cryptography, to appear.

[15] J. Tillich, G. Zémor, Group-theoretic hash functions, Algebraic Coding, First French-Israeli Workshop, Lecture Notes in Computer Science 781 (1994), pages 90–110.

[16] J. Tillich, G. Zémor, Hashing with SL_2, CRYPTO '94, Lecture Notes in Computer Science 839 (1991), 508–511.

[17] G. Zémor, Hash functions and graphs with large girths, Eurocrypt '91, Lecture Notes in Computer Science 547 (1991), pages 508–511.

Appendix A. The Petit–Quisquater–Tillich–Zémor algorithm

For the reader's convenience, we outline the generic algorithm of Petit, Quisquater, Tillich and Zémor [13] for finding collisions for q even. We describe their algorithm in a simplified language, generalize it to $p \geq 2$, and find optimal parameters: collisions of length $\approx 12\mathrm{lg}^2 q$ in time $O(\sqrt{q}\lg q)$. This ignores the complexity of the second phase (discrete logarithms and LLL) of their attack, which we assume is smaller than $\sqrt{q}$. Setting their parameters so as to reduce the running time below $O(\sqrt{q}\lg q)$ would render the length of the resulting collisions superpolynomial in $\lg q$.

A.1. **First phase: hashing into $\mathcal{T}$.** The performance estimations of this phase can be proved, asymptotically, as in Section 3. By Section 4.1, in time roughly $\sqrt{q}$ one can hash once into $\mathcal{T}$ with a string of length about $\lg q$. Doing this $N := \lg q$ times, we obtain (in time $\sqrt{q}\lg q$), bitstrings $w_1, \dots, w_N$ each of length about $\lg q$ such that

$$h_A(w_1), \dots, h_A(w_N) \in \mathcal{T}.$$

A.2. **Second phase: hashing into $\mathcal{K}$.** Denote by $\lambda_1, \dots, \lambda_N$ the upper left entries of $h_A(w_1), \dots, h_A(w_N)$, respectively.

Computing N discrete logarithms in $\mathbb{F}_q$ and using the LLL algorithm, find nonnegative integers $k_1, \dots, k_N$, with $\sqrt{k_1^2 + \dots + k_N^2}$ as small as possible, such that

$$\lambda_1^{k_1} \dots \lambda_N^{k_N} = 1.$$

Taking all possibilities $k_i \in \{0, 1\}$, $\lambda_1^{k_1} \dots \lambda_N^{k_N}$ takes about

$$2^N \approx 2^{\lg q} = q$$

values. Thus, it is expected (although, thus far, unproved) that the solution returned by the LLL algorithm satisfies

$$\sqrt{k_1^2 + \dots + k_N^2} \approx \sqrt{N} \approx \sqrt{\lg q}.$$

Let

$$v = w_1^{k_1} w_2^{k_2} \dots w_N^{k_N},$$

where exponentiation denotes string concatenation. By the Cauchy–Schwartz inequality,

$$|v| = k_1 \cdot |w_1| + \ldots + k_N \cdot |w_N| \le \sqrt{k_1^2 + \ldots + k_N^2} \cdot \sqrt{|w_1|^2 + \ldots + |w_N|^2},$$

with the right hand side being $\approx \sqrt{N} \cdot \sqrt{N \mathrm{lg}^2 q} = N \lg q = \lg^2 q$.

Now,

$$h(v) \begin{pmatrix} 1 \\ 0 \end{pmatrix} = h(w_1)^{k_1} \ldots h(w_n)^{k_n} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \lambda_1^{k_1} \ldots \lambda_n^{k_n} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix},$$

that is, for some $\beta \in \mathbb{F}_q$,

$$h(v) = \begin{pmatrix} 1 & \beta \\ 0 & 1 \end{pmatrix}.$$

As $q = p^n$, we have that

$$h(v^p) = \begin{pmatrix} 1 & p\beta \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},$$

that is, v^p and the empty message hash to the same value. We have that

$$|v^p| = p \cdot |v| \approx p \mathrm{lg}^2 q.$$

This completes our description of the Petit–Quisquater–Tillich–Zémor algorithm.

Note that p may be exponential in the security parameter. To obtain shorter collisions, note that in the definition of v, if u is obtained by any permutation of the order of the $k_1 + \ldots + k_N$ subwords w_i in the word $v = w_1^{k_1} \ldots w_N^{k_N}$, we still have by the same argument that

$$h(u) = \begin{pmatrix} 1 & \gamma \\ 0 & 1 \end{pmatrix}$$

for some $\gamma \in \mathbb{F}_q$. Thus, $h(v)$ *commutes with* $h(u)$, and we arrive at the collision

$$h(uv) = h(u)h(v) = h(v)h(u) = h(vu),$$

whose length is about $2 \lg^2 q$. Moreover, assuming for example that k_1 and k_2 are nonzero, let

$$w = w_1^{k_1 - 1} w_2^{k_2 - 1} \ldots w_N^{k_N}$$

Taking $v = w_1 w_2 w$ and $u = w_2 w_1 w$, we know that

$$h(vw_2w_1)h(w) = h(vu) = h(uv) = h(uw_1w_2)h(w),$$

and therefore

$$h(vw_2w_1) = h(uw_1w_2),$$

a collision of length roughly

$$|v| + |w_1| + |w_2| \approx \mathrm{lg}^2 q + 2 \lg q \approx \mathrm{lg}^2 q.$$

The algorithms presented in Section 4 are faster, and provide shorter collisions. We stress that, unlike the Petit et al. algorithm, our generic algorithm do not provide bitstrings hashing to the identity matrix (see, however, Remark 3.10).

Appendix B. The impossibility of palindromic collisions for $p > 2$

Let $q = 2^n$ and let α be a primitive element of $\mathbb{F}_{2^n}$. Let

$$A_0 = \begin{pmatrix} \alpha & -1 \\ 1 & 0 \end{pmatrix}, \quad A_1 = \begin{pmatrix} \alpha+1 & -1 \\ 1 & 0 \end{pmatrix}.$$

Grassl, Ilić, Magliveras, and Steinwandt [7] provide, in this case, an efficient algorithm for finding palindromes $v \in \{0,1\}^*$ of length $2n$ such that the palindromes $0v0$ and $1v1$ hash to the same value under h_A. This implies that the proposal in [16] is insecure. Grassl et al.'s method does not generalize in any conceivable way to odd prime powers q. In fact, we show here that for q odd there are *no* palindromes v such that $0v0$ and $1v1$ form a collision. Throughout, we write h for h_A.

Proposition B.1. *Let $v \in \{0,1\}^*$ be a palindrome. Then*

(1) *$h(v)$ is of the form* $\begin{pmatrix} a & b \\ -b & d \end{pmatrix}$.

(2) $h(0v0) - h(1v1) = \begin{pmatrix} -2a\alpha - a - 2b & a \\ -a & 0 \end{pmatrix}$.

(3) *If $p > 2$ then $h(0v0) \neq h(1v1)$.*

Proof. (1) We proceed by induction on the length of v. The induction base consists of $|v| = 1$ and $|v| = 2$. If $|v| = 1$, i.e., $v = \beta \in \{0,1\}$, then

$$h(v) = \begin{pmatrix} \alpha+\beta & -1 \\ 1 & 0 \end{pmatrix}$$

has the desired form.

Note that by direct calculation,

(1)
$$\begin{pmatrix} \alpha+\beta & -1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} a & b \\ -b & d \end{pmatrix}\begin{pmatrix} \alpha+\beta & -1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} a(\alpha+\beta)^2 + 2b(\alpha+\beta) - d & -(a\alpha + a\beta + b) \\ a\alpha + a\beta + b & -a \end{pmatrix}.$$

By Equation (1), we have in particular (for $a = d = 1$, $b = 0$) that, for each $\beta \in \{0,1\}$,

$$h(\beta\beta) = A_\beta I A_\beta$$

has the desired form. This completes the verification of the induction base.

Induction step: assume that

$$h(v) = \begin{pmatrix} a & b \\ -b & d \end{pmatrix}.$$

Then by Equation (1) $h(\beta v\beta) = A_\beta h(v) A_\beta$ has the desired form for each $\beta \in \{0,1\}$.

(2,3) Since v is a palindrome, we have by the above calculation that

$$h(0v0) - h(1v1) = \begin{pmatrix} -2a\alpha - a - 2b & a \\ -a & 0 \end{pmatrix}.$$

Hence, for $h(0v0) = h(1v1)$ to hold, a must be 0. This in turn implies that $2b = 0$, which for $p > 2$ implies that $b = 0$. Thus, $1 = \det(h(v)) = ad + b^2 = 0$, a contradiction. □

Interestingly, it is pointed out in [7] that, for q a power of 2 and a palindrome v, $h(0v1) = h(1v0)$ is equivalent to $h(0v0) = h(1v1)$. For q odd, we proved that $h(0v0) = h(1v1)$ is impossible (Proposition B.1), but that $h(0v1) = h(1v0)$ is provably possible (Theorem 3.1)!

Technische Universität Darmstadt, Fachbereich Informatik, Kryptographie und Computeralgebra, Hochschulstrasse 10, 64289 Darmstadt, Germany

Department of Mathematics, Bar Ilan University, 5290002 Ramat Gan, Israel
E-mail address: tsaban@math.biu.ac.il
URL: http://www.cs.biu.ac.il/~tsaban