Bounding the security of the Bromberg-Shpilrain-Vdovina hash function

Eilidh McKemmie*

November 2026

Abstract

The Bromberg-Shpilrain-Vdovina hash functions are a family of Cayley hash functions indexed by an integer parameter $k \geq 2$. Hashes are matrices in $SL_2(p)$ where p is a large prime. We present an attack which finds collisions of length $O(\log p)$ whose average-case time complexity is $O(p^{0.28})$ when $k = 2$ and $O(p^{0.36})$ when $k = 3$ under an assumption in the spirit of Zaremba's Conjecture on continued fractions. This is an improvement on the more general attacks of Mullan-Tsaban and Slaughter which have average-case time complexity $O(\sqrt{p})$. For any k, the average-case time complexity of our attack is $O(p^{1-\delta_k+o(1)})$ where δ_k is the Hausdorff dimension of a certain Cantor set. This tends to $O(\sqrt{p})$ as $k \to \infty$, so our attack has time complexity comparable to Mullan-Tsaban and Slaughter's attacks as k grows. We give examples of collisions for large primes found by our attack when $k = 2, 3$.

1 Introduction

Let $k \geq 1$ be an integer and let

$$A_k = \begin{pmatrix} 1 & k \\ 0 & 1 \end{pmatrix}, \qquad B_k = \begin{pmatrix} 1 & 0 \\ k & 1 \end{pmatrix}.$$

Let $H_k : \{0,1\}^* \to \mathrm{SL}_2(p)$ be defined by the following:

$$H(\emptyset) = I, \quad H_k(w0) = H_k(w)A_k, \quad H_k(w1) = H_k(w)B_k.$$

This is a Cayley hash function. When $k = 1$, this hash function was proposed by Zémor [23], and an efficient collision attack was found shortly afterwards by Tillich and Zémor [21]. Bromberg, Shpilrain and Vdovina [3] proposed using the hash function with $k = 2$ or $k = 3$, or even larger values of k. This proposal has not been broken.

*ORCID 0000-0002-5031-6412, eilidh.mckemmie@kean.edu,
Department of Mathematical Sciences, Kean University, Union, NJ, USA

There have been several attacks on Cayley hash functions which find collisions using palindromes or similar word reversal. Grassl et al [8] gave the original palindrome attack on the Tillich-Zémor hash function proposed in [22] (see [11] for another description of this attack). This attack was extended to pre-image and second pre-image attacks by Petit and Quisquater [14]. Mullan and Tsaban [13] use word reversal to construct collisions for Cayley hash functions in $SL_2(q)$ of length $O(\log q)$ is $O(\sqrt{q})$ where q is a power of 2. Recently, Slaughter [20] extended this result to odd prime q, and showed that no Cayley hash function on a pair of 2×2 generators over the field of order p has collision resistance greater than $O(\sqrt{p})$.

Mullan-Tsaban and Slaughter's attacks [13, 20] find collisions of length $O(\log p)$ in $O(\sqrt{p})$ time. We present an attack which finds collisions of length $O(\log p)$ in $O(p^{1-\delta_k+o(1)})$ time assuming a Zaremba-type heuristic, where δ_k is the Hausdorff dimension of a certain Cantor set. When $k = 2$ our attack has average-case time complexity $O(p^{0.28})$, and when $k = 3$ it is $O(p^{0.36})$. The time complexity of our attack tends to $O(\sqrt{p})$ as $k \to \infty$, so its advantages over Mullan-Tsaban and Slaughter vanish as k grows. We give examples of collisions for large primes found by our attack when $k = 2, 3$. Our attack uses word reversal and, like [21], it is a lifting attack using the fact that the Euclidean algorithm can factor target matrices which are related to a Cantor set whose dimension is known for $k = 2, 3$.

2 The monoid generated by A_k, B_k

Let $\mathcal{M}_k$ be the monoid generated by A_k, B_k.

Proposition 2.1.

$$\mathcal{M}_k \subseteq \left\{ \begin{pmatrix} k^2a+1 & kb \\ kc & k^2d+1 \end{pmatrix} : a, b, c, d \geq 0 \right\}.$$

Proof. This can be proved by a simple induction argument for any value of k. We note that when $k = 2$ the submonoid is free by Sanov [17]. □

Taking $k = 1$ in the hash function construction gives a hash function proposed by Zémor [23]. The monoid $\mathcal{M}_1$ is the monoid of all invertible 2×2 matrices with non-negative integer matrices, which leaves Zémor's hash function open to a lifting attack which efficiently finds collisions [21]. The attack uses the fact that matrices in $\mathcal{M}_1$ can be factored as products of A_1, B_1 using the Euclidean algorithm. This reduces the problem of finding pre-images to finding an appropriate lift of the matrix in $\mathrm{SL}_2(p)$ to $\mathrm{SL}_2(\mathbb{Z})$ which is then factored using the Euclidean algorithm. This collision attack can be turned into a pre-image attack when the attacker has an oracle for solving the integer factorization problem [12]

Bromberg, Shpilrain and Vdovina propose using $k > 1$ for hashing precisely because $\mathcal{M}_k$ does not have such a clean description for $k > 1$. Indeed, their Problem 1 asks for a description of the shape of matrix entries in this monoid.

One description of the matrices in $\mathcal{M}_k$ which may prove useful involves the Euclidean algorithm. If $\begin{pmatrix} x \\ y \end{pmatrix}$ appears as a column of a matrix in $\mathcal{M}_k$ then it arises by applying a word in A_k, B_k to either $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ or $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$. Since

$$A_k \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x+ky \\ y \end{pmatrix}, \qquad B_k \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x \\ y+kx \end{pmatrix},$$

we observe that $\begin{pmatrix} x \\ y \end{pmatrix}$ appears as a column of a matrix in $\mathcal{M}_k$ exactly when the quotients produced by the Euclidean algorithm applied to x and y are all multiples of k.

We can translate this into the language of continued fractions. Any rational number $\frac{x}{y} \in (0,1)$ may be written as a unique simple continued fraction $[a_1, a_2, ..., a_n]$ where the a_i are positive integers and $a_n \geq 2$, such that

$$[a_1, a_2, ..., a_n] = \frac{x}{y} = \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cdots}}}.$$

The a_i are called the partial quotients, and are exactly the same as the quotients produced by the Euclidean algorithm applied to x and y. Therefore, $\begin{pmatrix} x \\ y \end{pmatrix}$ appears as a column of a matrix in $\mathcal{M}_k$ exactly when the partial quotients in the continued fraction expansion of $\frac{x}{y}$ are all multiples of k.

We note that this description of $\mathcal{M}_k$ can also be seen by writing A_k and B_k as Möbius transformations

$$A_k \mapsto a(z) = z + k, \qquad B_k \mapsto b(z) = \frac{z}{kz+1}.$$

Let $E_k = \{[ka_1, ka_2, ...]\}$ be the set of all real numbers whose partial quotients are multiples of k. This is a Cantor set whose Hausdorff dimension δ_k is important to us, so we collect results on the dimension here.

Lemma 2.2. *With E_k and δ_k as above,*

- $0.71936 \leq \delta_2 \leq 0.71950$
- $0.63956 \leq \delta_3 \leq 0.64073$
- $\delta_k > \frac{1}{2}$ *for* $k \geq 2$
- $\delta_k \to \frac{1}{2}$ *as* $k \to \infty$

Proof. The first two results come from [4, Table 1].

In the language of [4, §2], the set E_k is an infinite conformal iterated function system (IFS) when $k \geq 2$, and therefore its Hausdorff dimension δ_k is the

unique zero of the *topological pressure* function P. The bounded distortion property gives $P(t) \geq \log Z_1(t) - t \log K$ where K is the *distortion constant* (see [4, Proposition 2.5]) and $Z_1(t) = \sum_{m \geq 1} (km)^{-2t}$ so as $t \to \frac{1}{2}$ from above, $Z_1(t) \to \infty$ and therefore $P(t) \to \infty$. By [4, Proposition 2.2], $P(t) = \infty$ for $t \leq \frac{1}{2}$ and P is decreasing on $[0, \infty)$ and continuous on $(\frac{1}{2}, \infty)$ with $P(t) \to -\infty$ as $t \to \infty$. Therefore Bowen's parameter $h := \inf\{t \geq 0 : P(t) \leq 0\} > \frac{1}{2}$, and [4, Theorem 2.3] gives us $\delta_k = h > \frac{1}{2}$.

Good [7] proved that the Hausdorff dimension of the set of reals in $[0, 1]$ whose continued fraction partial quotients are all at least k tends to $\frac{1}{2}$ as $k \to \infty$. Our E_k is contained in this set, so $\limsup_k \{\delta_k\} \leq \frac{1}{2}$, finishing the proof. □

As in [21], we will use the Euclidean algorithm to factor matrices in $\mathcal{M}_k$.

Proposition 2.3. *If $x < y$ such that*

- $\frac{x}{y} = [ka_1, ..., ka_n] \in E_k$
- $y \equiv 1 \pmod{k^2}$
- $x \equiv 0 \pmod{k}$

then

$$B_k^{a_1} A_k^{a_2} B_k^{a_3} \cdots = \begin{pmatrix} * & x \\ * & y \end{pmatrix}.$$

Proof. The congruence conditions come from Proposition 2.1, and the matrix identity follows from the definitions of A_k and B_k. □

3 A collision attack

3.1 The reverse of a word

For $w \in \{0,1\}^*$, define $\text{rev}(w)$ to be the reverse of w. If $C = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ then $A = CA^TC$ and $B = CB^TC$. Therefore, $H_k(\text{rev}(w)) = CH_k(w)^TC$.

If we know a word w such that $H_k(w) = \begin{pmatrix} a & b \\ c & dp \end{pmatrix}$, then

$$\begin{aligned} H_k(w)H_k(\text{rev}(w)) &= \begin{pmatrix} a & b \\ c & dp \end{pmatrix} \begin{pmatrix} dp & b \\ c & a \end{pmatrix} \\ &= \begin{pmatrix} adp + bc & 2ab \\ 2cdp & adp + bc \end{pmatrix} \equiv \begin{pmatrix} -1 & 2ab \\ 0 & -1 \end{pmatrix} \mod p. \end{aligned}$$

This matrix commutes with A_k, which gives the following collision

$$H_k(w\text{rev}(w)0) = H_k(0w\text{rev}(w)).$$

Note that $w\text{rev}(w)0 \neq 0w\text{rev}(w)$ unless every letter of w is a 0.

3.2 Finding appropriate lifts

It remains to find a reliable algorithm for finding a word w such that $H_k(w) = \begin{pmatrix} a & b \\ c & dp \end{pmatrix} \in \mathcal{M}_k$. By Proposition 2.1, we must have $dp \equiv 1 \pmod{k^2}$, so we fix a small value of d such that $dp \equiv 1 \pmod{k^2}$. The size of this value of d will be discussed in Section 4.2.

Start with a prefix word w_1 so that $H_k(w_1) \in \mathcal{M}_k$ by construction, and denote by (c_1, d_1) the bottom row of $H_k(w_1)$. We pick w_1 at random such that $c_1, d_1 \sim \sqrt{p}$.

If w_1 has length n then $c_1, d_1 \sim e^{n\lambda_k}$ where λ_k is the growth rate of a random product of $\{A_k, B_k\}$. This λ_k is called the Lyapunov exponent, and is of independent interest [15].

Lemma 3.1. *If w_1 is such that $c_1, d_1 \sim \sqrt{p}$ then the expected length of w_1 is $O(\log p)$.*

Proof. The group $\langle A_k, B_k \rangle$ is Zariski dense in $SL_2(\mathbb{R})$ since its Zariski closure contains all upper and lower unitriangular matrices, and must therefore be $SL_2(\mathbb{R})$. The measure $\frac{1}{2}(\delta_{A_k} + \delta_{B_k})$ has finite support, so we may use Furstenberg's Law of Large Numbers as stated in [1, Theorem 0.6], which shows that λ_k exists [5] and $\lambda_k > 0$ [6, Theorem 8.6]. Therefore for a fixed k, we expect w_1 to have length $\sim \frac{\log p}{2\lambda_k} = O(\log p)$. □

Now we consider solutions $x, y \sim \sqrt{p}$ to the linear equation $c_1 x + d_1 y = dp$. If $\frac{x}{y} \in E_k$, then the Euclidean algorithm gives us a word w_2 such that $H_k(w_2)$ has last column $\begin{pmatrix} x \\ y \end{pmatrix}$, as described in Proposition 2.3. Now the matrix $H_k(w_1 w_2)$ has $(2,2)$-entry $c_1 x + d_1 y = dp$, which gives us a collision as described in §3.1. If $\frac{x}{y} \notin E_k$ then we try again with another random prefix w_1.

The remainder of this section discusses how many random prefixes w_1 we must try on average.

Let $S(w_1)$ be the number of times w_1 appears as a successful prefix

$$S(w_1) := \#\left\{(x,y) : \frac{x}{y} \in E_k, x, y \leq \sqrt{p}, c_1 x + d_1 y = dp\right\}.$$

We continue choosing random prefixes w_1 until we find one with $S(w_1) \geq 1$. Since this follows the geometric distribution, the expected number of prefixes we must check is $1/\mathbb{P}(S \geq 1)$.

We will make the following assumption.

Assumption 3.1. *Assume as w_1 ranges over words of length $O(\log p)$ that $\frac{x}{y}$ is in E_k as frequently as a random rational $\frac{a}{b}$ with $a, b \leq \sqrt{p}$.*

Zaremba's conjecture, which states that every positive integer denominator has at least one coprime numerator so that the partial quotients are all bounded by a small constant, has been proved for a set of denominators of density 1 [2, 10]

and recently announced in full generality by Zhang [24] building upon work of Shkredov [19]. A generalization called the Local-Global Conjecture [2] has been proven false in several cases [16, 18].

The assumption we are making is purely conjectural, and is in the spirit of the Zaremba Conjecture but for an infinite alphabet. In Section 4, we explain what to do when finding appropriate x, y is more difficult than expected, and provide experimental evidence that the algorithm works well.

Proposition 3.2. *Under Assumption 3.1, for any $\epsilon > 0$ we have $\mathbb{P}(S \geq 1) \gg p^{\delta_k - 1 - \epsilon}$.*

First, we write down the density of E_k in $[0, 1]$ using the notation of the survey [10]. Let

$$\mathscr{R}_{\mathcal{A}}(N) := \left\{ \frac{x}{y} = [a_1, a_2, ..., a_n] : (x, y) = 1,\ 1 \leq x < y < N,\ a_i \in \mathcal{A},\ \forall i \right\}.$$

Lemma 3.3. *For every $\epsilon > 0$,*

$$\#\mathscr{R}_{E_k}(N) \gg N^{2\delta_k - \epsilon}.$$

Proof. Hensley [9] (see also [10, (2.17)]) showed that, when $\mathcal{A}$ is a finite set and $\delta_{\mathcal{A}}$ is the Hausdorff dimension of $\{[a_1, a_2, ...] : a_i \in \mathcal{A}, \forall i\}$, we have $\#\mathscr{R}_{\mathcal{A}}(N) \asymp N^{2\delta_{\mathcal{A}}}$. By [4, Theorem 2.3], $\delta_k = \sup\{\delta_{\mathcal{A}} : \mathcal{A} \subset E_k \text{ finite}\}$ so we get $\#\mathscr{R}_{E_k}(N) \gg N^{2\delta_k - \epsilon}$. □

Proof of Proposition 3.2. There are around $\frac{d}{k} = O(1)$ positive integer points on the line $c_1 x + d_1 y = dp$, and under Assumption 3.1 we may invoke Lemma 3.3 to see that these each have probability $\gg \sqrt{p}^{(2\delta_k - 2\epsilon)} / \sqrt{p}^2$ of having $\frac{x}{y} \in E_k$. □

Therefore for any $\epsilon > 0$ the expected number of prefixes w_1 we must check is at most $1/\mathbb{P}(S \geq 1) = O(p^{1 - \delta_k + \epsilon})$, the average-case time complexity of our attack under Assumption 3.1.

4 Implementation and experimental results

We used Claude Opus 4.8 to implement the attack. Since our algorithm relies on Assumption 3.1, we build in a few safeguards to ensure our word length and run time do not become too large. We describe these safeguards first, then include the results of testing which shows the algorithm performs with the claimed time complexity. Finally, we give examples of collisions for large primes which have not been found before.

4.1 Capping the length

If Assumption 3.1 is true then the algorithm as described gives words of length $O(\log p)$ in time $O(p^{1 - \delta_k + o(1)})$, but if the assumption fails the algorithm may return longer words. To ensure the algorithm will only return a collision when

it is short enough, we add a step which rejects any w_2 with $|w_2| > 8 \log_2 p$. Note that this restricts us to a finite subset of E_k which has dimension strictly smaller than δ_k. This cap on the length of w_2 was chosen after computational experiments, and following sections show that the algorithm works well using this cap.

4.2 The size of d

Whether or not Assumption 3.1 is close to the truth depends on our choice of d, so we change d if no appropriate lift is found after the expected number of trials.

The first choice of d is $0 < d \leq k^2$, and d is increased by k^2 if no appropriate lift is found after $2^{\lfloor (1-\delta_k)n \rfloor}$ words w_1 have been tried, where n is the number of bits of the prime p.

4.3 The algorithm

1. Choose $0 < d \leq k^2$ such that $dp \equiv 1 \pmod{k^2}$.
2. Choose a random word w_1 so that if (c_1, d_1) is the bottom row of $H_k(w_1)$ then $c_1, d_1 \sim \sqrt{p}$. The word w_1 has length $O(\log p)$.
3. Solve the linear equation $c_1 x + d_1 y = dp$ in the integers.
4. Apply the Euclidean algorithm to determine whether $\frac{x}{y} \in E_k$.
5. If $\frac{x}{y} \notin E_k$, return to Step 2. Otherwise, construct w_2 such that $H_k(w_2) = \begin{pmatrix} * & x \\ * & y \end{pmatrix}$ using Proposition 2.3.
6. If $|w_2| > 8 \log_2 p$, return to Step 2. Otherwise, we have found $w := w_1 w_2$ such that $H_k(w \mathrm{rev}(w) 0) = H_k(0 w \mathrm{rev}(w))$, a collision.
7. If we try more than $2^{\lfloor (1-\delta_k)n \rfloor}$ choices of w_1 without success, increase the value of d by k^2 and return to Step 2.

Under Assumption 3.1, this algorithm returns a collision of length $O(\log p)$ in $O(p^{1-\delta_k + o(1)})$ time, on average. By Lemma 2.2, this gives average-case time complexity of $O(p^{0.28})$ when $k = 2$, and $O(p^{0.36})$ when $k = 3$. As k grows, the average-case time complexity tends to $O(\sqrt{p})$, and so the advantages of our attack over Slaughter's attack [20] vanish.

4.4 Experimental evidence for time complexity

We implemented the algorithm in Python and ran it on 60 random primes per bit-length for $k = 2, 3$. The tables below record the mean and median number of prefixes w_1 tried until the algorithm returned a collision.

4.4.1 $k = 2$

For $k = 2$, the least-squares exponent of the mean trial count is 0.297, close to the predicted $1 - \delta_2 \approx 0.28$.

$\log_2 p$	mean trials	median trials
20	17.9	13
25	57.1	33
30	171.3	144
35	334.3	262
40	793.6	576
45	3502.2	2429
50	8862.5	5028
55	27613.1	19240
60	68548.5	36156

4.4.2 $k = 3$

For $k = 3$, the least-squares exponent of the mean trial count is 0.365, close to the predicted $1 - \delta_3 \approx 0.36$.

$\log_2 p$	mean trials	median trials
20	56.8	32
25	221.9	94
30	651.6	493
35	2248.4	1200
40	6408.8	5302
45	29010.3	23246
50	136093.4	101010

4.5 Examples of collisions for large primes

To find collisions for large primes, we implemented the algorithm in C using the GNU Multiple Precision Arithmetic Library (GMP). This runs much more efficiently than our Python implementation.

Using a 2-vCPU cloud container (Intel Xeon, 2.3 GHz, 8 GB RAM), we found collisions for $k = 2$ with a 112-bit prime, and for $k = 3$ with a 92-bit prime. For $k = 2$ with a 152-bit prime, we found a collision using a Google Compute Engine with 88 vCPUs (Intel Xeon Platinum 8481C, 2.7 GHz). For $k = 3$ with a 118-bit prime, we found a collision using a Google Compute Engine with 44 vCPUs (Intel Xeon Platinum 8481C, 2.7 GHz).

All collisions were independently checked using a simple Python script. The colliding words are too long to print here but are included in the Python script available with the arXiv version of this paper.

As far as the author knows, these are the largest primes for which collisions for these hash functions have been found. The number of words w_1 tried in each

case is listed in the following table as well as the length of the collision (i.e. the sum of the lengths of the colliding words, at most $2 + 32 \log_2 p$).

k	p	length	trials	$\approx p^{1-\delta_k}$	vCPUs
2	$2^{112} - 75$	878	2.9×10^9	2.7×10^9	2
2	$2^{152} - 17$	3626	2.3×10^{10}	6.5×10^{12}	88
3	$2^{92} - 83$	642	1.7×10^8	4.9×10^9	2
3	$2^{118} - 5$	2494	6.1×10^{10}	2.7×10^{12}	44

Code availability

The code for this project is available in an ancillary file in the arXiv version of this paper. It contains the Python implementation of the algorithm `bsv_search.py`, the faster version written in C `search.c`, and the Python code verifying the reported large-prime collisions `large_p_verification.py`.

Acknowledgements

The author thanks Chris Lutsko for interesting discussions, and in particular pointing out that the quotients of columns of positive products of A_2 and B_2 are those whose continued fraction expansions have even partial quotients. The author thanks Vladimir Shpilrain for comments on an early draft of this paper.

AI use acknowledgement

Aristotle by Harmonic was used in exploratory conversations and suggested the main approach of this paper, along with others that did not prove fruitful. The author found a significant gap in the initial suggestion and prompted Aristotle to patch the gap using Assumption 3.1. Claude Opus 4.8 wrote the code which produced the experimental results, and was used to teach the author about Zaremba's conjecture, proofread and check the article before submission for peer review. No LLM-generated writing appears in this paper. The author takes responsibility for the entire contents of this paper.

References

[1] Yves Benoist and Jean-François Quint. *Random Walks on Reductive Groups*. Springer International Publishing, 2016.

[2] Jean Bourgain and Alex Kontorovich. On Zaremba's conjecture. *Annals of Mathematics*, 180(1):137–196, 2014.

[3] 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, December 2015.

[4] Vasileios Chousionis, Dmitriy Leykekhman, and Mariusz Urbański. On the dimension spectrum of infinite subsystems of continued fractions. *Transactions of the American Mathematical Society*, 373(2):1009–1042, 2020.

[5] H. Furstenberg and H. Kesten. Products of random matrices. *The Annals of Mathematical Statistics*, 31(2):457–469, 1960.

[6] Harry Furstenberg. Noncommuting random products. *Transactions of the American Mathematical Society*, 108(3):377–428, 1963.

[7] I. J. Good. The fractional dimensional theory of continued fractions. *Mathematical Proceedings of the Cambridge Philosophical Society*, 37(3):199–228, 1941.

[8] Markus Grassl, Ivana Ilić, Spyros Magliveras, and Rainer Steinwandt. Cryptanalysis of the Tillich–Zémor hash function. *Journal of Cryptology*, 24(1):148–156, March 2010.

[9] Doug Hensley. The distribution of badly approximable numbers and continuants with bounded digits. *Théorie des nombres (Quebec, PQ, 1987)*, pages 371–385, 1989.

[10] Alex Kontorovich. From Apollonius to Zaremba: local-global phenomena in thin orbits. *Bulletin of the American Mathematical Society*, 50(2):187–228, 2013.

[11] Corentin Le Coz, Christopher Battarbee, Ramón Flores, Thomas Koberda, and Delaram Kahrobaei. Post-quantum hash functions using $\mathrm{SL}_n(\mathbb{F}_p)$. *Advances in Mathematics of Communications*, 19(3):996–1009, 2025.

[12] Eilidh McKemmie and Amol Srivastava. Preimages for Zémor's Cayley hash function. 2025.

[13] Ciaran Mullan and Boaz Tsaban. SL_2 homomorphic hash functions: worst case to average case reduction and short collision search. *Designs, Codes and Cryptography*, 81(1):83–107, August 2015.

[14] Christophe Petit and Jean-Jacques Quisquater. Preimages for the Tillich-Zémor hash function. In *Selected Areas in Cryptography*, pages 282–301. Springer Berlin Heidelberg, 2011.

[15] Mark Pollicott. Maximal Lyapunov exponents for random matrix products. *Inventiones mathematicae*, 181(1):209–226, April 2010.

[16] James Rickards and Katherine E. Stange. Reciprocity obstructions in semigroup orbits in sl(2,z). *Duke Mathematical Journal*, 174(15):3197–3244, October 2025.

[17] Ivan N Sanov. A property of a representation of a free group. In *Doklady Akad. Nauk SSSR (NS)*, volume 57, page 16, 1947.

[18] Hanqi Shi, Wenyuan Shi, Ian Whitehead, Ham Williams-Tracy, and Jeffrey Zhirui Zhang. The local-global conjecture is false for generalized circle packings, 2025.

[19] Ilya D. Shkredov. On some results of Korobov and Larcher and Zaremba's conjecture, 2026.

[20] Freeman Slaughter. CHAMPing at the bit: Short square-root collisions for two-generator cayley hashes. Cryptology ePrint Archive, Paper 2026/2713, 2026.

[21] Jean-Pierre Tillich and Gilles Zémor. Group-theoretic hash functions. In G. Cohen, S. Litsyn, A. Lobstein, and G. Zémor, editors, *Algebraic Coding*, pages 90–110, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[22] Jean-Pierre Tillich and Gilles Zémor. Hashing with SL_2. In Yvo G. Desmedt, editor, *Advances in Cryptology — CRYPTO '94*, pages 40–49. Springer Berlin Heidelberg, 1994.

[23] G. Zémor. Hash functions and graphs with large girths. In *Advances in Cryptology — EUROCRYPT '91*, pages 508–511. Springer Berlin Heidelberg, 1991.

[24] Xin Zhang. Expansion in $\mathrm{SL}_2(\digamma/q\digamma)$ and Zaremba's conjecture, 2026.