საქართველოს მეცნიერებათა ეროვნული აკადემიის მოამბე, ტ. 20(194), №3, 2026
BULLETIN OF THE GEORGIAN NATIONAL ACADEMY OF SCIENCES, vol. 20(194), no. 3, 2026

Informatics

A Cayley Hash Function with Double Cookies

Nick Inassaridze[*], Andria Inassaridze[**]

[*] *A. Razmadze Mathematical Institute, Ivane Javakhishvili Tbilisi State University; Tbilisi Centre for Mathematical Sciences, Georgia*
[**] *Tbilisi Centre for Mathematical Sciences, Georgia*

(Presented by Academy Member Hvedri Inassaridze)

Abstract. A variant of Cayley hash functions based on matrix semigroups is proposed in which a bit string is parsed by grouping consecutive identical bits into pairs, while remaining single bits are treated as "cookies". This parsing rule naturally leads to a hashing scheme with four generators in $SL_2(\mathbb{F}_p)$. We prove that the corresponding semigroup is free over $\mathbb{Z}$, which implies injectivity prior to reduction modulo p. Using growth estimates derived from known results on products in $SL_2(\mathbb{Z})$, logarithmic lower bounds are obtained on the girth of the associated Cayley graph and an explicit collision-free range is derived. Additionally, structural aspects of preimage resistance are analyzed, demonstrating that the proposed construction eliminates the deterministic greedy inversion strategy known for classical two-generator schemes. © *2026 Bull. Georg. Natl. Acad. Sci.*

Keywords: Cayley hash function, matrix semigroup, girth, preimage resistance, cryptography

Introduction

Motivation. Cayley type hash functions form a family of algebraic hash constructions in which an input string is interpreted as a word in generators of a semigroup and the hash value is obtained as the corresponding matrix product over a finite field. This paradigm was introduced by Zémor (1991), and further developed by Tillich and Zémor (1994).

In such schemes, collision resistance is closely related to lower bounds on the girth of the associated Cayley graph, while preimage resistance depends on the difficulty of decomposing a matrix into the given generators. When generators are chosen in $SL_2(\mathbb{Z})$ and reduced modulo p, distinct positive words remain distinct until matrix entries grow large enough for reduction modulo p to identify them.

Classical constructions based on two generators, such as $A(2) = X^2$ and $B(2) = Y^2$, exhibit favorable growth properties and yield logarithmic girth bounds. However, these schemes admit a deterministic greedy word recovery algorithm within the injective range, which motivates modifications of the parsing rule.

© 2026 Bull. Georg. Natl. Acad. Sci.

Shpilrain and Sosnovski (2025) introduced a 3-generator variant using so-called ``cookies'', showing that increasing the structural complexity of the parsing process can obstruct such inversion strategies while preserving efficient computation.

Contribution. In this paper, we introduce a 4-generator Cayley hash construction obtained by grouping identical consecutive bits into pairs and treating remaining single bits as cookies. The resulting generators are

$$A = X^2, B = Y^2, C = XY, D = YX.$$

Our contributions are as follows:

- We prove that the semigroup generated by A, B, C, D is free over $\mathbb{Z}$.
- Using known growth estimates for products in $SL_2(\mathbb{Z})$, we determine the maximal exponential growth rate of matrix entries in terms of the input length and derive explicit logarithmic lower bounds on collision length modulo p.
- We show that alternating bit strings produce extremal growth and obtain a concrete collision free interval.
- We analyze structural aspects of preimage resistance and demonstrate that, unlike the classical two generator case, the deterministic greedy inversion strategy no longer applies, leading to branching behavior in word recovery.

The proposed construction preserves the favorable collision properties of classical Cayley hashing while strengthening resistance against known structural preimage attacks.

1. Background

We briefly recall the Cayley hashing framework introduced by Zémor (1991) and further developed in (Tillich and Zemor, 1994). In these constructions, an input string is interpreted as a word in generators of a semigroup or group, and the hash value is computed as the corresponding product of matrices over a finite field.

Generators are typically chosen in matrix groups such as $SL_2(\mathbb{Z})$ or $SL_2(\mathbb{F}_p)$, where matrix multiplication is efficient and algebraic properties of the generators allow structural analysis of collisions. In particular, the absence of short relations among generators corresponds to a large girth of the associated Cayley graph, which implies that short collisions cannot occur.

A common approach is to select generators in $SL_2(\mathbb{Z})$ whose positive words form a free semigroup. Distinct words then represent distinct integer matrices, and collisions can arise only after reduction modulo p ceases to be injective on the set of matrices produced.

Another important aspect of Cayley hash functions is the growth of entries in products of generators. For the classical matrices

$$X = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, Y = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix},$$

alternating products of X and Y exhibit exponential growth of entries. This phenomenon provides logarithmic lower bounds on the minimal collision length modulo p (Bromberg et al., 2017).

More recent work has explored modifications of the parsing rule used to interpret bit strings. In particular, Shpilrain and Sosnovski (2025) proposed a cookie-based construction in which generator selection depends on local bit patterns rather than individual bits. Such modifications increase structural complexity while preserving computational efficiency.

In the present paper, we follow this general paradigm but introduce a different parsing rule that groups identical consecutive bits into pairs and treats remaining bits as cookies. This naturally leads to a four-generator construction while retaining the algebraic structure needed for growth estimates and security analysis.

2. Cayley Hashing with Double Cookies

Let A, B, C, and D be 2×2 matrices. Let u be a bit string of arbitrary length. To hash u, process the bits from left to right as follows:

(1) Every pair of consecutive 0 bits is hashed to the matrix A, while every pair of consecutive 1 bits is hashed to the matrix B.

(2) After Step (1), some single bits may remain unhashed. Each remaining 0 bit (a "cookie") is hashed to the matrix C, and each remaining 1 bit (another "cookie") is hashed to the matrix D. For example, the bit string $10011110001111101000 01$ is hashed to the matrix sequence $DABBACBBDCDAAD$.

The recommended matrices are:

$$A = \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}, B = \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}, C = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}, D = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}.$$

Proposition 2.1. The semigroup generated by the matrices A, B, C, and D over $\mathbb{Z}$ is free.

Proof. Define

$$X = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, Y = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}.$$

It is well known that the matrices X and Y generate a free semigroup over $\mathbb{Z}$; that is, distinct positive words in the alphabet $\{X, Y\}$ represent distinct matrices.

Observe that

$$\mathrm{A} = \mathrm{X}^2, \mathrm{B} = \mathrm{Y}^2, \mathrm{C} = \mathrm{XY}, \mathrm{D} = \mathrm{YX}.$$

Consider the four words in the alphabet $\{X, Y\}$:

$$X^2, Y^2, XY, YX.$$

Each of these words has length 2, and therefore none is a prefix of another. It follows that any concatenation of these words admits a unique decomposition into blocks of length 2. Hence, any word in the alphabet $\{A, B, C, D\}$ corresponds uniquely to a word in $\{X, Y\}$. Since distinct words in $\{X, Y\}$ represent distinct matrices, distinct words in $\{A, B, C, D\}$ also represent distinct matrices. Therefore, the semigroup generated by A, B, C, and D is free. □

Now consider the matrices A, B, C, and D over the field $\mathbb{F}_p$. Since the semigroup generated by these matrices over $\mathbb{Z}$ is free, distinct positive words in the alphabet $\{A, B, C, D\}$ represent distinct integer matrices. Therefore, when reduced modulo p, two different words can coincide only after reduction modulo p ceases to be injective on the set of matrices that arise. In particular, collisions cannot occur until at least one entry of a product of matrices becomes larger than p.

Of course, the hash function cannot be collision-free for arbitrarily long inputs, since the set of 2×2 matrices over $\mathbb{F}_p$ is finite, while the number of possible input words is infinite. Thus collisions must eventually occur, but the above argument shows that they cannot appear before the entries of the corresponding integer matrices reach size comparable to p.

One may also observe that relations in the group generated by $A, B, C, D \in SL_2(\mathbb{Z})$ involving inverses may lead, after reduction modulo p, to relations between positive powers. For instance,

$$CB^{-1}D = A$$

holds in $SL_2(\mathbb{Z})$. If t denotes the order of B in $SL_2(\mathbb{F}_p)$, then

$$B^t = I \pmod p,$$

so that $B^{-1} = B^{t-1} \pmod p$, and the above identity yields a relation

$$CB^{t-1}D = A \pmod p.$$

Since

$$B^k = \begin{pmatrix} 1 & 0 \\ 2k & 1 \end{pmatrix},$$

we have $B^k = I \pmod p$ if and only if $2k = 0 \pmod p$. For odd primes p, this implies that the order of B in $SL_2(\mathbb{F}_p)$ is exactly p.

Therefore, before reaching exponent $t - 1 = p - 1$, the integer entry $2k$ in the matrix B^k already reaches magnitude on the order of p. Consequently, relations of this type can only arise after matrix entries have grown to size comparable to p, which is precisely the regime where reduction modulo p may produce collisions.

2.1. Padding. To preserve the homomorphic property

$$H(uv) = H(u)\,H(v),$$

a minor padding of each input string is required.

Since the parsing rule groups consecutive identical bits into pairs, direct concatenation of two bit strings may alter the parsing at the boundary. To avoid this, we pad each bit string u by defining

$$\tilde{u} = 0\,u\,1.$$

Then the boundary between two padded strings $\tilde{u}$ and $\tilde{v}$ is always the substring 10, which does not affect the parsing into blocks. Consequently,

$$H(\widetilde{u\tilde{v}}) = H(\tilde{u})\,H(\tilde{v}),$$

and the homomorphic property is preserved.

3. Efficiency of the Construction

We estimate the computational cost of evaluating the hash function described in previous section.

Let u be a bit string of length n. After applying the parsing rule, the string is converted into a sequence of matrices from the set $\{A, B, C, D\}$. The number of matrices in this sequence is at most n, and for typical inputs is close to $n/2$. Computing the hash value $H(u)$ requires $k - 1$ matrix multiplications, where k is the length of this sequence.

3.1. Cost of multiplication. The generators

$$A = \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}, B = \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}, C = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}, D = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}$$

have entries $0, 1$ and 2 only. When multiplying a matrix

$$M = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

by any of these generators, each resulting entry is obtained using only additions in $\mathbb{F}_p$, since multiplication by 2 can be implemented as an addition. A direct calculation shows that each multiplication can be performed using at most four additions by reusing intermediate sums such as $a + b$ and $c + d$.

3.2. Overall complexity. Therefore computing $H(u)$ requires no general multiplications and at most $4(k-1)$ additions in $\mathbb{F}_p$. Since $k \leq n$, the complexity is linear in the input length, and in practice, because many bits are grouped into pairs, k is typically close to $n/2$.

4. Growth Estimates

In Cayley type hashing schemes, as explained in (Shpilrain et al., 2025), lower bounds on the minimal collision length modulo p can be obtained from upper bounds on the growth of entries in products of generators over $\mathbb{Z}$: as long as reduction modulo p is injective on the set of reachable integer matrices, collisions cannot occur.

We follow the same approach as in (Shpilrain and Sosnovski, 2025, Section 5) and, in particular, we use the same type of growth estimates that are derived there from results of (Bromberg et al., 2017).

A key difference in our setting is that an input bit string of length n does not correspond to a word of length m in the generator alphabet $\{A, B, C, D\}$. Indeed, the parsing rule groups many bits into pairs, and only leftover single bits become cookies. Thus the length of the resulting generator word is typically smaller than n.

4.1. From bit strings to generator words. Given a bit string $u \in \{0,1\}^n$, let $H(u)$ be the word over $\{A, B, C, D\}$ produced by the parsing rule in *Section 2*, and write $m(u)$ for its length. Then $H(u) \in SL_2(\mathbb{F}_p)$ and clearly $m(u) \leq n$, with $m(u) \approx n/2$ for typical inputs.

For a matrix M, write $\|M\| = max_{i,j}|M_{ij}|$ for the maximal absolute entry.

4.2. Growth problems in terms of the input length. In analogy with (Shpilrain and Sosnovski, 2025, Section 5), we consider the following two growth questions, stated in terms of the bit string length n.

Problem 4.1. (Worst-case growth for input length n). For $n \geq 1$, determine the asymptotic behavior of

$$F(n) = \max_{u \in \{0,1\}^n} \|H(u)\|.$$

Equivalently, among all bit strings of length n, how large can the entries of the corresponding integer matrix product become?

Problem 4.2. (Generic growth for input length n). Let U_n be a generic (e.g. uniformly random) bit string of length n and set $W_n = H(U_n)$ as a word in $\{A, B, C, D)\}$. Estimate the typical size of $\|W_n\|$ as a function of n, for example by studying the average growth rate $\mathbb{E}[\log\|W_n\|]$ or by obtaining bounds that hold with high probability for random inputs.

As in Shpilrain and Sosnovski, (2025), Problem 4.1 is relevant to worst-case (adversarial) inputs and provides conservative lower bounds on the collision length, whereas Problem 4.2 models the typical behavior for random inputs. In both cases, one first studies growth over $\mathbb{Z}$ and then transfers the conclusion to the modular setting via the injectivity observation: if $\|H(u)\| < p$ for every u in a given class of inputs, then no collisions can occur within that class.

The following proposition relates the growth of the proposed four-generator construction to known growth estimates for products in the classical generators X, Y used by Bromberg et al. (2017) (and summarized by Shpilrain and Sosnovski (2025, Section 5).

Proposition 4.3. Among all bit strings of length n, the alternating string $u_n = 0101$ ... yields hash matrices whose entries grow at the maximal exponential rate.

Proof. Let $u_n = 0101$ ... be the alternating bit string of length n. Since no two consecutive bits are equal, Step (1) of the parsing rule produces no pairs, and every bit becomes a cookie. Hence $H(u_n)$ is an alternating word in C and D of length n:

$$H(u_n) \in \{(CD)^{\lfloor n/2 \rfloor},\ (DC)^{\lfloor n/2 \rfloor}\}$$

up to a possible final factor when n is odd.

Recall that $C = XY$ and $D = YX$. Thus $H(u_n)$ expands to an alternating product in X and Y, which, up to fixed left and right factors, reduces to an alternating word in $A(2) = X^2$ and $B(2) = Y^2$ of length approximately $n/2$.

By [4], among all words $w\big(A(2), B(2)\big)$ of length m, the maximal growth of entries is achieved by alternating words $\big(A(2)B(2)\big)^{m/2}$.

Any other bit string of length n either produces fewer generators (due to pairs 00 or 11) or yields less alternation in the induced $A(2), B(2)$ word. Hence no other input of length n can exceed the growth rate obtained for the alternating string. □

4.3. A collision-free range. We now derive a lower bound on the minimal collision length modulo p, following the approach of (Shpilrain and Sosnovski, 2025, Section 5) and the growth estimates for $A(2), B(2)$ obtained in (Bromberg et al., 2017).

Since the semigroup generated by A, B, C, D is free over $\mathbb{Z}$, distinct positive words represent distinct integer matrices. Thus, two different inputs can collide modulo p only after reduction modulo p ceases to be injective. In particular, collisions cannot occur until some matrix entry reaches magnitude at least p.

By Proposition 4.3, the maximal growth for inputs, of length n is achieved by the alternating string 0101 ... For such inputs, the matrix entries grow like

$$\big(1 + \sqrt{2}\big)^n \quad \text{(Bromberg et al., 2017).}$$

Therefore reduction modulo p remains injective as long as

$$\big(1 + \sqrt{2}\big)^n < p,$$

which implies

$$n < \log_{1+\sqrt{2}} p.$$

If $p \approx 2^{256}$, then $\log_{1+\sqrt{2}} p \approx 203$. Thus no collisions can occur for inputs of length strictly less than about 203 bits.

5. Collision and Preimage Resistance

In this section we discuss heuristic evidence for collision resistance and preimage resistance of the proposed construction. Our reasoning follows the general framework used in (Zémor, 1991; Tillich and Zemor, 1994; Shpilrain and Sosnovski, 2025; Bromberg et al., 2017).

5.1. Collision resistance and girth. Collision resistance of Cayley type hash functions is closely related to lower bounds on the girth of the corresponding Cayley graph over $\mathbb{F}_p$. The girth is the length of the shortest nontrivial relation among the generators modulo p.

In our construction, the matrices A, B, C, D generate a free semigroup over $\mathbb{Z}$ (Proposition 2.1). Hence distinct positive words represent distinct integer matrices. Therefore two different inputs can collide modulo p only after reduction modulo p ceases to be injective on the set of integer matrices produced by words of the relevant length.

As shown in *Section 4*, the maximal growth of entries for inputs of length n is achieved by the alternating string 0101 ..., and the corresponding matrix entries grow on the order of

$$\left(1+\sqrt{2}\right)^n.$$

Thus reduction modulo p remains injective as long as

$$\left(1+\sqrt{2}\right)^n < p.$$

This yields a logarithmic lower bound on the girth:

$$\text{girth} \gtrsim \log_{1+\sqrt{2}} p.$$

Logarithmic lower bounds of this type are typical for Cayley hash constructions over $\mathbb{F}_p$ (Zémor, 1991; Tillich and Zemor, 1994; Bromberg et al., 2017). As in earlier proposals, the base of the logarithm is important, since it determines the concrete collision free interval.

For example, if $p \approx 2^{256}$, then no collisions can occur for input lengths strictly less than approximately 203 bits.

5.2. Preimage resistance. Preimage resistance concerns the difficulty of recovering a word in the generators from a given matrix in $SL_2(\mathbb{F}_p)$.

For the classical two-generator scheme based on $A(2) = X^2$ and $B(2) = Y^2$, there exists an efficient greedy recovery algorithm (see [4]): given $W = w\big(A(2), B(2)\big)$, exactly one of the matrices

$$WA(2)^{-1} \quad \text{or} \quad WB(2)^{-1}$$

strictly decreases a natural size function, for instance the sum of absolute values of entries. This makes deterministic reconstruction possible in time linear in the word length, as long as one remains in the injective (collision free) range.

The situation is different in the present construction. Since we use four generators

$$A = X^2, B = Y^2, C = XY, D = YX,$$

the matrices

$$WA^{-1}, \quad WB^{-1}, \quad WC^{-1}, \quad WD^{-1}$$

may simultaneously decrease natural size measures of W. There is no longer a unique inverse step that strictly reduces matrix size.

Consequently, a reconstruction algorithm must in general explore multiple candidate predecessors at some stages, leading to branching behavior. In the worst case, the number of possibilities grows exponentially with the length of the word. This eliminates the deterministic greedy strategy that is effective for the 2-generator case.

In this sense, the present four-generator construction removes the structural weakness that allows efficient preimage recovery for the pair $(A(2), B(2))$.

5.3. Brute force considerations. A brute force preimage attack on inputs of length n requires testing 2^n possible bit strings. Therefore, to achieve a security level of t bits, the input length should satisfy $n \geq t$. Current cryptographic recommendations typically require $t \geq 256$.

5.4. Remarks. As in other Cayley hash constructions, the above arguments provide heuristic evidence rather than formal cryptographic proofs. Security ultimately depends on the absence of efficient algorithms for solving the word decomposition problem in the corresponding semigroup modulo p.

Nevertheless, the combination of

- freeness over $\mathbb{Z}$,

- exponential growth of entries,
- logarithmic girth bounds, and
- absence of a deterministic greedy inversion strategy

suggests that the proposed construction inherits the favorable collision properties of classical Cayley hashing while improving resistance to known preimage attacks.

Acknowledgements

This work was supported by the Shota Rustaveli National Science Foundation of Georgia, grant FR-24-822.

ინფორმატიკა

კეილის ჰეშფუნქცია ორმაგი კუკიებით

ნიკოლოზ ინასარიძე*, ანდრია ინასარიძე**

** ივანე ჯავახიშვილის სახ. თბილისის სახელმწიფო უნივერსიტეტი, ა. რაზმაძის მათემატიკის ინსტიტუტი; თბილისის მათემატიკის მეცნიერების ცენტრი, საქართველო*
*** თბილისის მათემატიკის მეცნიერების ცენტრი, საქართველო*

(წარმოდგენილია აკადემიის წევრის ხვედრი ინასარიძის მიერ)

ნაშრომში წარმოდგენილია კეილის ტიპის ჰეშფუნქციების ვარიანტი, რომელიც დაფუძნებულია მატრიცულ ნახევარჯგუფებზე. ამ კონსტრუქციაში ბიტური სტრიქონი იშლება ისე, რომ თანმიმდევრული ერთნაირი ბიტები ერთიანდება წყვილებად, ხოლო დარჩენილი ერთეული ბიტები განიხილება როგორც „კუკიები“. აღნიშნული დაშლის წესს ბუნებრივად მივყავართ ოთხ წარმომქმნელიან ჰეშირების სქემამდე $SL_2(\mathbb{F}_p)$-ში. ვამტკიცებთ, რომ შესაბამისი ნახევარჯგუფი თავისუფალია $\mathbb{Z}$-ზე, რაც უზრუნველყოფს ინექციურობას p მოდულამდე. $SL_2(\mathbb{Z})$-ში ნამრავლების შესახებ ცნობილი შედეგებიდან მიღებული ზრდის შეფასებების გამოყენებით ვიღებთ ასოცირებული კეილის გრაფის სიმრუდის ლოგარითმულ ქვედა ზღვარს და კოლიზიისგან თავისუფალ ცხად დიაპაზონს. დამატებით ვაანალიზებთ წინასახის მედეგობის სტრუქტურულ ასპექტებს და ნაჩვენებია, რომ შემოთავაზებული კონსტრუქცია გამორიცხავს დეტერმინისტულ შებრუნებადობის სტრატეგიას, რომელიც ცნობილია კლასიკური 2-წარმომქმნელიანი სქემებისთვის.

REFERENCES

Bromberg, L., Shpilrain, V., Vdovina, A. (2017). Navigating in the Cayley graph of $SL_2(\mathbb{F}_p)$ and applications to hashing, *Semigroup Forum*, 314-324.

Shpilrain, V., Sosnovski, B. (2025). Cayley hashing with cookies, *Future of Information and Communication Conference (FICC 2025)*, 706-718.

Tillich, J.-P., Zémor, G. (1994). Hashing with {SL}_2. *Advances in Cryptology—CRYPTO '94*, Lecture Notes in Computer Science, *839,* 40-49.

Zémor, G. (1991). Hash functions and graphs with large girths. *Advances in Cryptology – EUROCRYPT '91*, Lecture Notes in Computer Science, *547,* 508-511.

Received March, 2026