

Navigating in the Cayley graph of $SL_2(\mathbb{F}_p)$ and applications to hashing

Vladimir Shpilrain (joint work with Lisa Bromberg and Alina Vdovina)
The City College of New York

June 2015

Definition

Let $n \in \mathbb{N}$ and let $H: \{0, 1\}^* \rightarrow \{0, 1\}^n$ be a function that takes a bit string of an arbitrary length to a bit string of a fixed length n . We require a hash function H to satisfy the following conditions:

- 1 *preimage resistance*: given output y , it is hard to find input x such that $H(x) = y$;
- 2 *second preimage resistance*: given input x_1 , it is hard to find another input $x_2 \neq x_1$ such that $H(x_1) = H(x_2)$;
- 3 *collision resistance*: it is hard to find inputs $x_1 \neq x_2$ such that $H(x_1) = H(x_2)$.

Early suggestions (especially the SHA family) did not really use any mathematical ideas to ensure the above properties were met; the main idea was just to 'create a mess' by using complex iterations.

An interesting direction is constructing hash functions that are provably as secure as the underlying assumptions, e.g. the discrete logarithm assumptions. These hash functions tend to not be efficient.

Cayley hash functions

We take a different approach, namely use two elements, A and B , of a semigroup S , such that the Cayley graph of the semigroup generated by A and B belongs to an *expander family*, in the hope that such a graph will have large girth and therefore there will be no short relations ('collisions').

To build a hash function from the Cayley graph, a message m (a bitstring comprised of 0's and 1's) corresponds to a word in the elements A and B of S , with A corresponding to 0 and B corresponding to 1. This is represented on the Cayley graph as a (nonbacktracking) walk; the endpoint of the walk is the hash value.

Tillich–Zémor hash function

The most popular implementation of hashing with the Cayley graph is probably the one due to Tillich and Zémor [5]. Unlike functions in the SHA family, the Tillich–Zémor hash function is not a block hash function, but rather each bit is hashed individually.

More specifically, the “0” bit and the “1” bit are hashed to matrices A and B (over a finite ring), respectively. Then, an arbitrary bit string is hashed by using multiplication of matrices. For example, the bit string 1001101 is hashed to the matrix BA^2B^2AB .

Tillich–Zémor hash function (continued)

Tillich and Zémor use matrices A, B from the group $SL_2(R)$, where R is a commutative ring defined by $R = \mathbb{F}_2[x]/(p(x))$. They took $p(x)$ to be the irreducible polynomial $x^{131} + x^7 + x^6 + x^5 + x^4 + x + 1$ over $\mathbb{F}_2[x]$. Thus, R is isomorphic to \mathbb{F}_{2^n} , where n is the degree of the irreducible polynomial $p(x)$. Then, the matrices A and B are

$$A = \begin{pmatrix} \alpha & 1 \\ 1 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} \alpha & \alpha + 1 \\ 1 & 1 \end{pmatrix},$$

where α is a root of $p(x)$.

This particular hash function was successfully attacked by Grassl, Ilić, Magliveras and Steinwandt in 2011 [1], and Petit and Quisquater in 2010 [2].

Another idea is to use a pair of 2×2 matrices A and B which generate a free monoid over \mathbb{Z} , and then reduce the entries modulo a large prime p to get matrices over \mathbb{F}_p .

Since there cannot be equality of two different products of positive powers of A and B unless at least one of the entries in at least one of the products is greater than or equal to p , this gives a lower bound on the minimum length of bit strings where a collision may occur. This bound is on the order of $\log p$; we will give more precise bounds in two particular examples.

An example of a pair of matrices over \mathbb{Z} which generate a free monoid is

$$A(1) = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad B(1) = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}.$$

These matrices are invertible and thus actually generate the *group* $SL_2(\mathbb{Z})$. This group is not free, but the *monoid* generated by $A(1)$ and $B(1)$ is free. Since only positive powers are used in hashing, this is all we need.

However, a collision for the hash function corresponding to these matrices over a large prime p was described by Tillich and Zémor [4] by using what they called a “lifting attack”.

Note that a pair of matrices

$$A(x) = \begin{pmatrix} 1 & x \\ 0 & 1 \end{pmatrix}, \quad B(y) = \begin{pmatrix} 1 & 0 \\ y & 1 \end{pmatrix},$$

generate a free subgroup of $SL_2(\mathbb{Z})$ if $xy \geq 4$ (Sanov [3] was the first to prove this in the case $x, y = 2$). We will consider the cases where $x = y = 2$ and $x = y = 3$.

The “lifting attack” on the hash function based on $A(1)$ and $B(1)$ described by Tillich and Zémor is the only published attack on that hash function. That particular attack does not work with $A(2)$ and $B(2)$ or with $A(3)$ and $B(3)$.

Recall that when considered as matrices over \mathbb{Z} , $A(1)$ and $B(1)$ generate (as a monoid) the entire monoid of 2×2 matrices over \mathbb{Z} with positive entries, $SL_2(\mathbb{Z}_+)$.

However, this is not the case with $A(2)$ and $B(2)$. Another result of Sanov says that the subgroup of $SL_2(\mathbb{Z})$ generated by $A(2)$ and $B(2)$ consists of all invertible matrices of the form

$$\begin{pmatrix} 1 + 4m_1 & 2m_2 \\ 2m_3 & 1 + 4m_4 \end{pmatrix},$$

where the m_i are integers. This does not say much about the *monoid* generated by these matrices, though. In fact, a generic matrix of the form above would not belong to this monoid.

Note that the number of matrices in the above form which are representable by positive words is negligible. In fact, the number of distinct elements represented by all freely reducible words in $A(2)$ and $B(2)$ of length $n \geq 2$ is $4 \cdot 3^{n-1}$, while the number of distinct elements represented by positive words of length $n \geq 2$ is 2^n .

Problem

*Find a form similar to Sanov's for matrices in the **monoid** generated by $A(2)$ and $B(2)$ over \mathbb{Z} .*

Tillich and Zémor's lifting attack can still give an efficient algorithm which finds relations of length $O(\log p)$ in the group generated by $A(2)$ and $B(2)$ considered as matrices over \mathbb{F}_p .

Theorem

There is an efficient heuristic algorithm that finds particular relations of the form $w(A(2), B(2)) = 1$, where w is a group word of length $O(\log p)$, and the matrices $A(2)$ and $B(2)$ are considered over \mathbb{F}_p .

This does not affect the security of the hash function based on $A(2)$ and $B(2)$ since only positive powers of $A(2)$ and $B(2)$ are used, and the group relations produced by the algorithm will involve both negative and positive powers with overwhelming probability.

Girth of the Cayley graph generated by $A(k)$ and $B(k)$

For hashing, we use only positive powers, so we need only consider products of positive powers of $A(k)$ and $B(k)$. We note that entries in a matrix of a length n product of positive powers of $A(k)$ and $B(k)$ grow fastest (as functions of n) in the alternating product of $A(k)$ and $B(k)$. This is summarized in the following proposition.

Proposition

Let $w_n(a, b)$ be an arbitrary positive word of even length n , and let $W_n = w_n(A(k), B(k))$, with $k \geq 2$. Let $C_n = (A(k) \cdot B(k))^{n/2}$. Then:
(a) the sum of entries in any row of C_n is at least as large as the sum of entries in any row of W_n ; **(b)** the largest entry of C_n is at least as large as the largest entry of W_n .

Thus we consider powers of the matrix

$$C(k) = A(k)B(k) \tag{1}$$

to get to entries larger than p “as quickly as possible”.

The matrix $C(2)$ is

$$\begin{pmatrix} 5 & 2 \\ 2 & 1 \end{pmatrix}.$$

If we denote

$$(C(2))^n = \begin{pmatrix} a_n & b_n \\ c_n & d_n \end{pmatrix},$$

then the following recurrence relations are easily proved by induction on n :

$$a_n = 5a_{n-1} + 2b_{n-1},$$

$$b_n = c_n = 2a_{n-1} + b_{n-1},$$

$$d_n = a_{n-1}.$$

Combining the recurrence relations for a_n and b_n , we get $2b_n = a_n - a_{n-1}$, so $2b_{n-1} = a_{n-1} - a_{n-2}$. Plugging this into the displayed recurrence relation for a_n gives

$$a_n = 6a_{n-1} - a_{n-2}.$$

Similarly, we get

$$b_n = 6b_{n-1} - b_{n-2}.$$

Solving these recurrence relations (with appropriate initial conditions), we get

$$a_n = \left(\frac{1}{2} + \frac{1}{\sqrt{8}}\right)(3 + \sqrt{8})^n + \left(\frac{1}{2} - \frac{1}{\sqrt{8}}\right)(3 - \sqrt{8})^n,$$
$$b_n = \frac{1}{\sqrt{8}}(3 + \sqrt{8})^n - \frac{1}{\sqrt{8}}(3 - \sqrt{8})^n.$$

Thus, a_n is the largest entry of $(C(2))^n$, and we conclude that no entry of $(C(2))^n$ is larger than p as long as $n < \log_{3+\sqrt{8}} p$. Since $C(2) = A(2)B(2)$ is a product of two generators, $(C(2))^n$ has length $2n$ as a word in the generators $A(2)$ and $B(2)$. Therefore, no two positive words of length $\leq m$ in the generators $A(2)$ and $B(2)$ (considered as matrices over \mathbb{F}_p) can be equal as long as

$$m < 2 \log_{3+\sqrt{8}} p = \log_{\sqrt{3+\sqrt{8}}} p.$$

In particular, the girth of the Cayley graph of the semigroup generated by $A(2)$ and $B(2)$ (considered as matrices over \mathbb{F}_p) is at least $\log_{\sqrt{3+\sqrt{8}}} p \approx \log_{2.4} p$.

Similarly, in the case of $A(3)$ and $B(3)$, the base of the logarithm is

$\sqrt{\frac{11+\sqrt{117}}{2}} \approx 3.3$. For example, if p is on the order of 2^{256} , then there are no collisions of the form $u(A(3), B(3)) = v(A(3), B(3))$ if positive words u and v are of length less than 149.

Problem

Find an analogue of “Sanov’s form” for the subgroup of $SL(2, \mathbb{Z})$ generated by $A(3)$, $B(3)$.





Since there is no known analog of Sanov’s result for $A(3)$ and $B(3)$, at this time there is no known efficient algorithm for producing relations of length $O(\log p)$ even in the *group* generated by $A(3)$ and $B(3)$, let alone in the *monoid*. We note that by the pigeonhole principle, such relations do in fact exist.

At this time, there are no known attacks on hash functions corresponding to the pair $A(2)$ and $B(2)$ nor on the pair $A(3)$ and $B(3)$. Therefore, there is no visible threat to their security.

Problem

Determine which words in the matrices $A(1), B(2)$ will have the fastest growth of their entries.

This problem is of interest because if we can show the alternating product again has fastest growth, then a similar calculation as was done for $A(2), B(2)$ and for $A(3), B(3)$ would show a lower bound with a smaller base. This means that the base of the logarithm is $\sqrt{2 + \sqrt{3}}$, which is about 1.93. So this would mean that for p on the order of 2^{256} , there will be no collisions of the form $u(A(1), B(2)) = v(A(1), B(2))$ if positive words u and v are of length less than 269. This is a stronger bound than for either the $A(2), B(2)$ case or the $A(3), B(3)$ case.

-  M. Grassl, I. Ilić, S. Magliveras, R. Steinwandt,
Cryptanalysis of the Tillich–Zémor hash function,
J. Cryptology **24** (2011) 148–156
-  C. Petit, J. Quisquater,
Preimages for the Tillich–Zémor hash function,
in SAC 10, Lecture Notes in Comp. Sci. **6544** (2010) 282–301
-  I. N. Sanov,
A property of a representation of a free group (Russian),
Doklady Akad. Nauk SSSR (N.S.) **57** (1947) 657–659
-  J.-P. Tillich and G. Zémor,
Group-theoretic hash functions,
in Proceedings of the First French–Israeli Workshop on Algebraic
Coding, Lecture Notes in Comp. Sci. **781** (1993) 90–110



J.-P. Tillich and G. Zémor,

Hashing with SL_2 ,

in CRYPTO 1994, Lecture Notes in Comp. Sci. **839** (1994) 40–49