PUBLIC KEY EXCHANGE USING MATRICES OVER GROUP RINGS

DELARAM KAHROBAEI, CHARALAMBOS KOUPPARIS, AND VLADIMIR SHPILRAIN

Abstract. We offer a public key exchange protocol in the spirit of Diffie-Hellman, but we use (small) matrices over a group ring of a (small) symmetric group as the platform. This "nested structure" of the platform makes computation very efficient for legitimate parties. We discuss security of this scheme by addressing the Decision Diffie-Hellman (DDH) and Computational Diffie-Hellman (CDH) problems for our platform.

1. Introduction

The beginning of public key cryptography can be traced back to the paper by Diffie and Hellman [2]. The simplest, and original, implementation of their key exchange protocol uses $\mathbb{Z}_p^*$, the multiplicative group of integers modulo a prime p, as the platform. There is also a public element $g \in \mathbb{Z}_p$, which is a primitive root mod p. The protocol itself is as follows:

(1) Alice chooses an integer a, computes $A = g^a \bmod p$ and publishes A
(2) Bob picks an integer b and computes $B = g^b \bmod p$, and publishes B
(3) Alice computes $K_A = B^a \bmod p$
(4) Bob computes $K_B = A^b \bmod p$

Both Alice and Bob are now in possession of a secret shared key K, as $g^{ab} \bmod p = g^{ba} \bmod p$ and hence $K := K_A = K_B$.

The protocol is considered secure provided G and g are chosen properly, see e.g. [5] for details. In order to recover the shared secret key, the eavesdropper Eve must be able to solve the *Diffie-Hellman problem* (recover g^{ab} from g, g^a and g^b). One could solve the Diffie-Hellman problem by solving the discrete logarithm problem, i.e., by recovering a from g and g^a. However, it is unknown whether the discrete logarithm problem is equivalent to the Diffie-Hellman problem.

We should note that there is still the "brute force" method of solving the discrete logarithm problem. The eavesdropper can simply start computing successively higher powers of g, until they match g^a. This requires at most $|g|$ multiplications, where $|g|$ is the order of g in the group G. It is usually the case however that $|g| \approx 10^{300}$ and hence this method is considered computationally infeasible.

Initially it may seem that the legitimate parties, Alice and Bob, will also have to perform a large number of multiplications, thus facing the same problem as the eavesdropper does. However, as the legitimate parties are in possession of a and b, they can use the "square and multiply" algorithm that requires $O(\log_2 a)$ multiplications, e.g. $g^{27} = (((g^2)^2)^2)^2 \cdot ((g^2)^2)^2 \cdot g^2 \cdot g$.

Research of the first author was partially supported by a PSC-CUNY grant from the CUNY research foundation, as well as the City Tech foundation.

Research of the third author was partially supported by the NSF grants DMS-0914778 and CNS-1117675.

There is some disadvantage to working with $\mathbb{Z}_p$, where p, a, and b are chosen to be fairly large. Computation with 300-digit numbers (or 1000-bit binary numbers) is not particularly efficient, and neither is reducing the result modulo p. This is one of the reasons why the Diffie-Hellman key agreement protocol with recommended parameters is not suitable for devices with limited computational resources. Hence, there is an ongoing search for other platforms where the Diffie-Hellman or a similar key exchange can be carried out more efficiently, in particular with public and/or private keys of smaller size.

The platform that we are proposing here is the semigroup of matrices (of a small size) over a group ring, with the usual matrix multiplication operation. More specifically, we are working with matrices over the group ring $\mathbb{Z}_n[S_m]$, where $\mathbb{Z}_n$ is the ring of integers modulo n and S_m is the symmetric group of degree m. To verify the security of using such a semigroup of matrices as the platform, we address the *Computational Diffie-Hellman* and *Decision Diffie-Hellman* problems (Section 3), along with questions about the structure of this semigroup.

Parameters that we suggest (2×2 or 3×3 matrices over $\mathbb{Z}_7[S_5]$) provide for a large key space ($7^{480} \sim 10^{406}$ for 2×2 matrices and $7^{1080} \sim 10^{913}$ for 3×3 matrices). Storing a single 2×2 matrix over $\mathbb{Z}_7[S_5]$ takes about 1440 bits, and a single 3×3 matrix about 3240 bits, so keys are of about the same size as in the "classical" Diffie-Hellman scheme (storing an integer of size about 10^{300} requires 997 bits). These storage requirements can be reduced by $\frac{1}{7}th$ if we do not store polynomial terms which have a 0 as their coefficient, thus bringing the key size down to about 1230 bits for 2×2 matrices and to about 2780 bits for 3×3 matrices.

What we believe is one of the main advantages of our platform over the standard $\mathbb{Z}_p$ platform in the original Diffie-Hellman scheme is that the multiplication of matrices over $\mathbb{Z}_7[S_5]$ is very efficient. In particular, in our setup multiplying elements is faster than multiplying numbers in $\mathbb{Z}_p$ for a large p. This is due to the fact that one can pre-compute the multiplication table for the group S_5 (of order 120), so in order to multiply two elements of $\mathbb{Z}_7[S_5]$ there is no "actual" multiplication in S_5 involved, but just re-arranging a bit string and multiplying coefficients in $\mathbb{Z}_7$. Also, in our multiplication there is no reduction of the result modulo p that slows down computation in $\mathbb{Z}_p$ for a large p. Informally speaking, the "nested structure" of our platform (*small* matrices over a group ring of a *small* group S_5 over a *small* ring $\mathbb{Z}_7$) provide for more efficient computation than just using $\mathbb{Z}_p$ with a very large p.

From a security standpoint, an advantage of our platform over the group $\mathbb{Z}_p$, or elliptic curves, is that "standard" attacks (baby–step giant–step, Pohlig-Hellman, Pollard's rho) do not work with our platform, as we show in Section 6. Furthermore, our platform proves secure against Shor's quantum algorithm which is a common pitfall on classical Diffie-Hellman algorithms, see Section 6.3.

2. Group Rings

Definition 2.1. *Let G be a group written multiplicatively and let R be any commutative ring with nonzero unity. The group ring $R[G]$ is defined to be the set of all formal sums*

$$\sum_{g_i \in G} r_i g_i$$

where $r_i \in R$, and all but a finite number of r_i are zero.

We define the sum of two elements in $R[G]$ by

$$\left(\sum_{g_i \in G} a_i g_i\right) + \left(\sum_{g_i \in G} b_i g_i\right) = \sum_{g_i \in G} (a_i + b_i) g_i.$$

Note that $(a_i + b_i) = 0$ for all but a finite number of i, hence the above sum is in $R[G]$. Thus $(R[G], +)$ is an abelian group.

Multiplication of two elements of $R[G]$ is defined by the use of the multiplications in G and R as follows:

$$\left(\sum_{g_i \in G} a_i g_i\right)\left(\sum_{g_i \in G} b_i g_i\right) = \sum_{g_i \in G}\left(\sum_{g_j g_k = g_i} a_j b_k\right) g_i.$$

As an example of a group ring, we consider the symmetric group S_5 and the ring $\mathbb{Z}_7$ and form the group ring $\mathbb{Z}_7[S_5]$. We will write the identity element of S_m as e. Sample elements and operations are

$$\begin{aligned}
a &= 5(123) + 2(15)(24) + (153) \\
b &= 3(123) + 4(1453) \\
a + b &= (123) + 2(15)(24) + (153) + 4(1453) \\
ab &= (5(123) + 2(15)(24) + (153))(3(123) + 4(1453)) \\
&= 15(132) + 20(145)(23) + 6(14235) + 8(124)(35) + 3(12)(35) + 4(1435) \\
&= (132) + 6(145)(23) + 6(14235) + (124)(35) + 3(12)(35) + 4(1435) \\
ba &= (3(123) + 4(1453))(5(123) + 2(15)(24) + (153)) \\
&= 15(132) + 6(15243) + 3(15)(23) + 20(12)(345) + 8(13)(254) + 4(1345) \\
&= (132) + 6(15243) + 3(15)(23) + 6(12)(345) + (13)(254) + 4(1345)
\end{aligned}$$

Now that group rings have been defined, it is clear how to define $M_2(\mathbb{Z}_n[S_m])$, the ring of 2×2 matrices over the group ring $\mathbb{Z}_n[S_m]$. We are only going to be concerned with multiplication of matrices in this ring; as an example using the same a and b defined above, we can define

$$M_1 = \begin{bmatrix} a & e \\ e & b \end{bmatrix}, M_2 = \begin{bmatrix} b & e \\ 0 & a \end{bmatrix}.$$

Then

$$\begin{aligned}
M_1 M_2 &= \begin{bmatrix} ab & 2a \\ b & e + ba \end{bmatrix} \\
&= \begin{bmatrix} ab & 3(123) + 4(15)(24) + 2(153) \\ 3(123) + 4(1453) & e + ba \end{bmatrix},
\end{aligned}$$

where ab and ba are computed above.

3. Computational Diffie-Hellman and Decision Diffie-Hellman

Recall that in the Diffie-Hellman key exchange Alice and Bob want to establish a secret shared key. Alice chooses a finite group G and an element g of the group G. Alice then picks a random a and publishes (g, G, g^a). Bob also picks a random b and publishes (g^b).

Alice's and Bob's secret key is now g^{ab}, which can be computed by both of them since $g^{ab} = (g^a)^b = (g^b)^a$. The security of the Diffie-Hellman key exchange relies on the assumption that it is computationally hard to recover g^{ab} given (g, G, g^a, g^b).

A passive eavesdropper, Eve, would try to recover g^{ab} from (g, G, g^a, g^b). One defines the Diffie-Hellman algorithm by $F(g, G, g^a, g^b) = g^{ab}$. We say that a group G satisfies the Computational Diffie-Hellman (CDH) assumption if no efficient algorithm exists to compute $F(g, G, g^a, g^b) = g^{ab}$. More precisely,

Definition 3.1. *A CDH algorithm F for a group G is a probabilistic polynomial time algorithm satisfying, for some fixed $\alpha > 0$ and all sufficiently large n,*

$$\mathbb{P}[F(g, G, g^a, g^b) = g^{ab}] > \frac{1}{n^\alpha}.$$

The probability is over a uniformly random choice of a and b. We say that the group G satisfies the CDH assumption if there is no CDH algorithm for G.

Even though a group may satisfy the CDH assumption, CDH by itself is not sufficient to prove that the Diffie-Hellman protocol is useful for practical cryptographic purposes. While Eve may not be able to recover the entire secret, she may still be able to recover valuable information about it. For example, even if CDH is true, Eve may still be able to predict 80% of the bits of g^{ab} with reasonable confidence [1].

Hence if we are using g^{ab} as the shared secret key, one must be able to bound the information Eve can extract about it given g, g^a and g^b. This is formally expressed by the much stronger Decision Diffie-Hellman (DDH) assumption.

Definition 3.2. *A DDH algorithm F for a group G is a probabilistic polynomial time algorithm satisfying, for some fixed $\alpha > 0$ and all sufficiently large n,*

$$\left|\mathbb{P}[F(g, G, g^a, g^b, g^{ab}) = \text{“}True\text{”}] - \mathbb{P}[F(g, G, g^a, g^b, g^c) = \text{“}True\text{”}]\right| > \frac{1}{n^\alpha}.$$

The probability is over a uniformly random choice of a, b and c. We say that the group G satisfies the DDH assumption if there is no DDH algorithm for G.

Essentially, the DDH assumption implies that there is no efficient algorithm which can distinguish between the two probability distributions (g^a, g^b, g^{ab}) and (g^a, g^b, g^c), where a, b and c are chosen at random.

4. Diffie-Hellman key exchange protocol using matrices over $\mathbb{Z}_n[S_m]$

While S_m is a relatively small group for small m, the size of the group ring $\mathbb{Z}_n[S_m]$ grows reasonably fast, even for small values of n and m. This is one reason we chose to look at the Diffie-Hellman key exchange protocol using these group rings. We propose to work with the group ring $\mathbb{Z}_7[S_5]$, which has the size $7^{5!} = 7^{120}$. The next step is to work with matrices over these group rings. Hence, say, the semigroup $M_3(\mathbb{Z}_7[S_5])$ of 3×3 matrices has the order $(7^{5!})^9 \approx 10^{913}$. This semigroup of matrices can now serve as the platform for the Diffie-Hellman key exchange protocol. The procedure Alice and Bob carry out is essentially the same.

Alice chooses a public matrix $M \in M_3(\mathbb{Z}_7[S_5])$ and a private large positive integer a, computes M^a, and publishes (M, M^a). Bob chooses another large integer b, and computes

and publishes (M^b). Both Alice and Bob can now compute the same shared secret key $K = (M^a)^b = (M^b)^a$.

As we have already mentioned in the Introduction, multiplication of matrices in the semigroup $M_3(\mathbb{Z}_7[S_5])$ is very efficient, and, of course, in this semigroup, as in any other semigroup, we can use the "square and multiply algorithm" for exponentiation.

To assess security of our proposal, we should address the two Diffie-Hellman assumptions, CDH and DDH. We investigate the (stronger) DDH assumption experimentally in Section 5.

Finally, some of the algebraic properties of $M_3(\mathbb{Z}_7[S_5])$ will be investigated.

5. Experimental results

The CDH assumption can only be answered theoretically, but the DDH assumption can be investigated experimentally. To construct our matrix semigroups we implemented the necessary group ring procedures in C++. We have the choice of which symmetric group to use and which ring $\mathbb{Z}_n$ to use as well. Next we used a standard uniform distribution implementation to allow for a random selection of an element from our group ring. Finally, we constructed random $k \times k$ matrices over our group ring. Experiments were carried out with various group rings $M_k(\mathbb{Z}_n[S_m])$.

We propose the use of S_5 as the group for our experiments since its underlying structure is understood and simple. When constructing the semigroup $\mathbb{Z}_n[S_5]$, one has the benefits of using the group S_5 as a building block. Namely, the group S_5 has the advantage of having only one normal subgroup, A_5, which has index 2 in S_5. Hence, trying to get some information about a from M^a by applying a non-trivial group homomorphism is limited only to the *sign homomorphism* S_5 to $\mathbb{Z}_2$ of a symmetric group.

We naturally implemented a "square and multiply" routine to speed up computations for exponentiation. With this procedure we can compute high powers of random matrices from our matrix semigroups fairly quickly, see table 1.

We note that the computations were carried out on an Intel Core2 Duo 2.26GHz machine, utilizing only one core, with 4GB of memory and the times were computed as an average time after 250 such exponentiations. No optimizations were in effect and only one processor was used. Thus computational time may be reduced significantly by using more than one core and by implementing any available optimizations for DH using our scheme.

As a comparison for computational times, we refer to recent results of [4] claiming new speed records for DH implementations. In the paper, an implementation of the DH signature exchange protocol over the elliptic curve P-224 is presented. Without any optimization they can carry out 1800 operations per second for the DH protocol, on a somewhat more powerful computer than ours. Recall that in P-224 you require approximately 340 operations for a single "exponentiation". Hence, they require about 0.2 seconds per DH exponentiation versus our 0.6 seconds in $M_2(\mathbb{Z}_7[S_5])$.

One additional thing we noticed was that the speed of computation is independent of the number of nonzero terms in the entries of our matrices M. One possible intuitive explanation is based on the fact that any symmetric group can be generated by a set of 2 particular elements. Since we selected 9 (or 4) random group ring elements for each matrix, there is a high probability that we have selected a pair of group elements that will generate all of

TABLE 1. Speed of Computation

Matrix Size	$\mathbb{Z}_n$	Exponent	Avg. Time (s)
2×2	2	10^{10}	0.06
2×2	3	10^{10}	0.06
2×2	5	10^{10}	0.06
2×2	7	10^{10}	0.06
2×2	2	10^{100}	0.58
2×2	3	10^{100}	0.58
2×2	5	10^{100}	0.58
2×2	7	10^{100}	0.59
2×2	2	10^{1000}	5.97
2×2	3	10^{1000}	6.11
2×2	5	10^{1000}	5.98
2×2	7	10^{1000}	6.66
3×3	2	10^{10}	0.19
3×3	3	10^{10}	0.20
3×3	5	10^{10}	0.20
3×3	7	10^{10}	0.20
3×3	2	10^{100}	1.95
3×3	3	10^{100}	1.95
3×3	5	10^{100}	1.94
3×3	7	10^{100}	1.94
3×3	2	10^{1000}	20.17
3×3	3	10^{1000}	20.15
3×3	5	10^{1000}	19.72
3×3	7	10^{1000}	19.74

our symmetric group. Once we have multiplied M by itself a few times we get group ring elements of random length mixing throughout the matrix entries.

Random group ring elements from $\mathbb{Z}_2[S_5]$ have coefficients either 0 or 1 for each of the 120 elements of S_5. A simple binomial distribution calculation shows that with probability around 93% a random element of this group ring has a total number of nonzero terms between 50 and 70.

5.1. **Experimental results on the Decision Diffie-Hellman assumption.** We should note that for those experiments that were carried out using 2×2 matrices, it is reasonable to assume that if the results hold in the smaller matrix size, they will also hold for 3×3 matrices. In order to test the DDH assumption we need to look at the two distributions: one generated by M^{ab} and the other generated by M^c. Ideally, we would like the two distributions to be indistinguishable. We picked a and b randomly from the interval $[10^{22}, 10^{28}]$, and c randomly from $[10^{44}, 10^{55}]$, so that c had about the same size as the product ab. To get a clearer picture of how different or similar these final matrices were, we looked at each entry of the matrix. For each choice of a random matrix M and random a, b, and c we computed the matrices M^{ab} and M^c. This was repeated 500 times and we created a table that was updated after each run with the distribution of elements of S_5 for each entry of the matrix. We were working with $M_2(\mathbb{Z}_7[S_5])$.

After 500 runs we created Q-Q plots of entries of M^{ab} versus entries of M^c, where we use the notation $M = \left(\begin{smallmatrix} a_1 & a_2 \\ a_3 & a_4 \end{smallmatrix}\right)$. Q-Q plots (or quantile plots) are a graphical method of comparing the quantiles of the cumulative distribution function (cdf) F versus the corresponding quantiles of the cdf G. The functions are parameterized by p, where $p \in [0, 1]$. One axis represents $F^{-1}(p)$ and the other axis represents $G^{-1}(p)$. If the two cdf's are identical, then the Q-Q plot will be that of $y = x$. It will also be a straight line if the distributions are of the same type, but have different mean and standard deviation, see [3] for more details.

As can be seen from Figure 1, it appears that the distributions of each of the matrices M^{ab} and M^c are indeed identical. Thus no information about M^{ab} is leaked by Alice and Bob from choosing their respective a and b, which experimentally confirms the validity of the DDH assumption in our situation.

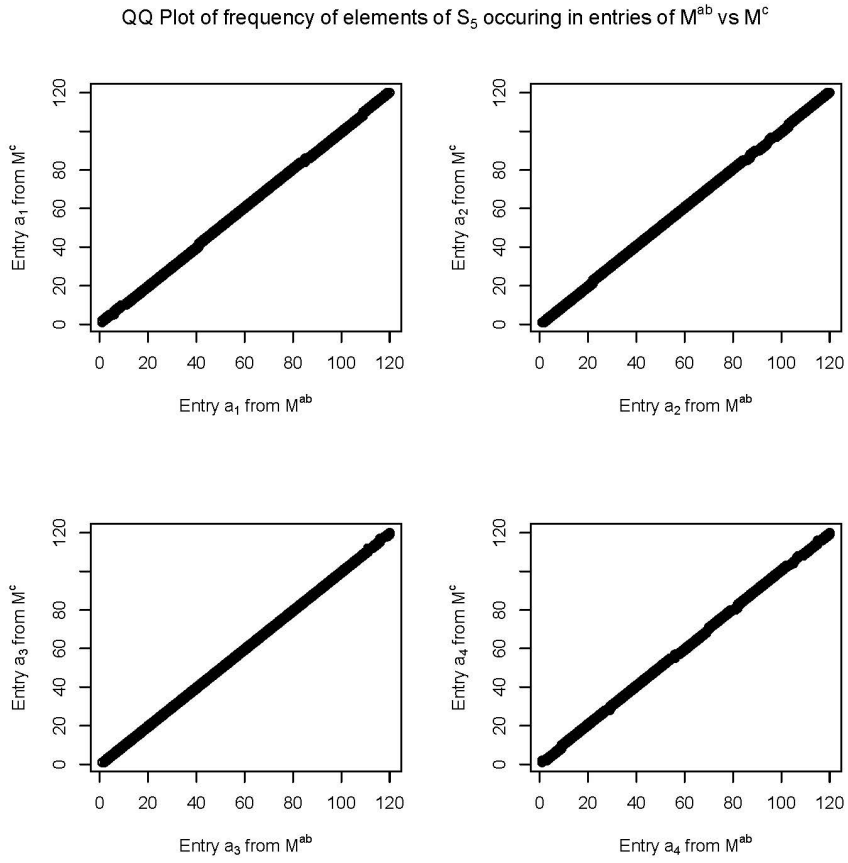

FIGURE 1. DDH results for M^{ab} vs. M^c

Furthermore, we also wanted to verify that no information was leaked about a by publishing M^a, for a given M. The experimental setup was identical to the previous one, only here we chose two random matrices M and N, and a random integer $a \in [10^{44}, 10^{55}]$. Again we produced a Q-Q plot for the two distributions, see figure 2.

From the above plot, it is clear that M^a is indistinguishable from a random matrix N.

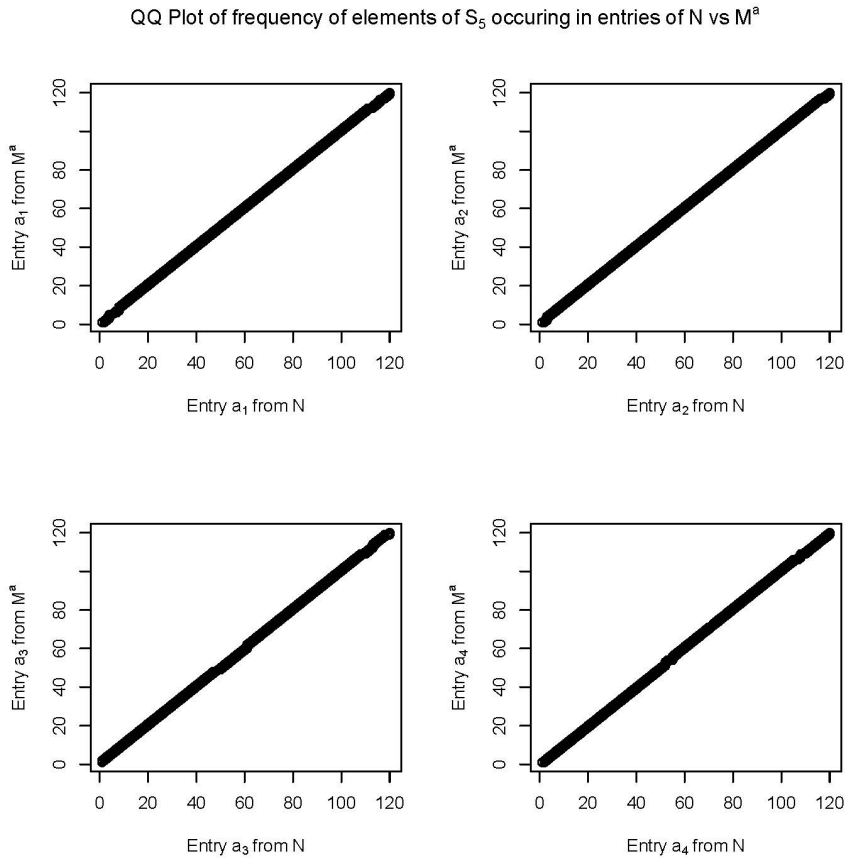

FIGURE 2. DDH results for N vs. M^a

5.2. **Experimental results on low orbits.** Here we address the following "low orbits" question: we want to make sure that powers of matrices in our semigroup do not end up in an orbit of low order. This means that if Alice chooses a random integer a and a random matrix M, we cannot have $M^n = M^k$, for $n < k << a$ (similarly for b chosen by Bob). If this were the case, then the eavesdropper Eve could first determine n and k, then she could find the values of c and d, where $1 \leq c, d \leq k$, such that $M^a = M^c$ and $M^b = M^d$. The shared secret key then could be computed as

$$M^{ab} = (M^a)^b = (M^c)^b = (M^b)^c = (M^d)^c = M^{cd}.$$

This is similar to the problem of finding a generator (i.e., an element of maximum order) in the multiplicative group of $\mathbb{Z}_p$, the original platform for the Diffie-Hellman protocol. Since we are dealing with a semigroup (of matrices) where most elements are not invertible and therefore do not have an "order" in the usual sense, we consider those orbits instead.

The problem of identifying orbits involves actually computing all powers of the given matrix M up to M^a; this makes an important difference with the problem of determining the order

of an invertible element: the latter can be done efficiently if the order of the whole (finite) group is known.

Once each power of M is computed, it needs to be stored and eventually compared to all other powers of M. Currently this is time and space prohibitive, which makes this kind of attack infeasible for the adversary, unless the orbit is of a really small size. One way to cut down the storage space and time required for comparisons is to only store the number of nonzero terms of the group ring element in each entry of the matrix. Thus, for example, for each power of a 2×2 matrix M we can only store 4 integers. This reduces checking for orbits in a list of matrices to checking for orbits in a list of what are essentially 4-dimensional vectors. There is an obvious caveat to this approach: we cannot tell if we have hit an orbit or not since matching the number of terms does not ensure that we actually have matching entries as well. We use this method to search for a match in number of terms, and then we actually compute the appropriate powers if there is a second consecutive match of terms.

Another sacrifice we had to make due to speed and space constraints is the value of a. We need to compute M^a for every integer in $[1, a]$ and also store each vector for the number of terms produced. Finally we have to cycle through this list and check that there are no orbits. We decided to work with $a = 10,000$ and $a = 1,000,000$.

We proceeded to run the low orbit experiment with various group rings. We produced 250 runs for each scenario. The results of the runs can be seen in table 2 below.

Table 2. Orbit match search

Group Ring	Exponent of Matrix	Percent Matches
$M_2(\mathbb{Z}_2[S_5])$	10^4	86.4%
$M_2(\mathbb{Z}_3[S_5])$	10^4	6.8%
$M_2(\mathbb{Z}_5[S_5])$	10^4	0.4%
$M_2(\mathbb{Z}_7[S_5])$	10^4	0.0%
$M_3(\mathbb{Z}_2[S_5])$	10^4	26.4%
$M_3(\mathbb{Z}_3[S_5])$	10^6	2.6%
$M_3(\mathbb{Z}_3[S_7])$	10^6	0.0%

As is apparent in the runs, increasing the order of the algebra decreases collisions the most. Furthermore, increasing the size of the matrices has an additional advantage. We only used 10^4 for the smaller matrices, as this sped up computations and gave us a lower bound for the number of matches when the exponent increases. From the results of the table and given that storing elements of $\mathbb{Z}_5$ and $\mathbb{Z}_7$ requires the same amount of bits, we propose using $\mathbb{Z}_7[S_5]$ as the group ring.

6. "Standard" Attacks

In this section, we discuss why three "standard" attacks on the "classical" discrete logarithm problem do not work with our platform semigroup.

6.1. **Baby–step giant–step algorithm.** One known method of attacking the "classical" discrete logarithm problem, due to Shanks [8], is the baby-step giant-step algorithm. The algorithm computes discrete logarithms in a group of order q in $O\left(\sqrt{q}\ \text{polylog}(q)\right)$ time, where polylog(q) is $O((\log(q))^c)$ for some constant c. If adapted to our situation, this algorithm

would look as follows.

Baby-step giant-step algorithm

Input: M, $A \in M_3(\mathbb{Z}_7[S_5])$, $n = |M_3(\mathbb{Z}_7[S_5])|$
Output: $x \in \mathbb{N}, \ni M^x = A$

Set $s := \lceil \sqrt{n} \rceil$
Set $t := \lceil n/s \rceil$
for $i = 0$ to s
    compute and store (i, AM^i)
for $j = 0$ to t
    compute $M_j = M^{js}$
    if $M_j = AM^i$, for some i, **return** $js - i$

There are a couple of points that have to be made about this algorithm. The first is that we need to produce a good method of storing the matrices. This could be possible with a hash function, in which case insertion and lookup is constant in time. However, our matrices are fairly complex objects, and we need to take into account the storage requirements of the algorithm.

Furthermore, we should note that the order of our chosen random matrix, M, is much smaller than that of the whole group ring. Hence, it may be possible to use a smaller value of n as an input. However, this requires knowledge of the order of M. As little is known about the structure of this group ring, the order is hard to find, and we are not guaranteed that such an order exists in the usual sense. We are basically back to looking for (low) orbit collisions as in our Section 5.2.

Each entry in the matrix can be represented by a sequence of 120 (three-bit) coefficients. We can use a 360 bit string where we encode each three-bit sequence with the value of the coefficient of that polynomial term in $\mathbb{Z}_7[S_5]$. Hence each matrix will need 360×4 bits of storage. In this algorithm we are required to store $\sqrt{|M_3(\mathbb{Z}_7[S_5])|} = \sqrt{7^{540}} \sim 10^{456}$ such matrices. In order to store all these matrices we would need 1440×10^{456} bits of space. This works out to about $10^{446}TB$ of (memory or hard drive) space. Thus, it looks like this algorithm is infeasible already in terms of space. Of course, storing the arrays can be optimized, e.g. we do not need to store entries with zeroes. However, the amount of information that we need to store, 10^{456} matrices, is still too big even if we only store the number of non-zero terms in the polynomials as was done for the low orbit search, see Section 5.2.

One approach often suggested to decrease space requirements is to decrease s, hence increasing t. In this case the algorithm instead of running in $O(\sqrt{n})$ time will run in $O(n/t)$ time. Every time we reduce by half the storage requirements, we end up doubling the running time of the algorithm. However, regardless of what s and t are chosen to be we still need to perform $s + t$ group operations in the two loops. Given our constraints, the number of group operations is minimized when $s = t = \sqrt{n}$. Hence, we need at least 10^{457} group operations to run this algorithm. Given previous results, this is again computationally infeasible.

6.2. **Other attacks.** There are two other algorithms that have been suggested for solving the "classical" discrete logarithm problem. The first is the Pohlig-Hellman algorithm [6]. This algorithm relies on the order of a group element and the generalized Chinese remainder theorem to break the problem into smaller subproblems.

Specifically, suppose the order of the element $g \in G$ is q. In the Diffie-Hellman scheme we wish to find an x such that $g^x = y$. Suppose we know a factorization

$$q = \prod_{i=1}^{n} q_i,$$

where the q_i are relatively prime. Then we have

$$\left(g^{q/q_i}\right)^x = (g^x)^{q/q_i} = y^{q/q_i}, \text{ for } i = 1, ..., n.$$

By the Chinese remainder theorem we can write

$$\mathbb{Z}_q \cong \mathbb{Z}_{q_1} \times \cdots \times \mathbb{Z}_{q_n}$$

and we are left to solve n instances of the discrete logarithm problem in the smaller groups, i.e., defining $g_i = g^{q/q_i}$, we must find the solutions $\{x_i\}_{i=1}^n$ for which $g_i^{x_i} = y^{q/q_i} = g^x$.

However, in our situation the order of matrices in $M_3(\mathbb{Z}_7[S_5])$ does not relate to the size of the whole ring $M_3(\mathbb{Z}_7[S_5])$. Again, under multiplication this ring is a semigroup, not a group, and the proportion of invertible elements in this semigroup is very small. Additionally, the size of this ring is 7^{1080}, so the Chinese remainder theorem does not really help in breaking this problem into smaller parts. If, however, there was a way to break the problem into smaller subproblems, we would still need to solve the discrete logarithm problem in our setting, which so far as we know can only be done via brute force.

The second algorithm proposed for solving the "classical" discrete logarithm problem is Pollard's rho algorithm [7]. The inputs are group elements M and N, and the output is an integer n such that $M^n = N$. The algorithm first looks for an orbit, which has the general form $M^a N^b = M^c N^d$, for a, b, c and $d \in \mathbb{N}$. This is done by using Floyd's cycle-finding algorithm. As long as $b \neq d$, one can take the logarithm with base M to determine n:

$$\begin{aligned} M^a N^b &= M^c N^d \\ \Rightarrow a + b \log_M N &= c + d \log_M N \\ \Rightarrow \frac{a-c}{d-b} &= \log_M N \\ \Rightarrow M^{\frac{a-c}{d-b}} &= N \end{aligned}$$

However, in applying Floyd's cycle-finding algorithm in Pollard's rho attack, the knowledge of the order of the cyclic group generated by M is essential. In our situation, not only is the order of M unknown, but more importantly, since a random M is not going to be invertible with overwhelming probability, order considerations are not applicable, and therefore neither is Pollard's rho attack, at least in its standard form.

6.3. **Quantum Algorithm Attacks.** It is well known that many cryptographic protocols are vulnerable to quantum algorithm attacks [9]. In particular, the Diffie-Hellman protocol can be attacked using Shor's algorithm. This algorithm basically recasts the discrete

logarithm problem as a hidden subgroup problem (HSP) and uses the quantum algorithms developed for HSP to recover the exponent.

We believe that our protocal is secure against such attacks. The HSP relies on the existence of a function $f : G \to S$, for some set S, such that f is constanct on cosets of the unknown subgroup $H \leq G$ and also takes on distinct values for each coset. For the discrete log we define $f : \mathbb{Z}_N \times \mathbb{Z}_N \to G$, such that $f(a, b) = g^a x^b$, where $a, b \in \mathbb{Z}_N$, $g, x \in G$, $g^\alpha = x$ and $|g| = N$. We can rewrite this as $f(a, b) = g^{a+b\cdot log_g x}$, and hence f is constant on the sets $L_c = \{(a, b)|a + b \log_g x = c\}$.

In this setup the hidden subgroup we are seeking is

$$H = L_0 = \{(0, 0), (\log_g x, -1), (2 \log_g x, -2), \cdots , (N \log_g x, -N)\}.$$

To be able to apply this algorithm one would need to know the order of a matrix. However, this is not known a priori and it is also the case that invertible matrices are sparse in our setup. Hence in our setup the function f is ill-defined.

Furthermore, given a random non-invertible matrix it is unlikely that the function f will be distinct on cosets of the subgroup H or even constant on the different cosets. To see this assume M is a non-invertible matrix, then powers of M will either end up in an orbit or will eventually become the zero matrix. If we are in an orbit, assume for example that $M^9 = M^{15}$ and the exponent we are seeking is $\alpha = 12$. The subgroup we are trying to identify is $H = \{(0, 0), (12, -1), (24, -2), (36, -3), \cdots\}$. From the setup we note that $(36, -3) \sim (18, -3)$, but $(18, -3) \notin H$, for if it were then $(36, -3)-(18, -3) = (18, 0) \in H$, which is a contradiction. On the other hand, assume some power of M is the zero matrix, say $M^{20} = 0$, and again $\alpha = 12$. In this case f is no longer constant on the subgroup H as $0 = f(24, -2) \neq f(12, -1) = I$.

7. Conclusions

Our contribution here is proposing the semigroup of matrices (of a small size, 2×2 or 3×3) over the group ring $\mathbb{Z}_7[S_5]$, with the usual matrix multiplication operation, as the platform for the Diffie-Hellman key exchange scheme. What we believe is the main advantage of our platform over the standard $\mathbb{Z}_p^*$ platform in the original Diffie-Hellman scheme is that the multiplication of matrices over $\mathbb{Z}_7[S_5]$ is very efficient. In particular, in our setup multiplying elements is faster than multiplying numbers in $\mathbb{Z}_p$ for a large p. This is due to the fact that one can pre-compute the multiplication table for the group S_5 (of order 120), so in order to multiply two elements of $\mathbb{Z}_7[S_5]$ there is no "actual" multiplication involved, but just re-arrangement of a bit string of length 3×120. Also, no reduction modulo a large p is involved.

To verify the security of using such a semigroup of matrices as the platform, we have experimentally addressed the *Decision Diffie-Hellman* assumption (Section 5) and showed, by using Q-Q plots (or quantile plots) that after 500 runs of the experiment, two distributions, one generated by M^{ab} and the other generated by M^c for a random c, are indistinguishable, thereby experimentally confirming the DDH assumption for our platform. Furthermore, no information is leaked from M^a by comparing it to a random matrix N.

From the security point of view, the advantages of our platform over the $\mathbb{Z}_p$ also include the fact that neither "standard" attacks (baby–step giant–step, Pohlig-Hellman, Pollard's rho) nor quantum algorithm attacks work with our platform, as we showed in Section 6.

References

[1] D. Boneh, *The Decision Diffie-Hellman Problem*, ANTS 1998, pp. 48–63.

[2] W. Diffie and M. E. Hellman, *New Directions in Cryptography*, IEEE Transactions on Information Theory **IT-22** 1976, 644–654.

[3] J. D. Gibbons, S. Chraborti, *Nonparametric Statistical Inference*, CRC Press, 1992.

[4] E. Kasper, *Fast Elliptic Curve Cryptography in OpenSSL*, Financial Cryptography and Data Security, 2011.

[5] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press 1996.

[6] S. Pohlig and M. Hellman, *An Improved Algorithm for Computing Logarithms over GF(p) and its Cryptographic Significance*, IEEE Transactions on Information Theory **IT-24**, 1978, 106–110.

[7] J. Pollard, *Monte Carlo methods for index computation mod p*, Mathematics of Computation **32**, 1978, 331-334.

[8] D. Shanks, *Class number, a theory of factorization and genera*, Analytic Number Theory, Proceedings of Symposia on Pure Mathematics, **20**, American Mathematical Society, 1971, pp. 415–440.

[9] P. Shor, *Algorithms for Quantum Computation: Discrete Logarithms and Factoring*, Proc. 35th Annual Symposium on Foundations of Computer Science (1994). IEEE Comput. Soc. Press, pp. 124–134.

8. Appendix: a challenge

Here we present a challenge relevant to our Diffie-Hellman-like scheme: given explicit 3×3 matrices M, M^a, and M^b over the group ring $\mathbb{Z}_2[S_5]$, recover the matrix M^{ab}. Note that our recommended platform ring is actually $\mathbb{Z}_7[S_5]$, but we believe that breaking our challenge is currently infeasible even for $\mathbb{Z}_2[S_5]$.

Below are the entries for M:

$a_{11} = \epsilon + (243) + (24) + (1234) + (1243) + (124) + (13)(24) + (1324) + (1432) + (142) + (14) + (23) + (13) + (354) + (2354) + (24)(35) + (253) + (254) + (25)(34) + (12)(45) + (12)(345) + (12)(354) + (12)(35) + (12345) + (1235) + (12453) + (1245) + (124)(35) + (12534) + (13542) + (1352) + (1345) + (13)(254) + (13)(25) + (134)(25) + (13425) + (1452) + (142)(35) + (1453) + (145) + (1435) + (14)(235) + (14235) + (14253) + (143)(25) + (14)(253) + (14325) + (14)(25) + (15432) + (1532) + (152)(34) + (15423) + (154)(23) + (15)(23) + (15234) + (15)(234) + (153)(24) + (15324) + (15)(243) + (15)(24)$

$a_{21} = \epsilon + (243) + (24) + (1243) + (1342) + (13)(24) + (1324) + (142) + (1423) + (14)(23) + (13) + (45) + (2345) + (2453) + (245) + (253) + (25) + (2534) + (25)(34) + (12)(345) + (12)(35) + (123)(45) + (1235) + (12453) + (1245) + (125) + (12534) + (13452) + (13542) + (1352) + (13)(45) + (1345) + (135)(24) + (13)(254) + (134)(25) + (1452) + (142)(35) + (1453) + (145) + (14)(35) + (1435) + (145)(23) + (143)(25) + (14)(25) + (1425) + (15432) + (1542) + (152) + (153) + (154) + (1534) + (15)(34) + (15423) + (1523) + (154)(23) + (15)(234) + (153)(24) + (15243) + (15324) + (15)(243) + (1524)$

$a_{31} = (243) + (124) + (1342) + (13)(24) + (1324) + (14) + (1423) + (23) + (12) + (123) + (132) + (345) + (35) + (2453) + (245) + (253) + (12)(45) + (12)(345) + (12)(354) + (12345) + (1245) + (12543) + (1253) + (125) + (12534) + (125)(34) + (13542) + (13)(45) + (135) + (13245) + (135)(24) + (13)(254) + (13254) + (134)(25) + (13425) + (1452) + (1453) + (1435) + (14523) + (14)(235) + (14325) + (14)(25) + (15432) + (152) + (153) + (154) + (15)(34) + (1523) + (153)(24) + (15243) + (15324) + (15)(243) + (1524)$

$a_{12} = (243) + (24) + (12)(34) + (1234) + (1324) + (1432) + (142) + (143) + (1423) + (13) + (45) + (345) + (354) + (23)(45) + (2345) + (235) + (2453) + (245) + (2435) + (253) + (2534) + (25)(34) + (12)(354) + (12)(35) + (12354) + (1245) + (124)(35) + (1253) + (12534) + (132)(45) + (1352) + (13)(45) + (1354) + (13)(245) + (13524) + (13254) + (1325) + (134)(25) + (14532) + (142)(35) + (1453) + (145) + (1435) + (145)(23) + (143)(25) + (14)(253) + (14325) + (14)(25) + (1425) + (15432) + (1532) + (152) + (152)(34) + (153) + (154) + (1523) + (15)(23) + (15234) + (153)(24) + (15324) + (15)(243)$

$a_{22} = (34) + (24) + (1234) + (1243) + (124) + (1342) + (142) + (143) + (14) + (1423) + (132) + (13) + (345) + (354) + (23)(45) + (2354) + (24)(35) + (2543) + (253) + (25) + (2534) + (12)(45) + (12)(354) + (12)(35) + (123)(45) + (124)(35) + (12435) + (12543) + (1253) + (125) + (13452) + (13542) + (13)(45) + (1345) + (1354) + (135) + (13)(245) + (13245) + (13524) + (13)(254) + (13)(25) + (13425) + (1452) + (14352) + (1453) + (14523) + (14)(235) + (14253) + (14)(253) + (14325) + (14)(25) + (1425) + (1532) + (1542) + (15342) + (1543) + (15) + (1534) + (15)(234) + (153)(24) + (15243) + (15324) + (15)(24)$

$a_{32} = (234) + (12)(34) + (1234) + (124) + (1342) + (134) + (1324) + (142) + (1423) + (23) + (12) + (123) + (132) + (45) + (2453) + (24)(35) + (2435) + (2543) + (254) + (2534) + (12)(45) + (1245) + (124)(35) + (1253) + (12534) + (1352) + (1345) + (135) + (13245) + (13524) + (13)(254) + (13)(25) + (13254) + (142)(35) + (1453) + (145) + (14523) + (14)(235) + (14325) + (14)(25) + (15432) + (1542) + (152)(34) + (1543) + (153) + (15) + (1534) + (15423) + (154)(23) + (15)(23) + (153)(24) + (15324) + (15)(243) + (15)(24)$

$a_{13} = \epsilon+(243)+(24)+(12)(34)+(1234)+(1243)+(124)+(13)(24)+(1432)+(142)+(14)+(23)+(12)+(132)+(345)+(35)+(23)(45)+(2345)+(24)(35)+(2435)+(2543)+(253)+(254)+(2534)+(25)(34)+(12)(45)+(12)(354)+(12)(35)+(12345)+(12453)+(12543)+(125)+(132)(45)+(1352)+(1345)+(1354)+(135)+(13)(245)+(13524)+(135)(24)+(13254)+(1325)+(134)(25)+(13425)+(14532)+(1452)+(142)(35)+(14352)+(1453)+(145)+(14)(35)+(1435)+(145)(23)+(14)(235)+(14235)+(143)(25)+(14)(253)+(14325)+(1425)+(15432)+(1532)+(15342)+(152)(34)+(1543)+(154)+(15)+(1534)+(15)(34)+(1523)+(15)(23)+(15234)+(15243)+(1524)$

$a_{23} = \epsilon + (24) + (12)(34) + (124) + (134) + (13)(24) + (1324) + (1432) + (14)(23) + (12) + (123) + (132) + (354) + (35) + (23)(45) + (2345) + (2354) + (245) + (2435) + (25) + (25)(34) + (12)(345) + (12453) + (1245) + (12435) + (12543) + (1253) + (125) + (12534) + (13452) + (13542) + (1352) + (13)(45) + (1345) + (135) + (13245) + (13)(254) + (13)(25) + (134)(25) + (13425) + (14532) + (1452) + (14352) + (1453) + (14)(35) + (14523) + (145)(23) + (14)(235) + (14235) + (143)(25) + (14)(253) + (14325) + (14)(25) + (1532) + (152) + (15342) + (152)(34) + (153) + (154) + (15) + (1534) + (1523) + (154)(23) + (153)(24) + (15)(243) + (1524) + (15)(24)$

$a_{33} = \epsilon+(1243)+(124)+(1342)+(142)+(143)+(1423)+(14)(23)+(123)+(132)+(345)+(35)+(2345)+(245)+(2435)+(2534)+(12)(345)+(123)(45)+(12354)+(1245)+(12543)+(1253)+(1254)+(125)+(125)(34)+(132)(45)+(13542)+(13)(45)+(13)(245)+(13524)+(13)(254)+(13)(25)+(13254)+(1325)+(13425)+(1452)+(14352)+(1453)+(14)(35)+(1435)+(14523)+(145)(23)+(14253)+(14)(253)+(14325)+(14)(25)+(1425)+(1532)+(1542)+(152)+(1543)+(153)+(154)+(15)(34)+(1523)+(154)(23)+(15)(234)+(15243)+(15)(24)$

Below are the entries for M^a:

$a_{11} = \epsilon + (34) + (234) + (24) + (124) + (1342) + (13)(24) + (1324) + (142) + (143) + (14) + (14)(23) + (23) + (12) + (45) + (345) + (245) + (24)(35) + (2435) + (2543) + (25) + (25)(34) + (12)(345) + (12)(35) + (123)(45) + (12345) + (12354) + (1235) + (12453) + (1245) + (124)(35) + (12435) + (1254) + (125) + (132)(45) + (13542) + (13)(45) + (1354) + (135) + (13)(245) + (13245) + (13524) + (135)(24) + (13)(254) + (13)(25) + (1325) + (134)(25) + (13425) + (1452) + (14352) + (1435) + (145)(23) + (14)(235) + (14235) + (14253) + (14)(253) + (14325) + (14)(25) + (1425) + (15432) + (1532) + (1542) + (152) + (15342) + (154) + (15) + (15423) + (1523) + (154)(23) + (15324) + (15)(243) + (1524)$

$a_{21} = (34) + (243) + (124) + (1342) + (134) + (13)(24) + (1324) + (1432) + (143) + (14)(23) + (12) + (13) + (45) + (345) + (23)(45) + (2345) + (235) + (2453) + (24)(35) + (2543) + (254) + (25) + (12345) + (1235) + (12543) + (1253) + (1254) + (12534) + (125)(34) + (132)(45) + (13452) + (1345) + (135) + (13)(245) + (13524) + (135)(24) + (13)(25) + (13425) + (14532) + (14352) + (145) + (14523) + (14)(253) + (14325) + (1425) + (1532) + (152)(34) + (154) + (15)(23) + (15234) + (15243) + (15)(243) + (1524)$

$a_{31} = (234) + (243) + (1243) + (134) + (142) + (14) + (14)(23) + (23) + (123) + (45) + (345) + (354) + (35) + (23)(45) + (2354) + (245) + (24)(35) + (2435) + (2543) + (253) + (254) + (2534) + (12)(45) + (12)(354) + (12)(35) + (12354) + (1235) + (1245) + (13452) + (13542) + (1352) + (13)(45) + (135) + (13)(245) + (13245) + (13)(254) + (13254) + (1325) + (13425) + (14532) + (1452) + (142)(35) + (14352) + (145) + (1435) + (14523) + (14235) + (14325) + (1425) + (152)(34) + (154) + (15) + (15423) + (15243) + (15324)$

$a_{12} = (234) + (1234) + (1243) + (124) + (1342) + (13)(24) + (1432) + (14)(23) + (23) + (123) + (13) + (45) + (354) + (35) + (2345) +$

$(2354)+(235)+(2453)+(24)(35)+(253)+(254)+(25)+(2534)+(12)(45)+(12)(345)+(12)(35)+(123)(45)+(1235)+(124)(35)+(1253)+(1254)+(12534)+(125)(34)+(132)(45)+(1352)+(13)(45)+(1345)+(1354)+(13245)+(135)(24)+(13)(254)+(13)(25)+(13254)+(1325)+(134)(25)+(1452)+(14)(35)+(1435)+(145)(23)+(14)(235)+(14253)+(14)(253)+(14325)+(14)(25)+(1425)+(15432)+(1532)+(1542)+(152)+(152)(34)+(15)+(1523)+(154)(23)+(15)(23)+(15324)+(15)(243)+(15)(24)$

$a_{22} = (34)+(243)+(12)(34)+(1234)+(1243)+(124)+(134)+(1324)+(14)(23)+(12)+(123)+(132)+(13)+(345)+(23)(45)+(2345)+(2354)+(235)+(2453)+(245)+(24)(35)+(253)+(2534)+(25)(34)+(12)(345)+(12)(354)+(123)(45)+(12354)+(1235)+(12453)+(1245)+(12543)+(1254)+(13452)+(13542)+(1352)+(13)(45)+(135)+(13524)+(13)(254)+(13254)+(1325)+(134)(25)+(14532)+(1452)+(142)(35)+(145)+(14)(35)+(14)(235)+(14253)+(143)(25)+(14)(253)+(14325)+(15432)+(1532)+(1542)+(152)(34)+(15)+(15423)+(1523)+(154)(23)+(15243)+(15324)+(15)(243)+(15)(24)$

$a_{32} = (34)+(243)+(24)+(12)(34)+(124)+(1342)+(134)+(1324)+(142)+(143)+(1423)+(14)(23)+(23)+(132)+(45)+(345)+(35)+(23)(45)+(235)+(245)+(24)(35)+(2435)+(2543)+(25)+(2534)+(25)(34)+(12)(354)+(12345)+(1235)+(12435)+(1254)+(132)(45)+(1352)+(13)(45)+(1345)+(135)+(13)(245)+(13245)+(13)(254)+(13)(25)+(13254)+(134)(25)+(13425)+(1452)+(14352)+(1453)+(14)(35)+(1435)+(145)(23)+(14)(235)+(14253)+(143)(25)+(14)(253)+(14)(25)+(1425)+(1532)+(152)+(15342)+(152)(34)+(1543)+(15)(34)+(1523)+(154)(23)+(15)(23)+(153)(24)+(15243)+(15324)+(1524)$

$a_{13} = \epsilon+(34)+(1243)+(1342)+(1324)+(14)+(1423)+(14)(23)+(12)+(132)+(45)+(354)+(35)+(23)(45)+(2345)+(24)(35)+(2543)+(254)+(2534)+(12)(45)+(12)(35)+(12354)+(1235)+(125)(34)+(1345)+(13524)+(135)(24)+(13254)+(134)(25)+(14532)+(1453)+(14523)+(143)(25)+(14)(25)+(15432)+(1532)+(152)+(15342)+(153)+(15)+(15423)+(153)(24)+(15)(243)$

$a_{23} = \epsilon+(34)+(24)+(1243)+(124)+(14)+(23)+(123)+(132)+(13)+(45)+(354)+(23)(45)+(2453)+(245)+(24)(35)+(2435)+(2543)+(253)+(254)+(25)+(25)(34)+(12)(35)+(12345)+(12453)+(124)(35)+(1254)+(125)+(132)(45)+(13542)+(1352)+(13)(45)+(1345)+(1354)+(135)+(13)(245)+(13524)+(135)(24)+(1325)+(13425)+(142)(35)+(145)+(14)(35)+(1435)+(145)(23)+(14)(235)+(14253)+(14)(253)+(14)(25)+(1425)+(1542)+(152)+(152)(34)+(153)+(15)(34)+(1523)+(154)(23)+(15234)+(15324)+(1524)+(15)(24)$

$a_{33} = (12)(34)+(1234)+(1243)+(1342)+(134)+(13)(24)+(1324)+(143)+(14)+(1423)+(23)+(12)+(123)+(13)+(354)+(35)+(23)(45)+(235)+(2435)+(2534)+(25)(34)+(12345)+(1235)+(124)(35)+(12435)+(1253)+(1254)+(12534)+(125)(34)+(13452)+(13542)+(1352)+(1345)+(1354)+(135)+(13)(245)+(13245)+(135)(24)+(13)(25)+(142)(35)+(14352)+(1435)+(14)(235)+(14253)+(143)(25)+(14)(253)+(14325)+(14)(25)+(1425)+(1532)+(15342)+(1543)+(154)+(15)+(1534)+(15)(34)+(154)(23)+(15)(234)+(15324)+(15)(243)+(15)(24)$

Below are the entries for M^b:

$a_{11} = \epsilon+(34)+(234)+(24)+(124)+(1342)+(13)(24)+(1324)+(142)+(143)+(14)+(14)(23)+(23)+(12)+(45)+(345)+(245)+(24)(35)+(2435)+(2543)+(25)+(25)(34)+(12)(345)+(12)(35)+(123)(45)+(12345)+(12354)+(1235)+(12453)+(1245)+(124)(35)+$

$(12435)+(1254)+(125)+(132)(45)+(13542)+(13)(45)+(1354)+(135)+(13)(245)+(13245)+(13524)+(135)(24)+(13)(254)+(13)(25)+(1325)+(134)(25)+(13425)+(1452)+(14352)+(1435)+(145)(23)+(14)(235)+(14235)+(14253)+(14)(253)+(14325)+(14)(25)+(1425)+(15432)+(1532)+(1542)+(152)+(15342)+(154)+(15)+(15423)+(1523)+(154)(23)+(15324)+(15)(243)+(1524)$

$a_{21} = (34)+(243)+(124)+(1342)+(134)+(13)(24)+(1324)+(1432)+(143)+(14)(23)+(12)+(13)+(45)+(345)+(23)(45)+(2345)+(235)+(2453)+(24)(35)+(2543)+(254)+(25)+(12345)+(1235)+(12543)+(1253)+(1254)+(12534)+(125)(34)+(132)(45)+(13452)+(1345)+(135)+(13)(245)+(13524)+(135)(24)+(13)(25)+(13425)+(14532)+(14352)+(145)+(14523)+(14)(253)+(14325)+(1425)+(1532)+(152)(34)+(154)+(15)(23)+(15234)+(15243)+(15)(243)+(1524)$

$a_{31} = (234)+(243)+(1243)+(134)+(142)+(14)+(14)(23)+(23)+(123)+(45)+(345)+(354)+(35)+(23)(45)+(2354)+(245)+(24)(35)+(2435)+(2543)+(253)+(254)+(2534)+(12)(45)+(12)(354)+(12)(35)+(12354)+(1235)+(1245)+(13452)+(13542)+(1352)+(13)(45)+(135)+(13)(245)+(13245)+(13)(254)+(13254)+(1325)+(13425)+(14532)+(1452)+(142)(35)+(14352)+(145)+(1435)+(14523)+(14235)+(14325)+(1425)+(152)(34)+(154)+(15)+(15423)+(15243)+(15324)$

$a_{12} = (234)+(1234)+(1243)+(124)+(1342)+(13)(24)+(1432)+(14)(23)+(23)+(123)+(13)+(45)+(354)+(35)+(2345)+(2354)+(235)+(2453)+(24)(35)+(253)+(254)+(25)+(2534)+(12)(45)+(12)(345)+(12)(35)+(123)(45)+(1235)+(124)(35)+(1253)+(1254)+(12534)+(125)(34)+(132)(45)+(1352)+(13)(45)+(1345)+(1354)+(13245)+(135)(24)+(13)(254)+(13)(25)+(13254)+(1325)+(134)(25)+(1452)+(14)(35)+(1435)+(145)(23)+(14)(235)+(14253)+(14)(253)+(14325)+(14)(25)+(1425)+(15432)+(1532)+(1542)+(152)+(152)(34)+(15)+(1523)+(154)(23)+(15)(23)+(15324)+(15)(243)+(15)(24)$

$a_{22} = (34)+(243)+(12)(34)+(1234)+(1243)+(124)+(134)+(1324)+(14)(23)+(12)+(123)+(132)+(13)+(345)+(23)(45)+(2345)+(2354)+(235)+(2453)+(245)+(24)(35)+(253)+(2534)+(25)(34)+(12)(345)+(12)(354)+(123)(45)+(12354)+(1235)+(12453)+(1245)+(12543)+(1254)+(13452)+(13542)+(1352)+(13)(45)+(135)+(13524)+(13)(254)+(13254)+(1325)+(134)(25)+(14532)+(1452)+(142)(35)+(145)+(14)(35)+(14)(235)+(14253)+(143)(25)+(14)(253)+(14325)+(15432)+(1532)+(1542)+(152)(34)+(15)+(15423)+(1523)+(154)(23)+(15243)+(15324)+(15)(243)+(15)(24)$

$a_{32} = (34)+(243)+(24)+(12)(34)+(124)+(1342)+(134)+(1324)+(142)+(143)+(1423)+(14)(23)+(23)+(132)+(45)+(345)+(35)+(23)(45)+(235)+(245)+(24)(35)+(2435)+(2543)+(25)+(2534)+(25)(34)+(12)(354)+(12345)+(1235)+(12435)+(1254)+(132)(45)+(1352)+(13)(45)+(1345)+(135)+(13)(245)+(13245)+(13)(254)+(13)(25)+(13254)+(134)(25)+(13425)+(1452)+(14352)+(1453)+(14)(35)+(1435)+(145)(23)+(14)(235)+(14253)+(143)(25)+(14)(253)+(14)(25)+(1425)+(1532)+(152)+(15342)+(152)(34)+(1543)+(15)(34)+(1523)+(154)(23)+(15)(23)+(153)(24)+(15243)+(15324)+(1524)$

$a_{13} = \epsilon+(34)+(1243)+(1342)+(1324)+(14)+(1423)+(14)(23)+(12)+(132)+(45)+(354)+(35)+(23)(45)+(2345)+(24)(35)+(2543)+(254)+(2534)+(12)(45)+(12)(35)+(12354)+(1235)+(125)(34)+(1345)+(13524)+(135)(24)+(13254)+(134)(25)+(14532)+(1453)+(14523)+(143)(25)+(14)(25)+(15432)+(1532)+(152)+(15342)+(153)+(15)+(15423)+(153)(24)+(15)(243)$

$$a_{23} = \epsilon + (34) + (24) + (1243) + (124) + (14) + (23) + (123) + (132) + (13) + (45) + (354) + (23)(45) + (2453) + (245) + (24)(35) + (2435) + (2543) + (253) + (254) + (25) + (25)(34) + (12)(35) + (12345) + (12453) + (124)(35) + (1254) + (125) + (132)(45) + (13542) + (1352) + (13)(45) + (1345) + (1354) + (135) + (13)(245) + (13524) + (135)(24) + (1325) + (13425) + (142)(35) + (145) + (14)(35) + (1435) + (145)(23) + (14)(235) + (14253) + (14)(253) + (14)(25) + (1425) + (1542) + (152) + (152)(34) + (153) + (15)(34) + (1523) + (154)(23) + (15234) + (15324) + (1524) + (15)(24)$$

$$a_{33} = (12)(34) + (1234) + (1243) + (1342) + (134) + (13)(24) + (1324) + (143) + (14) + (1423) + (23) + (12) + (123) + (13) + (354) + (35) + (23)(45) + (235) + (2435) + (2534) + (25)(34) + (12345) + (1235) + (124)(35) + (12435) + (1253) + (1254) + (12534) + (125)(34) + (13452) + (13542) + (1352) + (1345) + (1354) + (135) + (13)(245) + (13245) + (135)(24) + (13)(25) + (142)(35) + (14352) + (1435) + (14)(235) + (14253) + (143)(25) + (14)(253) + (14325) + (14)(25) + (1425) + (1532) + (15342) + (1543) + (154) + (15) + (1534) + (15)(34) + (154)(23) + (15)(234) + (15324) + (15)(243) + (15)(24)$$

CUNY Graduate Center and City Tech, City University of New York
E-mail address: DKahrobaei@GC.Cuny.edu

CUNY Graduate Center, City University of New York
E-mail address: ckoupparis@GC.Cuny.edu

The City College of New York and CUNY Graduate Center
E-mail address: shpil@groups.sci.ccny.cuny.edu