Recall that we can use the SIS problem to create one way functions: any matrix $A \in \mathbb{Z}_q^{n \times m}$ (with $m \geq 2n \log q$) gives the one-way function $f_A : \{0,1\}^m \rightarrow \mathbb{Z}_q^n$ defined by $f_A(x) = Ax \mod q$. Here $n$ is the main security parameter and might, for example, be 128, 256, or something larger. If, for simplicity, we take $q = n = 2^8$ (and $m = 2n \log q$), then the key size (i.e. the size of the matrix $A$) is $nm = 2^{18}$ bytes, i.e. 256 kilobytes, which is prohibitively large. To remedy this, we will only choose matrices of a specific form which can be compactly represented, and for which computing matrix products can be done efficiently.

### 1 Ring SIS

When choosing a matrix $A \in \mathbb{Z}_q^{n \times m}$ (with $m = 2n \log q$), we can break $A$ up into $n \times n$ blocks $(A_1 | A_2 | \cdots | A_{2\log q})$ and choose each $A_i$ such that it has a compact representation. We will choose $A_i$’s of the form

$$
\begin{pmatrix}
a_0 & a_1 & a_2 & \ldots & a_{n-1} \\
a_{n-1} & a_0 & a_1 & \ldots & a_{n-2} \\
a_{n-2} & a_{n-1} & a_0 & \ldots & a_{n-3} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
a_1 & a_2 & a_3 & \ldots & a_0
\end{pmatrix}
$$

for some $a_0, a_1, \ldots, a_n \in \mathbb{Z}_q$. A matrix of this form is called a circulant matrix. Note that $A_i$ may be represented by the vector $(a_0, a_1, \ldots, a_{n-1})$ since all other rows of $A_i$ can be obtained by cyclically permuting this vector. Furthermore, we will show that the product of two circulant matrices can be computed in $O(n \log n)$ time using the Fast Fourier Transform.

To see this, first note that the set $C \subseteq \mathbb{Z}_q^{n \times n}$ of circulant matrices is closed under multiplication: If $R_a, R_b$ are circulant matrices whose first rows are $a = (a_0, a_1, \ldots, a_{n-1})$ and $b = (b_0, b_1, \ldots, b_{n-1})$, respectively, then one can check (for example, by observing that the $(i,j)$ entry of $R_a$ is $a_k$ where $k \in \{0,1,\ldots,n-1\}$ and $k \equiv j - i \mod n$, and similarly for $R_b$) that $R_a \cdot R_b = R_c$, where $c = (c_0, c_1, \ldots, c_{n-1})$ with

$$
c_k = \sum_{i+j=k \mod n} a_i b_j.
$$

Now, consider the ring $\mathbb{Z}_q[x]/(x^n - 1)$. For notational simplicity, we identify each element of $\mathbb{Z}_q[x]/(x^n - 1)$ (i.e. each coset of $(x^n - 1)$) with its unique representative which is a polynomial of degree $\leq n - 1$. We define a function $\varphi : \mathbb{Z}_q[x]/(x^n - 1) \rightarrow C$ by

$$
\varphi(a_0 + a_1 x + \cdots + a_{n-1} x^{n-1}) = R_a, \quad \text{where } a = (a_0, a_1, \ldots, a_{n-1})
$$
It is clear that $\varphi$ is a bijection and that

$$\varphi \left( \sum_{i=0}^{n-1} a_i x^i + \sum_{i=0}^{n-1} b_i x^i \right) = \varphi \left( \sum_{i=0}^{n-1} a_i x^i \right) + \varphi \left( \sum_{i=0}^{n-1} b_i x^i \right).$$

Also, our reasoning above shows that

$$\varphi \left( \left( \sum_{i=0}^{n-1} a_i x^i \right) \cdot \left( \sum_{i=0}^{n-1} b_i x^i \right) \right) = \varphi \left( \sum_{i=0}^{n-1} a_i x^i \right) \cdot \varphi \left( \sum_{i=0}^{n-1} b_i x^i \right).$$

Thus, $\varphi$ is a ring isomorphism. In particular, multiplying two circulant matrices is the same as multiplying two polynomials in $\mathbb{Z}_q[x]/(x^n - 1)$, which we can do efficiently using the Fast Fourier Transform.

## 2 Fast Fourier Transform

Suppose $n$ is a power of 2. Given two polynomials $A(x), B(x) \in \mathbb{Z}_q[x]/(x^n - 1)$ with $A(x) = \sum_{i=0}^{n-1} a_i x^i, B(x) = \sum_{i=0}^{n-1} b_i x^i$, we compute their product $A(x)B(x) \in \mathbb{Z}_q[x]/(x^n - 1)$ as follows:

1. Fix a primitive $n$th root of unity $\omega$.
2. Evaluate $A(\omega^i), B(\omega^i)$ for all $i \in \{0, 1, \ldots, n-1\}$.
3. Use $A(\omega^i), B(\omega^i)$ to compute $AB(\omega^i) = A(\omega^i)B(\omega^i)$ for all $i$.
4. Use the $AB(\omega^i)$ to find $c_0, c_1, \ldots, c_{n-1} \in \mathbb{Z}_q$ such that $A(x)B(x) = \sum_{i=0}^{n-1} c_i x^i$.

To do step 2, we write $A(x) = A_0(x^2) + xA_1(x^2)$, where

$$A_0(x) = \sum_{i=0}^{n/2-1} a_{2i} x^i, \quad A_1(x) = \sum_{i=0}^{n/2-1} a_{2i+1} x^i.$$

Now, to evaluate $A(x)$ at $1, \omega, \omega^2, \ldots, \omega^{n-1}$, we can simply evaluate $A_0(x)$ and $A_1(x)$ at $1, \omega^2, (\omega^2)^2, \ldots, (\omega^2)^{n-2}$ and put these results together using the fact that $A(x) = A_0(x^2) + xA_1(x^2)$. Since $\omega$ is an $n$th root of unity, we have that $\omega^{2i} = \omega^{2(i-\frac{n}{2})}$ for all $i$. Thus, we only have to evaluate $A_0(x)$ and $A_1(x)$ at the $\frac{n}{2}$ distinct points $1, \omega^2, (\omega^2)^2, \ldots, (\omega^2)^{\frac{n}{2}-1}$. By continuing this process on $A_0$ and $A_1$, we get a divide-and-conquer algorithm that computes $A(\omega^i)$ for all $i \in \{0, 1, \ldots, n-1\}$ in $O(n \log n)$ time.

For step 4, assume we have computed $AB(\omega^i) = A(\omega^i)B(\omega^i)$ for all $i$. Suppose $A(x)B(x) = \sum_{i=0}^{n-1} c_i x^i$. Recall that this equality is actually happening in $\mathbb{Z}_q[x]/(x^n - 1)$, so as elements
of $\mathbb{Z}_q[x]$ we have $A(x)B(x) = \sum_{i=0}^{n-1} c_i x^i + (x^n - 1)f(x)$ for some $f(x) \in \mathbb{Z}_q[x]$. However, since $(x^n - 1)$ vanishes at each $\omega^j$, it follows that evaluating $\sum_{i=0}^{n-1} c_i x^i$ at $\omega^j$ gives us precisely $A(\omega^j)B(\omega^j)$.

Hence, we have the following matrix equation

$$
\begin{pmatrix}
1 & 1 & 1 & \ldots & 1 \\
1 & \omega & \omega^2 & \ldots & \omega^{n-1} \\
1 & \omega^2 & (\omega^2)^2 & \ldots & (\omega^2)^{n-1} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & \omega^{n-1} & (\omega^{n-1})^2 & \ldots & (\omega^{n-1})^{n-1}
\end{pmatrix}
\begin{pmatrix}
c_0 \\
c_1 \\
c_2 \\
\vdots \\
c_{n-1}
\end{pmatrix}
= 
\begin{pmatrix}
AB(1) \\
AB(\omega) \\
AB(\omega^2) \\
\vdots \\
AB(\omega^{n-1})
\end{pmatrix}. 
$$

Let $V$ denote the leftmost matrix above. Our work in step 2 gave us an efficient way to calculate $AB(1), AB(\omega), \ldots, AB(\omega^{n-1})$ if we know $c_0, c_1, \ldots, c_{n-1}$. However, now we know $AB(1), AB(\omega), \ldots, AB(\omega^{n-1})$ and want to compute $c_0, c_1, \ldots, c_{n-1}$. We can accomplish this by multiplying both sides of (1) by $V^{-1}$. Now, we claim that $V^{-1} = \frac{1}{n} \overline{V}$. To see this, note that since every entry in $V$ has absolute value 1, taking the conjugate of $V$ amounts to inverting every entry, i.e.

$$
\overline{V} = 
\begin{pmatrix}
1 & 1 & 1 & \ldots & 1 \\
1 & \omega^{-1} & \omega^{-2} & \ldots & \omega^{-(n-1)} \\
1 & \omega^{-2} & (\omega^{-2})^2 & \ldots & (\omega^{-2})^{n-1} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & \omega^{-(n-1)} & (\omega^{-(n-1)})^2 & \ldots & (\omega^{-(n-1)})^{n-1}
\end{pmatrix}.
$$

Thus, the $(i, j)$ entry of $V\overline{V}$ is

$$
\sum_{k=0}^{n-1} (\omega^k)^{i-j} = \begin{cases} 
  n & \text{if } i = j \\
  0 & \text{if } i \neq j
\end{cases},
$$

where the second case follows from the fact that if $i \neq j$, then $\omega^{i-j} \neq 1$ while

$$(\omega^{i-j} - 1) \sum_{k=0}^{n-1} (\omega^k)^{i-j} = 0.$$

It follows that $V^{-1} = \frac{1}{n} \overline{V}$, as claimed. Observe that $\overline{V}$ has the same form as $V$ but with $\omega$ replaced by $\omega^{-1}$ (another primitive $n$th root of unity). So to compute the product

$$
\begin{pmatrix}
c_0 \\
c_1 \\
c_2 \\
\vdots \\
c_{n-1}
\end{pmatrix}
= 
V^{-1}
\begin{pmatrix}
AB(1) \\
AB(\omega) \\
AB(\omega^2) \\
\vdots \\
AB(\omega^{n-1})
\end{pmatrix}
$$
we may simply use the same algorithm as in part 2, but with using $\omega^{-1}$ as our primitive $n$th root of unity rather than $\omega$. Thus, step 4 can be done in $O(n \log n)$ time as well. Overall, this gives a $O(n \log n)$ time algorithm for multiplying polynomials in $\mathbb{Z}_q[x]/(x^n - 1)$.

In [2], Micciancio shows that Ring SIS, as described above, is a one-way function. However, Lyubashevsky and Micciancio show in [1] that this version of Ring SIS is not collision resistant. They also show that if we replace $\mathbb{Z}_q[x]/(x^n - 1)$ by $\mathbb{Z}_q[x]/(x^n + 1)$ (with $n$ a power of 2) in all the constructions above, then we get both one-wayness and collision resistance.

References
