Message Recovery Attack on ACES

Samuel Jaques

April 24, 2024

https://arxiv.org/abs/2401.13255 (hereinafter "the paper") is an interesting and ambitious attempt to use category theory to unify homomorphic properties in different cryptographic schemes. It also proposes a bootstrapping-free FHE scheme called ACES, which unfortunately has an efficient key recovery attack, detailed below.

1 ACES

Translating the ACES cryptosystem into more familiar terms for lattice cryptographers, version 2 is similar to a module LWE scheme with some NTRU-like elements. The usual notation for module LWE is with a ring

$$R_q = \mathbb{Z}_q[x]/(u(x)). \tag{1}$$

The paper uses $\mathbb{Z}_q[x]_u$ to refer to this ring. The scheme is parameterized by q, u, a root ω such that $u(\omega) \equiv 0 \mod q$, module parameters k_1 and k_2, and a second smaller prime p. This is in fact a slight generalization of the scheme, which used $k_2 = 1$.

KeyGen: Alice selects a random matrix $A \in R_q^{k_2 \times k_1}$, a secret vector $s \in \mathbb{R}_q^{k_1}$, and a secret error $e \in R_q^{k_2}$, but chosen specifically so that each component e_i satisfies $e_i(\omega) \equiv kp \mod q$. She outputs an LWE public key $(A, b = As + e)$ and retains s as the secret key.

(in the notation of the paper, $f_0 = A$, $f_1 = b$, $x = s$, $e = e$, and $n = k_1$).

Encrypt: Bob selects a random vector of polynomials $r \in R_q^{k_2}$, a random error $e' \in R_q$ such that $e'(\omega) \equiv kp \mod q$ for some k, and a random message encoding polynomial I'll call $n \in R_q$, chosen such that $n(\omega) \equiv m \mod q$, where m is the message he wants to encrypt. He computes $c_1 = r^T A \in R_q^{k^2}$ and $c_2 = r^T b + e' + n \in R_q$ and outputs (c_1, c_2).

(in the notation of the paper, $b = r$ and $r(m) = n$ and $e' = e$

Decrypt: Alice receives c_1 and c_2 and computes $c_2 - c_1 s \in R_q$ and evalutes this polynomial at ω. The result will be $m \mod q$.

Homomorphisms: To compute homomorphisms, Alice's public key also includes a 3-tensor $\lambda = \lambda_{ij}^k \in \mathbb{Z}_q^{k_1^3}$ satisfying

$$s_i \cdot s_j = \sum_{k=1}^{k_1} \lambda_{ij}^k s_k \tag{2}$$

1.1 Break

We can evaluate the homomorphism equation at the root ω. This gives us

$$s_i(\omega) \cdot s_j(\omega) = \sum_{k=1}^{k_1} \lambda_{ij}^k s_k \tag{3}$$

For notational convenience, I'll let $S_i = s_i(\omega)$. This is an element of $\mathbb{Z}_q$.

Fix some $i \in \{1, \ldots, k_1\}$. Then we see that

$$S_i S_j = \sum_{k=1}^{k_1} \lambda_{ij}^k S_k \tag{4}$$

Let Λ_i be the matrix where the entry in the jth row and kth column is λ_{ij}^k, and let S be the vector $(S_1, \dot{,} S_{k_1})$. Then we can see that the above equation becomes

$$S_i S = \Lambda_i S \tag{5}$$

In other words, S_i is an eigenvalue of Λ_i.

Since Λ_i is public, if q is a prime it is straightforward to find its eigenvalues. One of them is the ith component of the secret. More important, the associated eigenvector is the entire secret itself!

The remaining question is how to decide which eigenvector is the secret. There are many methods; since we will only have n eigenvectors at most, we could simply try using them to decrypt. I used a slightly different method of filtering through eigenvectors and eigenvalues:

1. First, the eigenvalue must lie in $\mathbb{Z}_q$, not an extension field.
2. Second, if we used Λ_i, we know that the associated eigenvalue must be the ith entry of the eigenvector. Thus, we can normalize the eigenvector.
3. Second, we know that the homomorphism equation must hold generally, so we can test it for all i, j, k.
4. Third, we know that the public key (A, b) satisfies $b = As + e$, and evaluted at all entries, $b(\omega) \equiv A(\omega)s(\omega) + e(\omega) \equiv A(\omega)s(\omega) + kp \mod q$. Thus, we test whether this holds.

Once this all holds, we have found $s(\omega)$. This is enough to recover all messages, since we can simply compute $c_2(\omega) - c_1(\omega)s(\omega) \mod q$ in the decryption step because evaluation at ω is a homomorphism.

1.2 Composite moduli

The above description works for prime moduli, and this is what the sage script implements. This should extend straightforwardly to square-free composite moduli by reducing by modulo each factor, and possibly would work for prime-power moduli (and hence all moduli) with Hensel lifting.