FHE-MPC Notes

Lecturer: Nigel Smart
Scribe: Jake Loftus

Lecture # 5

1 Modular reduction

For an integer t we let $[t]_q$ denote the reduction $t \pmod q$ into the interval $[-q/2, q/2)$. This can be computed as $t - q \cdot \lfloor t/q \rceil$ (e.g $15 \pmod 7 = 1 = 15 - 7 \cdot \lfloor 15/7 \rceil$ and $8 \pmod 5 = 8 - 5 \cdot \lfloor 8/5 \rceil = -2$). Suppose q is an odd modulus and t is an integer. Then we can compute $[t]_q \pmod 2$ as

$$\begin{aligned} [t]_q &= t - q \cdot \lfloor t/q \rceil \pmod 2 \\ &= t - \lfloor t/q \rceil \pmod 2 \end{aligned}$$

Now express t/q in binary expansion *i.e* $t/q = \Sigma_{i=-\infty}^{\infty} e_i \cdot 2^i$ where all but finitely many of the e_i are zero. If we round this and take the result $\pmod 2$ then in fact the result is simply the xor of the bits either side of the decimal point *i.e* $[t]_q \pmod 2 = e_0 \oplus e_{-1}$. This can easily be seen by writing out a few examples.

Suppose now we wish to compute the sum of l integers $[\Sigma t_i]_q \pmod 2$. First compute a big table of binary expansions as

$$\begin{array}{ll} t_1/q & \ldots 11.01001 \ldots \\ t_2/q & \ldots 01.10011 \ldots \\ \vdots & \qquad \vdots \\ t_l/q & \ldots 10.11010 \ldots \end{array}$$

Then because of carries we can't just xor things and add. We actually need to retain the approximately first $log_2(l)$ bits. This addition and rounding is essentially the non-linear part of the bootstrapping operation.

2 Bootstrapping

The key idea behind bootstrapping is to evaluate the decryption circuit homomorphically resulting in a clean encryption of the original message. In the original SV/GH schemes [3], [2] this was achieved by augmenting the public key with some additional information, namely integers $\{x_i\}_{i=0}^{i=S}$ such that s of those integers add up to the secret key w for the initial somewhat homomorphic encryption scheme with $s << S$. The secret key $\{\sigma_i\}_{i=0}^{i=S}$ is now the characteristic vector of that sparse subset. We also include encryptions $\mathfrak{c}_i = \mathsf{Enc}(\sigma_i, pk)$ of the new secret key bits in the public key. Note $w = \Sigma_0^S \sigma_i \cdot x_i \pmod d$.

Bootstrapping then procedes in the following stages:

- Write down a matrix of s by $p = \lceil log_2(s+1) \rceil$ bits $\{b_{i,j}\}$ which correspond to the first p bits in the binary expansion of cx_i/d (similar to above, now as a matrix.
- Encrypt each of these bits to obtain *clean* (*ie* with small noise) ciphertexts $c_{i,j}$.

- Multiply each row of this matrix by the corresponding encryption $\mathfrak{c}_i$ of the secret key bits to obtain a matrix $\{\mathfrak{c}_i \cdot c_{i,j} \pmod d\}$.
- Now we need to compute the encryption of the sum of the plaintext bits $\sigma_i \cdot b_{i,j}$ in each of the columns separately.
  - Labeling in reverse corresponding to lower bits (as above), for column $-j$ we compute the carry bit to be sent to column $-j+t$ as the elemmentary symmetric polynomial $\pmod 2$ of degree 2^t in the bits of column $-j$. This is just the $t'th$ bit of the hamming weight of that column.
  - Form a suitable matrix and use carry-save-adders (see ([4])) in [3] or "grade-school" addition in [2] to reduce it to a matrix with two rows.
- In the final stage we need to xor the two remaining encrypted bits to obtain the clean encryption of the original message.

This performs the function required even if it does seem a little cumbersome. In particular bootstrapping is possible if we can evaluate elementary symmetric polynomial up to a certain degree in in [3] and Gentry's original scheme or if we can use the "school-book" addition method as found in [2].

Recall in [1] for the RLWE variant, ciphertexts are vectors of elements of a ring $R_q = \mathbb{Z}_q[x]/(f)$. After key switching a ciphertext will be of the form (c_0, c_1) and decryption can be computed as $[c_0 - s \cdot c_1]_q \pmod 2$. Let $s = \Sigma s_i X^i, c_0 = \Sigma u_i X^i$ and $c_1 = \Sigma v_i X^i$. Then the $i'th$ coefficient of $c_o - s \cdot c_1$ is given as $\Sigma s_j \cdot w_j + w_{-1}$ where w_{-1} is the additional term appearing due to reduction by the field polynomial. In this case we then represent the bits of each w_j/d in a matrix and apply the same method as above to bootstrap (we assume the coefficients of the secret key are in $\{0, 1\}$ for simplicity - other keys can easily be dealt with). Note in particular the abscense of any sparse subset sum problems.

References

[1] Z. Brakerski, C. Gentry and V. Vaikuntanathan Fully Homomorphic Encryption without Bootstrapping To appear in Innovations in Theoretical Computer Science 2012.

[2] C. Gentry and S. Halevi. Implementing Gentrys Fully-Homomorphic Encryption Scheme. In *EUROCRYPT 2011*, volume 6632 of Lecture Notes in Computer Science, pages 129148. Springer, 2011.

[3] N.P. Smart and F. Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In *Public Key Cryptography – PKC 2010*, Springer LNCS 6056, 420–443, 2010

[4] `http://en.wikipedia.org/wiki/Carry-save_adder`