In this lecture we review Gentry’s somewhat homomorphic encryption (SWHE) scheme. In Gentry’s scheme, the plaintext space and the ciphertext space are rings (support addition and multiplication), and given encryptions of $\ell$ messages, $c_1, \ldots, c_\ell$, where $c_i \leftarrow \text{Enc}(m_i)$, and a polynomial $Q$ of bounded degree (and not-too-many terms), we have (except for negligible probability)

$$Q(m_1, \ldots, m_\ell) = \text{Dec}(Q(c_1, \ldots, c_\ell)).$$

1 Background: GGH-type Cryptosystems

We briefly recall Micciancio’s “cleaned-up version” of GGH cryptosystems [GGH97, Mic01]. The secret and public keys are “good” and “bad” bases of some lattice $\Lambda$. More specifically, the key-holder generates a good basis by choosing $B_{\text{sk}}$ to be a basis of short, “nearly orthogonal” vectors. Then it sets the public key to be the Hermite normal form of the same lattice, $B_{\text{pk}} \overset{\text{def}}{=} \text{HNF}(\Lambda(B_{\text{sk}}))$.

A ciphertext in a GGH-type cryptosystem is a vector $\vec{c}$ close to the lattice $\Lambda(B_{\text{pk}})$, and the message which is encrypted in this ciphertext is somehow encoded in the distance from $\vec{c}$ to the nearest lattice vector. To encrypt a message $m$, the sender chooses a short “error vector” $\vec{e}$ that encodes $m$, and then computes the ciphertext as $\vec{c} \leftarrow \vec{c} \mod B_{\text{pk}}$. Note that if $\vec{c}$ is short enough (i.e., less than $\lambda_1(\Lambda)/2$), then it is indeed the distance between $\vec{c}$ and the nearest lattice point.

To decrypt, the key-holder uses its “good” basis $B_{\text{sk}}$ to recover $\vec{e}$ by setting $\vec{c} \leftarrow \vec{c} \mod B_{\text{sk}}$, and then recovers $m$ from $\vec{e}$. The reason decryption works is that, if the parameters are chosen correctly, then the parallelepiped $\mathcal{P}(B_{\text{sk}})$ of the secret key will be a “plump” parallelepiped that contains a sphere of radius bigger than $\|\vec{c}\|$, so that $\vec{c}$ is indeed the unique point inside $\mathcal{P}(B_{\text{sk}})$ that equals $\vec{c}$ modulo $\Lambda$. On the other hand, the parallelepiped $\mathcal{P}(B_{\text{pk}})$ of the public key will be very skewed, and will not contain a sphere of large radius, making it useless for solving BDDP.

More algebraically, the secret-key basis $B_{\text{sk}}$ is chosen so that all the columns of $B_{\text{sk}}^{-1}$ have Euclidean length smaller than $1/2\|\vec{e}\|$. Recall that $\vec{c} = \vec{v} + \vec{e}$ for some $\vec{v} \in \Lambda$, so we can write $\vec{c} = \vec{\alpha}B_{\text{sk}} + \vec{e}$ for some integer coefficient vector $\vec{\alpha}$. Also, reducing $\vec{c} \mod B_{\text{sk}}$ is done by computing

$$\boxed{\text{Modulo to nearest integer}}$$

$$\vec{c} \mod B_{\text{sk}} = [(\vec{c}B_{\text{sk}}^{-1}) \bmod B_{\text{sk}}] B_{\text{sk}} = [(\vec{\alpha}B_{\text{sk}} + \vec{e})B_{\text{sk}}^{-1}] B_{\text{sk}} \overset{\text{(*)&}}{=} [\vec{\alpha} + \vec{e}B_{\text{sk}}^{-1}] B_{\text{sk}} = \vec{\alpha}B_{\text{sk}} + \vec{e} \mod B_{\text{sk}}$$

where Equality $(\ast)$ follows since $\vec{\alpha}$ is an integer vector and $[\cdot]$ means taking only the fractional part. Each entry of $\vec{e}B_{\text{sk}}^{-1}$ is the inner product of $\vec{e}$ with a column of $B_{\text{sk}}^{-1}$, and as the column is shorter than $1/2\|\vec{e}\|$ then that entry is smaller than $1/2$ in absolute value. It follows that the fractional part $[\vec{e}B_{\text{sk}}^{-1}]$ equals $\vec{e}B_{\text{sk}}^{-1}$ exactly. Thus,

$$\vec{c} \mod B_{\text{sk}} = [\vec{e}B_{\text{sk}}^{-1}] B_{\text{sk}} = \vec{e}B_{\text{sk}}^{-1}B_{\text{sk}} = \vec{e}.$$

Note that if the encoding of $m$ into $\vec{c}$ is linear, then this scheme is already “somewhat” additively homomorphic, since for two ciphertexts $\vec{c}_1 = \vec{v}_1 + \vec{e}_1$ and $\vec{c}_2 = \vec{v}_2 + \vec{e}_2$, we get that $\vec{c} = \vec{e}_1 + \vec{e}_2$ encodes $m_1 + m_2$. If $\vec{c}$ is still short enough then decryption will recover it and thus returns $m_1 + m_2$. 

For example, if in order to encode \( m \in \{0, 1\} \) we denote \( \vec{m} = (m, 0, \ldots, 0) \in \{0, 1\}^n \), choose a short integer vector \( \vec{r} \) and set \( \vec{e} = 2\vec{r} + \vec{m} \), then

\[
\vec{c}_1 + \vec{c}_2 = (\vec{v}_1 + 2\vec{r}_1 + \vec{m}_1) + (\vec{v}_2 + 2\vec{r}_2 + \vec{m}_2) = (\vec{v}_1 + \vec{v}_2) + 2(\vec{r}_1 + \vec{r}_2) + (\vec{m}_1 + \vec{m}_2) = \vec{v} + \vec{e},
\]

where \( \vec{v} = \vec{v}_1 + \vec{v}_2 \in \Lambda \), and \( \vec{e} \equiv (m_1 \oplus m_2, 0, \ldots, 0) \mod 2 \). If \( \vec{e} \) is short then we decrypt \( m_1 \oplus m_2 \).

Recall that a lattice is a discrete additive subgroup of \( \mathbb{Z}^n \). In order to obtain an encryption scheme that is (somewhat) homomorphic w.r.t. multiplication we need a ring structure as we have in ideal lattices. Consider the encryption scheme from the “GGH example” above, where \( \Lambda = \Lambda_J \) is an ideal lattice with the underlying ring \( R_n = \mathbb{Z}[x]/(x^n + 1) \), then we have

\[
\vec{c}_1 \times \vec{c}_2 = (\vec{v}_1 + 2\vec{r}_1 + \vec{m}_1) \times (\vec{v}_2 + 2\vec{r}_2 + \vec{m}_2) = \frac{\vec{v}_1 \times (2\vec{r}_2 + \vec{m}_2) + \vec{v}_2 \times (2\vec{r}_1 + \vec{m}_1) + 2(\vec{r}_1 \times \vec{r}_2 + \vec{r}_1 \times \vec{m}_2 + \vec{m}_1 \times \vec{r}_2) + \vec{m}_1 \times \vec{m}_2}{\vec{e}}
\]

where \( \vec{v} \in \Lambda_J \) since \( \vec{v}_1, \vec{v}_2 \in \Lambda_J \) and \( J \) is an ideal. Note that if \( \vec{m}_i = (m_i, 0, \ldots, 0) \), with the leftmost entry being the free term in the corresponding polynomial, then we have \( \vec{m}_1 \times \vec{m}_2 = (m_1m_2, 0, \ldots, 0) \). If \( \vec{e} \) is still small enough then we can recover it by \( \vec{m}_1 \times \vec{m}_2 \equiv \vec{e} \mod 2 \).

## 2 Gentry’s Somewhat-Homomorphic Encryption (SWHE) Scheme

The SWHE scheme that underlies Gentry’s scheme is a GGH-type cryptosystem where the public key specifies an ideal lattice \( \Lambda_J \). Here we only cover a special case of Gentry’s scheme where all the ideals are principal and the ring that is used for polynomial arithmetic is \( R_n = \mathbb{Z}[x]/(x^n + 1) \), with \( n \) a power of two. (This is the variant that was implemented in [SV10] and [GH11].)

The relation in the ring \( R_n \) is \( x^n \equiv -1 \), hence \( R_n \) is closed under “rotation-negation”, i.e. if

\[
\vec{v} = (v_0, \ldots, v_{n-1}) = v_0 + v_1x + \ldots + v_{n-1}x^{n-1} \in R_n,
\]

then so is

\[
x\vec{v} = x \times \sum_{i=0}^{n-1} v_ix^i = -v_{n-1} + v_0x + v_1x^2 + \ldots + v_{n-2}x^{n-1} = (-v_{n-1}, v_0, \ldots, v_{n-2}).
\]

Therefore, given \( \vec{v} = (v_0, \ldots, v_{n-1}) \in R_n \), we can define the rotation basis of \( \vec{v} \) as

\[
V = \begin{pmatrix}
\vec{v} \\
x\vec{v} \\
\vdots \\
x^{n-1}\vec{v}
\end{pmatrix} = \begin{pmatrix}
v_0 & v_1 & \cdots & v_{n-1} \\
-v_{n-1} & v_0 & \cdots & v_{n-2} \\
\vdots & \vdots & \ddots & \vdots \\
-v_1 & -v_2 & \cdots & v_0
\end{pmatrix}.
\]

**Parameters:** The security parameter is \( n = 2^m \), in addition we have 3 other size parameters \( \rho, \sigma, \tau \) that satisfy \( \tau \geq \sigma n \log n \) and \( \tau > (pn \log n)^{4\sqrt{\pi}}. \) For example one can set \( \sigma = n \), and then determine \( \rho, \tau \).

**Key Gen:** Choose \( \vec{s} \leftarrow \mathcal{N}(0, \sigma^2)^n \) and set \( \vec{v} = (\tau, 0, \ldots, 0) + \lceil \vec{s} \rceil \). Ensure that \( \det(V) \) is odd and that \( \| \lceil \vec{s} \rceil \| < \sigma n \log n \). The secret key is \( \vec{v} \) whereas the public key is \( B = \text{HNF}(\vec{v}) \), the HNF basis for the lattice spanned by the rows of \( V \) (corresponding to the ideal \( \langle \vec{v} \rangle \).)

**Encrypt** \( B(m) \): Given \( m \in \{0, 1\} \) choose at random \( \vec{r} \leftarrow \mathcal{N}(0, \rho^2)^n \), and set

\[
\vec{c} = 2\lceil \vec{r} \rceil + (m, 0, \ldots, 0) \mod B.
\]

2
Decrypt_\ell(\vec{c}): Let V be the rotation basis of \vec{v}, compute \vec{m} = (\vec{c} \mod V) \mod 2, and output the first entry, i.e. if W = V^{-1}, then \vec{m} = ([cW]V) \mod 2 (where [\cdot] is the fractional part in the range \([-\frac{1}{2}, \frac{1}{2}])

As in the GGH scheme, in order for the decryption to work we require that \|cW\|_{\infty} < \frac{1}{2}, so that we have [cW]V = cWV = \vec{c}.

**Claim 1.** Let \vec{e} \in \mathbb{R}^n such that \|\vec{e}\|_{\infty} < \frac{\tau}{4}, then \|cW\|_{\infty} < \frac{1}{2}.

**Proof.** Since every entry of \vec{c}W is an inner product of \vec{e} with a column of W, it is enough to show that every column of W is small enough.

Assume that \|cW\|_{\infty} \geq \frac{1}{2}, and we will show that w.h.p. \|\vec{e}\|_{\infty} \geq \frac{\tau}{4}. Let \vec{t} = cW = \vec{e}V^{-1}, i.e. \vec{e} = iV = \sum_j t_j (x^j \vec{v}). Let i be the largest such that |t_i| \geq \frac{1}{2}. In the key generation procedure we set \vec{v} = (\tau, 0, \ldots, 0) + [\vec{s}], therefore x^j \vec{v} = (0, \ldots, 0, \tau, 0, \ldots, 0) + [x^j \vec{s}], and the i^{th} entry of \vec{e} is

\[
e_i = t_i \tau + \sum_{j=0}^{i} t_j [s_{i-j}] - \sum_{j=0}^{n-1-(i+1)} t_{j+i+1} [s_{n-1-j}].
\]

It follows that

\[
|e_i| = |t_i \tau + \sum_{j=0}^{i} t_j [s_{i-j}] - \sum_{j=0}^{n-1-(i+1)} t_{j+i+1} [s_{n-1-j}]| \\
\geq |t_i \tau - \sum_{j=0}^{i} t_j [s_{i-j}] - \sum_{j=0}^{n-1-(i+1)} t_{j+i+1} [s_{n-1-j}]| \\
\geq |t_i \tau - \sum_{j=0}^{i} t_j [s_{i-j}] - \sum_{j=0}^{n-1-(i+1)} t_1 [s_{n-1-j}]| \\
= |t_i||\tau - \sum_{j=0}^{n-1} [s_{j}]|| \\
= |t_i||\tau - \|\vec{s}\|_{1}|.
\]

However, since |t_i| \geq \frac{1}{2}, \|\vec{s}\|_{1} < \sigma n \log n and \tau \geq \sigma n \log n we get

\[
|e_i| \geq \frac{1}{2} |\tau - \sigma n \log n| \geq \frac{\tau}{4}.
\]

It follows that \|\vec{e}\|_{\infty} \geq \frac{\tau}{4}, and we get a contradiction.

The following claim explains the somewhat homomorphic nature of the encryption scheme.

**Claim 2.** Let \(Q(x_1, \ldots, x_\ell)\) be a binary polynomial of degree at most \(\sqrt{n}\) in each variable, with at most \(n^{2\sqrt{n}}\) terms. For \(i = 1, \ldots, \ell\) let \(m_i \in \{0, 1\}\) and set \(c_i' \leftarrow \text{Enc}_B(m_i)\). In addition, set \(\vec{c'} = Q(c_1', \ldots, c_\ell')\) (where evaluation is over \(R_n\)). Then w.h.p. \(\text{Dec}(\vec{c'}) = Q(m_1, \ldots, m_\ell) \mod 2\).

**Proof.** With high probability each one of the \(c_i'\) is of the form \(c_i' = \vec{u}_i + \vec{e}_i\), for some \(\vec{u}_i \in \langle \vec{v} \rangle\), with \(\|\vec{e}_i\|_{\infty} < \rho \log n\) and \(c_i' \equiv (m_i, 0, \ldots, 0) \mod 2\). It follows that \(Q(c_1', \ldots, c_\ell') = \vec{u} + Q(\vec{e}_1', \ldots, \vec{e}_\ell')\) for some \(\vec{u} \in \langle \vec{v} \rangle\) (because the \(\vec{u}_i\) are in the ideal). Similarly, since \(e_i = 2r_i + m_i\) we have \(Q(\vec{e}_1', \ldots, \vec{e}_\ell') = 2\vec{r} + Q(m_1, \ldots, m_\ell) \equiv Q(\vec{m}_1, \ldots, \vec{m}_\ell) \mod 2\).

Note that for \(\vec{a}, \vec{b} \in R_n\) we have \(\|\vec{a} \times \vec{b}\|_{\infty} \leq n \cdot \|\vec{a}\|_{\infty} \cdot \|\vec{b}\|_{\infty}\), hence

\[
\|\vec{e}\|_{\infty} = \|Q(\vec{e}_1', \ldots, \vec{e}_\ell')\|_{\infty} \leq (\max_i \|\vec{e}_i\|_{\infty})^{\sqrt{n}} \cdot n^{\sqrt{n} - 1} \cdot (\# \text{ of terms}) \leq (pn \log n)^{\sqrt{n}} n^{2\sqrt{n}} \leq (pn \log n)^{4\sqrt{n}} \ll \tau/4.
\]

So by Claim 1 decryption will recover \(\vec{e} = Q(\vec{e}_1', \ldots, \vec{e}_\ell')\), and therefore also \(Q(\vec{m}_1, \ldots, \vec{m}_\ell)\).
3 Security of Gentry’s SWHE Scheme

Claim 3. The scheme is CPA-secure if for \( \vec{v} \leftarrow (\tau, 0, \ldots, 0) + [\mathcal{N}(0, \sigma^2)^n] \) it is hard to distinguish \([\mathcal{N}(0, \rho^2)^n] \mod B\) from a uniform integer vector \( \mod B\), where \( B \) is the HNF of the lattice \( \Lambda(\vec{v}) \), assuming \( \det(V) \) is odd.

Before we prove the claim we need to play a bit with some algebra. Let \( V \) be the rotation basis of \( \vec{v} \) and denote \( d = \det(V) \). We know that \( d \neq 0 \). Assume \( d \) is odd, an denote the adjoint matrix of \( V \) by \( A = dV^{-1} \), \( A \) is an integer matrix as it is the adjoint of an integer matrix. Let \( \vec{a} = (a_0, \ldots, a_{n-1}) \) be the first row of \( A \). On one hand, since \( AV = dI \) we have \( \vec{a}V = (d, 0, \ldots, 0) \), which is in fact the constant polynomial \( d \in R_n \). On the other hand we have

\[
\vec{a}V = \sum_{i=0}^{n-1} a_i(x^i\vec{v}) = \sum_{i=0}^{n-1} (a_ix^i) \times \vec{v} \mod (x^n + 1) = \vec{a} \times \vec{v} \in R_n.
\]

It follows that \( \vec{a} \times \vec{v} = d \) (the constant polynomial \( d \)). Note that \( x\vec{a} \times \vec{v} = x\vec{d} = (0, d, 0, \ldots, 0) \), hence the second row of \( A \) is \( x\vec{a} \). In fact \( A \) is the rotation basis of \( \langle \vec{a} \rangle \), and \( \vec{a} \) is the scaled inverse of \( \vec{v} \).

Now, since \( d \) is odd, \( \frac{d-1}{2} \in \mathbb{Z} \), and we can consider the constant polynomial \( \frac{d-1}{2} \in R_n \). It holds that

\[
\vec{a} \times \vec{v} - 2\frac{d-1}{2} = d - (d-1) = 1 \in R_n,
\]

namely the polynomials \( \vec{a} \) and \( 2 \) are coprime in \( R_n \). It follows that the map \( \vec{x} \mapsto 2\vec{x} \mod \langle \vec{a} \rangle \) is a permutation.

What do we actually mean by \( \vec{x} \mapsto 2\vec{x} \mod \langle \vec{a} \rangle \)? Since \( \langle \vec{a} \rangle \) is an ideal in \( R_n \), we can consider the quotient ring \( R_n/\langle \vec{a} \rangle \) and the natural projection \( R_n \rightarrow R_n/\langle \vec{a} \rangle \). Now \( \vec{x} \mod \langle \vec{a} \rangle \) is simply the image of this projection (by abuse of notation we write \( \vec{x} \mod \langle \vec{a} \rangle \)). We can look at the doubling map over \( R_n/\langle \vec{a} \rangle \), sending \( \vec{x} \in R_n/\langle \vec{a} \rangle \) to \( 2\vec{x} \in R_n/\langle \vec{a} \rangle \). Since \( 2 \) and \( \langle \vec{a} \rangle \) are coprime in \( R_n \), \( 2 \) has an inverse \( \frac{1}{d} \in R_n/\langle \vec{a} \rangle \). Thus doubling induces a permutation on \( R_n/\langle \vec{a} \rangle \):

\[
2\vec{x} \times \frac{1-d}{2} = \vec{x} \times (1 - \vec{a} \times \vec{v}) = \vec{x} \mod \langle \vec{a} \rangle.
\]

Two polynomials \( \vec{a}, \vec{b} \in R_n \) are congruent \( \mod \langle \vec{a} \rangle \) if \( \vec{a} \equiv \vec{b} \mod \langle \vec{a} \rangle \), i.e. there is some \( \vec{u} \in R_n \) such that \( \vec{a} = \vec{b} + \vec{u}V \), however \( \vec{u}V = \vec{u}V \), hence \( \vec{a}, \vec{b} \) are congruent \( \mod \langle \vec{a} \rangle \) iff \( \vec{a}, \vec{b} \) are congruent \( \mod B \), and we can conclude that the mapping \( \vec{x} \mapsto 2\vec{x} \mod V \) is a permutation on \( R_n/\langle \vec{a} \rangle \). We are now ready to prove claim 3.

Proof of Claim 3. Let \( \mathcal{A} \) be a CPA adversary with advantage \( \epsilon \). We will show how to utilize it and construct a distinguisher between \( ([\mathcal{N}(0, \rho^2)^n] \mod B) \) from a uniform integer vector in \( \mathcal{P}(B) \), where \( \vec{v} \) is chosen as in the key generation algorithm of the scheme and \( B \) is the HNF basis of \( \langle \vec{a} \rangle \).

Given \( B \) and \( \vec{x} \), we need to decide if \( \vec{x} \) is uniform \( \mod B \) or Gaussian \( \mod B \). We give \( \mathcal{A} \) the basis \( B \) as public key, and \( \mathcal{A} \) gives us two bits \( m_0, m_1 \). We choose a random bit \( b \in \{0, 1\} \), and give \( \mathcal{A} \) the ciphertext \( \vec{c} = 2\vec{x} + (m_b, 0, \ldots, 0) \mod B \). When \( \mathcal{A} \) returns a bit \( b' \) we output 1 if \( b = b' \) and 0 otherwise.

If \( \vec{x} \) is Gaussian then this is a perfect simulation of the scheme, hence \( \mathcal{A} \) guesses correctly with probability \( \frac{1}{2} + \epsilon \).

If \( \vec{x} \) is uniform \( \mod B \) then \( 2\vec{x} \mod B = (2\vec{x} \mod V) \mod B \), and since \( \vec{x} \mod V \) is uniform in \( \mathcal{P}(V) \) and doubling is a permutation, then \( 2\vec{x} \mod V \) is also uniform in \( \mathcal{P}(V) \), hence \( 2\vec{x} \mod B \) is uniform in \( \mathcal{P}(B) \). It follows that \( 2\vec{x} + m_b \mod B \) is uniform in \( \mathcal{P}(B) \) regardless of \( b \). Therefore \( \mathcal{A} \) guesses correctly in this case with probability \( \frac{1}{2} - \epsilon \). \( \square \)
So how hard is it to distinguish between uniform and Gaussian mod $B$? We don’t really know, however one way is to solve the BDD problem for the Gaussian case. Note that when $\vec{x}$ is Gaussian then w.h.p. $\|\vec{x}\| \sim \rho$, whereas

$$\det(\Lambda(V)) \leq \prod_{i=0}^{n-1} \|x^i v\| \leq (\tau + \sigma \log n)^n < (2\tau)^n.$$ 

It follows that the ratio between the error distance $(\vec{c}, \Lambda)$ and $\sqrt[n]{\det(\Lambda)}$ is

$$\frac{\sqrt[n]{\det(\Lambda)}}{\rho} < \frac{2\tau}{\rho} < 2^4 \sqrt{n},$$

and we do not know how to solve BDD with this ratio.

References


