Fully Homomorphic Encryption
Part II

Mehdi Tibouchi

NTT Secure Platform Laboratories

EPIT 2013, 2013–03–22
Outline

Breaking things with lattices
  Yesterday’s game
  Howgrave-Graham on approximate GCDs

Fully homomorphic encryption from LWE
  Recap on LWE
  A secret key homomorphic scheme
  Achieving homomorphic multiplication
  Obtaining fully homomorphic encryption
The scheme we wanted to break

- Recall yesterday’s “improved” variant of vDGHV:
  - Only two public key elements \( x_0 = q_0 \cdot p, \ x_1 = q_1 \cdot p + 2r_1 \).
  - Encrypt \( m \) as \( c = m + 2r'_0 + r'_1x_1 \mod x_0 \) for small random \( r'_0, r'_1 \).
  - Decrypt \( c \) as \( m = (c \mod p) \mod 2 \).

In particular, all ciphertexts are of the form:

\[
c = (m + 2r'_0) + A \cdot x_0 + B \cdot x_1
\]

were \( A, B, C \) are small unknown integers (less than \( \rho \) bits, say), and \( x_0, x_1 \) are very large public constants (\( \gamma \) bit long, with \( \gamma \gg \rho \)).

- This is a weighted knapsack problem: we should be able to recover the coefficients of \( x_0, x_1 \) and 1 with lattice reduction!
The scheme we wanted to break

- Recall yesterday’s “improved” variant of vDGHV:
  - Only two public key elements $x_0 = q_0 \cdot p$, $x_1 = q_1 \cdot p + 2r_1$.
  - Encrypt $m$ as $c = m + 2r'_0 + r'_1 x_1 \mod x_0$ for small random $r'_0$, $r'_1$.
  - Decrypt $c$ as $m = (c \mod p) \mod 2$.

- In particular, all ciphertexts are of the form:

$$c = (m + 2r'_0) + A \cdot x_0 + B \cdot x_1$$

were $A$, $B$, $C$ are small unknown integers (less than $\rho$ bits, say), and $x_0$, $x_1$ are very large public constants ($\gamma$ bit long, with $\gamma \gg \rho$).

- This is a weighted knapsack problem: we should be able to recover the coefficients of $x_0$, $x_1$ and 1 with lattice reduction!
The scheme we wanted to break

- Recall yesterday’s “improved” variant of vDGHV:
  - Only two public key elements $x_0 = q_0 \cdot p$, $x_1 = q_1 \cdot p + 2r_1$.
  - Encrypt $m$ as $c = m + 2r'_0 + r'_1 x_1 \mod x_0$ for small random $r'_0$, $r'_1$.
  - Decrypt $c$ as $m = (c \mod p) \mod 2$.
- In particular, all ciphertexts are of the form:

$$c = (m + 2r'_0) + A \cdot x_0 + B \cdot x_1$$

were $A$, $B$, $C$ are small unknown integers (less than $\rho$ bits, say), and $x_0$, $x_1$ are very large public constants ($\gamma$ bit long, with $\gamma \gg \rho$).
- This is a weighted knapsack problem: we should be able to recover the coefficients of $x_0$, $x_1$ and 1 with lattice reduction!
Knapsack lattice

- $c = (m + 2r_0') + A \cdot x_0 + B \cdot x_1$

- Consider the integer lattice $L$ generated by the rows of:

$$L = \begin{pmatrix}
1 & 0 & 0 & S \cdot 1 \\
0 & 1 & 0 & S \cdot x_0 \\
0 & 0 & 1 & S \cdot x_1 \\
0 & 0 & 0 & S \cdot c
\end{pmatrix}, \quad S \in \mathbb{N}^*$$

- $L$ clearly contains $v = (m + 2r_0', A, B, 0)$ of norm $\|v\| \leq \sqrt{3} \cdot 2^\rho$.

- Heuristic: this is considerably smaller than $(\det L)^{1/4}$ so LLL should find it... but not if $S$ is too small?

- Provable claim: pick $S$ large enough, say $S = 5 \cdot 2^\rho > 2^{(4-1)/2} \|v\|$. With overwhelming probability on the choice of public key elements $x_0, x_1$, $\|v\| = \lambda_1(L)$ and the first vector of any LLL-reduced basis is $\pm v$. 
Knapsack lattice

- \( c = (m + 2r_0') + A \cdot x_0 + B \cdot x_1 \)
- Consider the integer lattice \( L \) generated by the rows of:

\[
L = \begin{pmatrix}
1 & 0 & 0 & S \cdot 1 \\
0 & 1 & 0 & S \cdot x_0 \\
0 & 0 & 1 & S \cdot x_1 \\
0 & 0 & 0 & S \cdot c
\end{pmatrix}, \quad S \in \mathbb{N}^*
\]

- \( L \) clearly contains \( v = (m + 2r_0', A, B, 0) \) of norm \( \|v\| \leq \sqrt{3} \cdot 2^\rho \).
- Heuristic: this is considerably smaller than \((\det L)^{1/4}\) so LLL should find it... but not if \( S \) is too small?
- Provable claim: pick \( S \) large enough, say \( S = 5 \cdot 2^\rho > 2^{(4-1)/2} \|v\| \). With overwhelming probability on the choice of public key elements \( x_0, x_1 \), \( \|v\| = \lambda_1(L) \) and the first vector of any LLL-reduced basis is \( \pm v \).
Knapsack lattice

- \( c = (m + 2r'_0) + A \cdot x_0 + B \cdot x_1 \)
- Consider the integer lattice \( L \) generated by the rows of:

\[
L = \begin{pmatrix}
1 & 0 & 0 & S \cdot 1 \\
0 & 1 & 0 & S \cdot x_0 \\
0 & 0 & 1 & S \cdot x_1 \\
0 & 0 & 0 & S \cdot c
\end{pmatrix}, \quad S \in \mathbb{N}^*
\]

- \( L \) clearly contains \( \mathbf{v} = (m + 2r'_0, A, B, 0) \) of norm \( \| \mathbf{v} \| \leq \sqrt{3} \cdot 2^\rho \).
- Heuristic: this is considerably smaller than \( (\det L)^{1/4} \) so LLL should find it... but not if \( S \) is too small?
- Provable claim: pick \( S \) large enough, say \( S = 5 \cdot 2^\rho > 2^{(4-1)/2}\|\mathbf{v}\| \). With overwhelming probability on the choice of public key elements \( x_0, x_1 \), \( \| \mathbf{v} \| = \lambda_1(L) \) and the first vector of any LLL-reduced basis is \( \pm \mathbf{v} \).
Knapsack lattice

- $c = (m + 2r'_0) + A \cdot x_0 + B \cdot x_1$
- Consider the integer lattice $L$ generated by the rows of:

$$L = \begin{pmatrix}
1 & 0 & 0 & S \cdot 1 \\
0 & 1 & 0 & S \cdot x_0 \\
0 & 0 & 1 & S \cdot x_1 \\
0 & 0 & 0 & S \cdot c
\end{pmatrix}, \quad S \in \mathbb{N}^*$$

- $L$ clearly contains $v = (m + 2r'_0, A, B, 0)$ of norm $\|v\| \leq \sqrt{3} \cdot 2^\rho$.
- Heuristic: this is considerably smaller than $(\det L)^{1/4}$ so LLL should find it... but not if $S$ is too small?
- Provable claim: pick $S$ large enough, say $S = 5 \cdot 2^\rho > 2^{(4-1)/2}\|v\|$. With overwhelming probability on the choice of public key elements $x_0, x_1$, $\|v\| = \lambda_1(L)$ and the first vector of any LLL-reduced basis is $\pm v$. 

©2013 NTT Secure Platform Laboratories
Knapsack lattice

- \( c = (m + 2r_0') + A \cdot x_0 + B \cdot x_1 \)
- Consider the integer lattice \( L \) generated by the rows of:

\[
L = \begin{pmatrix}
1 & 0 & 0 & S \cdot 1 \\
0 & 1 & 0 & S \cdot x_0 \\
0 & 0 & 1 & S \cdot x_1 \\
0 & 0 & 0 & S \cdot c
\end{pmatrix}, \quad S \in \mathbb{N}^*
\]

- \( L \) clearly contains \( \mathbf{v} = (m + 2r_0', A, B, 0) \) of norm \( \|\mathbf{v}\| \leq \sqrt{3} \cdot 2^\rho \).
- Heuristic: this is considerably smaller than \( (\det L)^{1/4} \) so LLL should find it... but not if \( S \) is too small?
- Provable claim: pick \( S \) large enough, say \( S = 5 \cdot 2^\rho > 2^{(4-1)/2} \|\mathbf{v}\| \). With overwhelming probability on the choice of public key elements \( x_0, x_1, \|\mathbf{v}\| = \lambda_1(L) \) and the first vector of any LLL-reduced basis is \( \pm \mathbf{v} \).
Outline

Breaking things with lattices
   Yesterday’s game
   Howgrave-Graham on approximate GCDs

Fully homomorphic encryption from LWE
   Recap on LWE
   A secret key homomorphic scheme
   Achieving homomorphic multiplication
   Obtaining fully homomorphic encryption
Lattices against PACD

- The security of (the compact variant of) vDGHV is based on the partial approximate GCD problem:
  - Given $x_0 = q_0 \cdot p$, $x_1 = q_0 \cdot p + r$, find $p$.
  - Depends on the sizes $\gamma \gg \eta \gg \rho$ of $x_i, p, r$.
  - More samples $x_i$ possible...

- Howgrave-Graham [H01] proposes of Coppersmith-like approach to solving the problem for some parameter sets.
  - Polynomials of the form
    \[
    Q_{i,j}(X) = x_0^{u-i} \cdot (X + x_1)^i X^j
    \]
    satisfy $p^u | Q_{ij}(-r)$.
  - Suppose we can find a linear combination $Q$ of these with small coefficients, such that $|Q|(|r|) < p^u$. Then $-r$ is a root of $Q$ in $\mathbb{Z}$ so we can find it and recover $p$. 
Lattices against PACD

- The security of (the compact variant of) vDGHV is based on the partial approximate GCD problem:
  - Given \( x_0 = q_0 \cdot p, \quad x_1 = q_0 \cdot p + r \), find \( p \).
  - Depends on the sizes \( \gamma \gg \eta \gg \rho \) of \( x_i, p, r \).
  - More samples \( x_i \) possible...

- Howgrave-Graham [H01] proposes of Coppersmith-like approach to solving the problem for some parameter sets.
  - Polynomials of the form
    \[
    Q_{i,j}(X) = x_0^{u-j} \cdot (X + x_1)^i X^j
    \]
    satisfy \( p^u \mid Q_{ij}(-r) \).
  - Suppose we can find a linear combination \( Q \) of these with small coefficients, such that \( |Q|(|r|) < p^u \). Then \(-r\) is a root of \( Q \) in \( \mathbb{Z} \) so we can find it and recover \( p \).
Howgrave-Graham’s condition (1)

- Good subfamily of \( Q_{i,j}(X) = x_0^{u-i} \cdot (X + x_1)^i X^j \)?
- Right polynomial family to consider: \( P_k = Q_{k,0} \) for \( k \leq u \) and \( P_k = Q_{u,k-u} \) for \( u < k \leq h \).
- For \( u = 2, h = 4 \), this gives the Coppersmith lattice:

\[
L = \begin{pmatrix}
  x_0^2 & 0 & 0 & 0 & 0 & 0 \\
  x_0 x_1 & x_0 B & 0 & 0 & 0 & 0 \\
  x_1^2 & 2x_1 B & B^2 & 0 & 0 & 0 \\
  0 & x_1^2 B & 2x_1 B^2 & B^3 & 0 & 0 \\
  0 & 0 & x_1^2 B^2 & 2x_1 B^3 & B^4 & 0 \\
  0 & 0 & 0 & 0 & 0 & 0 \\
\end{pmatrix}, \quad B = 2^\rho
\]

- Thus we get, for general \( u, h \):

\[
\det L = x_0^{u(u+1)/2} B^{h(h+1)/2} \approx 2^{\gamma u(u+1)/2 + \rho h(h+1)/2}
\]

and we expect to find short vectors in \( L \) of length

\[
\approx (\det L)^{1/(h+1)}.
\]
Howgrave-Graham’s condition (1)

- Good subfamily of $Q_{i,j}(X) = x_0^{u-i} \cdot (X + x_1)^i X^j$?
- Right polynomial family to consider: $P_k = Q_{k,0}$ for $k \leq u$ and $P_k = Q_{u,k-u}$ for $u < k \leq h$.
- For $u = 2$, $h = 4$, this gives the Coppersmith lattice:

$$L = \begin{pmatrix}
x_0^2 & 0 & 0 & 0 & 0 \\
x_0 x_1 & x_0 B & 0 & 0 & 0 \\
x_1^2 & 2x_1 B & B^2 & 0 & 0 \\
0 & x_1^2 B & 2x_1 B^2 & B^3 & 0 \\
0 & 0 & x_1^2 B^2 & 2x_1 B^3 & B^4
\end{pmatrix}, \quad B = 2^\rho$$

- Thus we get, for general $u, h$:

$$\det L = x_0^{u(u+1)/2} B^{h(h+1)/2} \approx 2^{\gamma u(u+1)/2 + \rho h(h+1)/2}$$

and we expect to find short vectors in $L$ of length

$$\approx (\det L)^{1/(h+1)}.$$
Good subfamily of \( Q_{i,j}(X) = x_0^{u-i} \cdot (X + x_1)^i X^j \)?

Right polynomial family to consider: \( P_k = Q_{k,0} \) for \( k \leq u \) and \( P_k = Q_{u,k-u} \) for \( u < k \leq h \).

For \( u = 2 \), \( h = 4 \), this gives the Coppersmith lattice:

\[
L = \begin{pmatrix}
  x_0^2 & 0 & 0 & 0 & 0 \\
x_0 x_1 & x_0 B & 0 & 0 & 0 \\
x_1^2 & 2x_1 B & B^2 & 0 & 0 \\
0 & x_1^2 B & 2x_1 B^2 & B^3 & 0 \\
0 & 0 & x_1^2 B^2 & 2x_1 B^3 & B^4
\end{pmatrix}, \quad B = 2^\rho
\]

Thus we get, for general \( u, h \):

\[
\det L = x_0^{u(u+1)/2} B^{h(h+1)/2} \approx 2^{\gamma u(u+1)/2 + \rho h(h+1)/2}
\]

and we expect to find short vectors in \( L \) of length

\[
\approx (\det L)^{1/(h+1)}.
\]
Howgrave-Graham’s condition (1)

- Good subfamily of $Q_{i,j}(X) = x_0^{u-i} \cdot (X + x_1)^i X^j$?
- Right polynomial family to consider: $P_k = Q_{k,0}$ for $k \leq u$ and $P_k = Q_{u,k-u}$ for $u < k \leq h$.
- For $u = 2$, $h = 4$, this gives the Coppersmith lattice:

$$L = \begin{pmatrix}
    x_0^2 & 0 & 0 & 0 & 0 \\
    x_0 x_1 & x_0 B & 0 & 0 & 0 \\
    x_1^2 & 2x_1 B & B^2 & 0 & 0 \\
    0 & x_1^2 B & 2x_1 B^2 & B^3 & 0 \\
    0 & 0 & x_1^2 B^2 & 2x_1 B^3 & B^4
\end{pmatrix}, \quad B = 2^\rho$$

- Thus we get, for general $u$, $h$:

$$\det L = x_0^{u(u+1)/2} B^{h(h+1)/2} \approx 2^\gamma u(u+1)/2 + \rho h(h+1)/2$$

and we expect to find short vectors in $L$ of length

$\approx (\det L)^{1/(h+1)}$. 
Howgrave-Graham’s condition (2)

- Norms of the short vectors we expect to find:

\[(\det L)^{1/(h+1)} \approx 2^{\gamma \frac{u(u+1)}{2(h+1)} + \rho \frac{h}{2}}\]

- We were looking for vectors of length \(p^u \approx 2^\eta u\).
- Hence the condition to the attack to work:

\[\frac{u + 1}{h + 1} \gamma + \frac{h}{u} \rho \lesssim 2\eta\]

- LHS minimal for \(u/h \sim \sqrt{\rho/\gamma}\), which gives the asymptotic condition \(\rho \gamma \lesssim \eta^2\).
- Thus, pick \(\rho \gamma\) much larger than \(\eta^2\) to thwart the attack.
- Generalization to many samples by Cohn and Heninger; proposed parameters remain safe, however.
Howgrave-Graham’s condition (2)

- Norms of the short vectors we expect to find:

\[
(\det L)^{1/(h+1)} \approx 2^{\gamma \frac{u(u+1)}{2(h+1)} + \rho \frac{h}{2}}
\]

- We were looking for vectors of length \( \lesssim p^u \approx 2^{\eta u} \).

- Hence the condition to the attack to work:

\[
\frac{u + 1}{h + 1} \gamma + \frac{h}{u} \rho \lesssim 2\eta
\]

- LHS minimal for \( u/h \sim \sqrt{\rho/\gamma} \), which gives the asymptotic condition \( \rho \gamma \lesssim \eta^2 \).

- Thus, pick \( \rho \gamma \) much larger than \( \eta^2 \) to thwart the attack.

- Generalization to many samples by Cohn and Heninger; proposed parameters remain safe, however.
Howgrave-Graham’s condition (2)

- Norms of the short vectors we expect to find:

\[(\det L)^{1/(h+1)} \approx 2^{\gamma \frac{u(u+1)}{2(h+1)} + \rho \frac{h}{2}}\]

- We were looking for vectors of length \(p^u \approx 2^{\eta u}\).
- Hence the condition to the attack to work:

\[\frac{u + 1}{h + 1} \gamma + \frac{h}{u} \rho \leq 2\eta\]

- LHS minimal for \(u/h \approx \sqrt{\rho/\gamma}\), which gives the asymptotic condition \(\rho \gamma \lesssim \eta^2\).
- Thus, pick \(\rho \gamma\) much larger than \(\eta^2\) to thwart the attack.
- Generalization to many samples by Cohn and Heninger; proposed parameters remain safe, however.
Howgrave-Graham’s condition (2)

- Norms of the short vectors we expect to find:

\[
\frac{(\det L)^{1/(h+1)}}{u^2(u+1)2^{(h+1)/2} + \rho h^2} = \frac{(h+1)^2}{u^2(u+1)2^{(h+1)/2} + \rho h^2}
\]

- We were looking for vectors of length \( \leq \rho^u \approx 2^{\eta u} \).

- Hence the condition to the attack to work:

\[
\frac{u+1}{h+1} + \frac{h}{u^2(u+1)2^{(h+1)/2} + \rho h^2} \leq 2\eta
\]

- LHS minimal for \( u/h \approx \sqrt{\rho/\gamma} \), which gives the asymptotic condition \( \rho \gamma \leq \eta^2 \).

- Thus, pick \( \rho \gamma \) much larger than \( \eta^2 \) to thwart the attack.

- Generalization to many samples by Cohn and Heninger; proposed parameters remain safe, however.
Howgrave-Graham’s condition (2)

- Norms of the short vectors we expect to find:
  \[
  (\det L)^{1/(h+1)} \approx 2^{\gamma \frac{u(u+1)}{2(h+1)} + \rho \frac{h}{2}}
  \]

- We were looking for vectors of length \( \leq \rho u \approx 2^{\eta u} \).
- Hence the condition to the attack to work:
  \[
  \frac{u + 1}{h + 1} \gamma + \frac{h}{u} \rho \leq 2\eta
  \]

- LHS minimal for \( u/h \leq \sqrt{\rho/\gamma} \), which gives the asymptotic condition \( \rho \gamma \leq \eta^2 \).
- Thus, pick \( \rho \gamma \) much larger than \( \eta^2 \) to thwart the attack.
- Generalization to many samples by Cohn and Heninger; proposed parameters remain safe, however.
Howgrave-Graham’s condition (2)

- Norms of the short vectors we expect to find:
  \[(\det L)^{1/(h+1)} \approx 2^{\gamma \frac{u(u+1)}{2(h+1)} + \rho \frac{h}{2}}\]

- We were looking for vectors of length \(\lesssim p^u \approx 2^{\eta u}\).

- Hence the condition to the attack to work:
  \[\frac{u + 1}{h + 1} \gamma + \frac{h}{u} \rho \lesssim 2\eta\]

- LHS minimal for \(u/h \sim \sqrt{\rho/\gamma}\), which gives the asymptotic condition \(\rho \gamma \lesssim \eta^2\).

- Thus, pick \(\rho \gamma\) much larger than \(\eta^2\) to thwart the attack.

- Generalization to many samples by Cohn and Heninger; proposed parameters remain safe, however.
Outline

Breaking things with lattices
  Yesterday’s game
  Howgrave-Graham on approximate GCDs

Fully homomorphic encryption from LWE
  Recap on LWE
  A secret key homomorphic scheme
  Achieving homomorphic multiplication
  Obtaining fully homomorphic encryption
The LWOE problem

- I have a secret vector $s \in \mathbb{Z}_q^n (q = \text{poly}(n))$.
- I give you access to an oracle that reveals the projection of $s$ along some random vector $a \in \mathbb{Z}_q^n$, i.e. outputs $(a, \langle s, a \rangle)$.
- Can you recover $s$ in polynomial time?
- Of course: after $O(n)$ queries, the queries gives vectors $a_1, \ldots, a_n$ forming a basis of $\mathbb{Z}_q^n$ and the corresponding projections, so recovering $s$ is simple linear algebra.
The LWOE problem

- I have a secret vector \( s \in \mathbb{Z}_q^n \) (\( q = \text{poly}(n) \)).
- I give you access to an oracle that reveals the projection of \( s \) along some random vector \( a \in \mathbb{Z}_q^n \), i.e. outputs \( (a, \langle s, a \rangle) \).
- Can you recover \( s \) in polynomial time?
- Of course: after \( O(n) \) queries, the queries gives vectors \( a_1, \ldots, a_n \) forming a basis of \( \mathbb{Z}_q^n \) and the corresponding projections, so recovering \( s \) is simple linear algebra.
The LWOE problem

- I have a secret vector \( \mathbf{s} \in \mathbb{Z}_q^n \) \((q = \text{poly}(n))\).
- I give you access to an oracle that reveals the projection of \( \mathbf{s} \) along some random vector \( \mathbf{a} \in \mathbb{Z}_q^n \), i.e. outputs \((\mathbf{a}, \langle \mathbf{s}, \mathbf{a} \rangle)\).
- Can you recover \( \mathbf{s} \) in polynomial time?
- Of course: after \( O(n) \) queries, the queries gives vectors \( \mathbf{a}_1, \ldots, \mathbf{a}_n \) forming a basis of \( \mathbb{Z}_q^n \) and the corresponding projections, so recovering \( \mathbf{s} \) is simple linear algebra.
The LWOE problem

- I have a secret vector $s \in \mathbb{Z}_q^n$ ($q = \text{poly}(n)$).
- I give you access to an oracle that reveals the projection of $s$ along some random vector $a \in \mathbb{Z}_q^n$, i.e. outputs $(a, \langle s, a \rangle)$.
- Can you recover $s$ in polynomial time?
- Of course: after $O(n)$ queries, the queries gives vectors $a_1, \ldots, a_n$ forming a basis of $\mathbb{Z}_q^n$ and the corresponding projections, so recovering $s$ is simple linear algebra.
The (search) LWE problem

- I have a secret vector $s \in \mathbb{Z}_q^n$ ($q = \text{poly}(n)$).
- I give you access to an oracle that reveals the projection of $s$ along some random vector $a \in \mathbb{Z}_q^n$ with some random small error $e$, i.e. outputs $(a, \langle s, a \rangle + e)$.
- Can you recover $s$ in polynomial time?
- Probably not: for an appropriate distribution of the noise values $e$, this is as hard as solving worst-case lattice problems (Regev, Peikert).
The (search) LWE problem

- I have a secret vector $s \in \mathbb{Z}_q^n$ ($q = \text{poly}(n)$).
- I give you access to an oracle that reveals the projection of $s$ along some random vector $a \in \mathbb{Z}_q^n$ with some random small error $e$, i.e. outputs $(a, \langle s, a \rangle + e)$.
- Can you recover $s$ in polynomial time?
- Probably not: for an appropriate distribution of the noise values $e$, this is as hard as solving worst-case lattice problems (Regev, Peikert).
The (search) LWE problem

- I have a secret vector $s \in \mathbb{Z}_q^n$ ($q = \text{poly}(n)$).
- I give you access to an oracle that reveals the projection of $s$ along some random vector $a \in \mathbb{Z}_q^n$ with some random small error $e$, i.e. outputs $(a, \langle s, a \rangle + e)$.
- Can you recover $s$ in polynomial time?
- Probably not: for an appropriate distribution of the noise values $e$, this is as hard as solving worst-case lattice problems (Regev, Peikert).
The (search) LWE problem

- I have a secret vector \( \mathbf{s} \in \mathbb{Z}_q^n \) (\( q = \text{poly}(n) \)).
- I give you access to an oracle that reveals the projection of \( \mathbf{s} \) along some random vector \( \mathbf{a} \in \mathbb{Z}_q^n \) with some random small error \( e \), i.e. outputs \((\mathbf{a}, \langle \mathbf{s}, \mathbf{a} \rangle + e)\).
- Can you recover \( \mathbf{s} \) in polynomial time?
- **Probably not:** for an appropriate distribution of the noise values \( e \), this is as hard as solving worst-case lattice problems (Regev, Peikert).
Previous slide: it is hard to find $s$ given polynomially many samples $(a, \langle s, a \rangle + e)$.

Equivalently, it is hard to find $s$ given a random matrix $A \in \mathbb{Z}_q^{n \times m}$ and the vector $s \cdot A + e$ for some random short vector $e \in \mathbb{Z}_q^m$.

Decision problem: distinguish between $(A, s \cdot A + e)$ and $(A, u)$, $u \in \mathbb{Z}_q^m$ uniformly random.

There is a search-to-decision reduction: the decision problem is as hard as the search version (as proved in Vadim’s talk).

Very convenient assumption to construct lattice-based schemes: encryption, (H)IBE, signatures, group signatures, oblivious transfer... and fully homomorphic encryption.
The decision LWE problem

- Previous slide: it is hard to find $s$ given polynomially many samples $(a, \langle s, a \rangle + e)$.
  - Equivalently, it is hard to find $s$ given a random matrix $A \in \mathbb{Z}_q^{n \times m}$ and the vector $s \cdot A + e$ for some random short vector $e \in \mathbb{Z}_q^m$.

- Decision problem: distinguish between $(A, s \cdot A + e)$ and $(A, u)$, $u \in \mathbb{Z}_q^m$ uniformly random.
  - There is a search-to-decision reduction: the decision problem is as hard as the search version (as proved in Vadim’s talk).
  - Very convenient assumption to construct lattice-based schemes: encryption, (H)IBE, signatures, group signatures, oblivious transfer... and fully homomorphic encryption.
The decision LWE problem

- Previous slide: it is hard to find $s$ given polynomially many samples $(a, \langle s, a \rangle + e)$.
  - Equivalently, it is hard to find $s$ given a random matrix $A \in \mathbb{Z}_q^{n \times m}$ and the vector $s \cdot A + e$ for some random short vector $e \in \mathbb{Z}_q^m$.

- Decision problem: distinguish between $(A, s \cdot A + e)$ and $(A, u)$, $u \in \mathbb{Z}_q^m$ uniformly random.

- There is a search-to-decision reduction: the decision problem is as hard as the search version (as proved in Vadim’s talk).

- Very convenient assumption to construct lattice-based schemes: encryption, (H)IBE, signatures, group signatures, oblivious transfer... and fully homomorphic encryption.
The decision LWE problem

- Previous slide: it is hard to find \( s \) given polynomially many samples \( (a, \langle s, a \rangle + e) \).
  - Equivalently, it is hard to find \( s \) given a random matrix \( A \in \mathbb{Z}_q^{n \times m} \) and the vector \( s \cdot A + e \) for some random short vector \( e \in \mathbb{Z}_q^m \).

- Decision problem: distinguish between \( (A, s \cdot A + e) \) and \( (A, u), u \in \mathbb{Z}_q^m \) uniformly random.

- There is a search-to-decision reduction: the decision problem is as hard as the search version (as proved in Vadim’s talk).

- Very convenient assumption to construct lattice-based schemes: encryption, (H)IBE, signatures, group signatures, oblivious transfer... and fully homomorphic encryption.
Drawbacks of yesterday’s schemes

- Both of the schemes presented yesterday (Gentry, vDGHV) suffer from a number of problems.
  - Security is not easy to obtain.
    - Gentry’s scheme: need to sample ideal lattices with both a really good basis (for correct decryption) and a really bad basis (for BDD to be hard).
    - vDGHV: hardness of approximate GCDs not well understood.
  - Noise grows very fast.
  - Squashing is difficult, messy, and requires additional assumption (hardness of sparse subset sums).
  - Bootstrapping is brilliant, but has high overhead and requires circular security.
  - LWE schemes by Brakerski et al. offer elegant solutions to most of these problems.
Drawbacks of yesterday’s schemes

- Both of the schemes presented yesterday (Gentry, vDGHV) suffer from a number of problems.
- Security is not easy to obtain.
  - Gentry’s scheme: need to sample ideal lattices with both a really good basis (for correct decryption) and a really bad basis (for BDD to be hard).
  - vDGHV: hardness of approximate GCDs not well understood.
- Noise grows very fast.
- Squashing is difficult, messy, and requires additional assumption (hardness of sparse subset sums).
- Bootstrapping is brilliant, but has high overhead and requires circular security.
- LWE schemes by Brakerski et al. offer elegant solutions to most of these problems.
Drawbacks of yesterday’s schemes

- Both of the schemes presented yesterday (Gentry, vDGHV) suffer from a number of problems.
- Security is not easy to obtain.
  - Gentry’s scheme: need to sample ideal lattices with both a really good basis (for correct decryption) and a really bad basis (for BDD to be hard).
  - vDGHV: hardness of approximate GCDs not well understood.
- Noise grows very fast.
  - Squashing is difficult, messy, and requires additional assumption (hardness of sparse subset sums).
  - Bootstrapping is brilliant, but has high overhead and requires circular security.
  - LWE schemes by Brakerski et al. offer elegant solutions to most of these problems.
Drawbacks of yesterday’s schemes

- Both of the schemes presented yesterday (Gentry, vDGHV) suffer from a number of problems.
- Security is not easy to obtain.
  - Gentry’s scheme: need to sample ideal lattices with both a really good basis (for correct decryption) and a really bad basis (for BDD to be hard).
  - vDGHV: hardness of approximate GCDs not well understood.
- Noise grows very fast.
- Squashing is difficult, messy, and requires additional assumption (hardness of sparse subset sums).
  - Bootstrapping is brilliant, but has high overhead and requires circular security.
  - LWE schemes by Brakerski et al. offer elegant solutions to most of these problems.
Drawbacks of yesterday’s schemes

- Both of the schemes presented yesterday (Gentry, vDGHV) suffer from a number of problems.
- Security is not easy to obtain.
  - Gentry’s scheme: need to sample ideal lattices with both a really good basis (for correct decryption) and a really bad basis (for BDD to be hard).
  - vDGHV: hardness of approximate GCDs not well understood.
- Noise grows very fast.
- Squashing is difficult, messy, and requires additional assumption (hardness of sparse subset sums).
- Bootstrapping is brilliant, but has high overhead and requires circular security.
- LWE schemes by Brakerski et al. offer elegant solutions to most of these problems.
Drawbacks of yesterday’s schemes

- Both of the schemes presented yesterday (Gentry, vDGHV) suffer from a number of problems.
- Security is not easy to obtain.
  - Gentry’s scheme: need to sample ideal lattices with both a really good basis (for correct decryption) and a really bad basis (for BDD to be hard).
  - vDGHV: hardness of approximate GCDs not well understood.
- Noise grows very fast.
- Squashing is difficult, messy, and requires additional assumption (hardness of sparse subset sums).
- Bootstrapping is brilliant, but has high overhead and requires circular security.
- LWE schemes by Brakerski et al. offer elegant solutions to most of these problems.
Outline

Breaking things with lattices
  Yesterday’s game
  Howgrave-Graham on approximate GCDs

Fully homomorphic encryption from LWE
  Recap on LWE
  A secret key homomorphic scheme
  Achieving homomorphic multiplication
  Obtaining fully homomorphic encryption
First imagine we’re trying to construct a secret-key homomorphic encryption scheme based on LWE.

Here’s a first attempt:

- Shared secret key: $sk = s = (1, -s_0) \in \mathbb{Z}_q^{n+1}$, where $s_0 \in \mathbb{Z}_q^n$ is uniformly random.
- $E_{sk}(b) = c = (\langle s_0, a \rangle + 2e + b, a)$ for uniformly random $a \in \mathbb{Z}_q^n$ and a small $e \in \mathbb{Z}_q$.
- $D_{sk}(c) = [\langle s, c \rangle]_q \mod 2$.

Clearly, under LWE, this is secure: $E_{sk}(0) \cong E_{sk}(1)$ since both are indistinguishable from a uniformly random vector in $\mathbb{Z}_q^{n+1}$.

Additively homomorphically (somewhat): $E_{sk}(b_1) + E_{sk}(b_2)$ decrypts to $b_1 \oplus b_2$.

How about multiplication? Encryptions are vectors, we cannot multiply them!
A secret key scheme

- First imagine we’re trying to construct a secret-key homomorphic encryption scheme based on LWE.
- Here’s a first attempt:
  - Shared secret key: $sk = s = (1, -s_0) \in \mathbb{Z}_{q+1}^{n}$, where $s_0 \in \mathbb{Z}_q^n$ is uniformly random.
  - $E_{sk}(b) = c = (\langle s_0, a \rangle + 2e + b, a)$ for uniformly random $a \in \mathbb{Z}_q^n$ and a small $e \in \mathbb{Z}_q$.
  - $D_{sk}(c) = [\langle s, c \rangle]_q \mod 2$.

- Clearly, under LWE, this is secure: $E_{sk}(0) \cong E_{sk}(1)$ since both are indistinguishable from a uniformly random vector in $\mathbb{Z}_q^{n+1}$.
- Additively homomorphically (somewhat): $E_{sk}(b_1) + E_{sk}(b_2)$ decrypts to $b_1 \oplus b_2$.
- How about multiplication? Encryptions are vectors, we cannot multiply them!
First imagine we’re trying to construct a secret-key homomorphic encryption scheme based on LWE.

Here’s a first attempt:

- Shared secret key: \( sk = s = (1, -s_0) \in \mathbb{Z}_q^{n+1} \), where \( s_0 \in \mathbb{Z}_q^n \) is uniformly random.
- \( E_{sk}(b) = c = (\langle s_0, a \rangle + 2e + b, a) \) for uniformly random \( a \in \mathbb{Z}_q^n \) and a small \( e \in \mathbb{Z}_q \).
- \( D_{sk}(c) = [\langle s, c \rangle]_q \mod 2. \)

Clearly, under LWE, this is secure: \( E_{sk}(0) \cong E_{sk}(1) \) since both are indistinguishable from a uniformly random vector in \( \mathbb{Z}_q^{n+1} \).

Additively homomorphically (somewhat): \( E_{sk}(b_1) + E_{sk}(b_2) \) decrypts to \( b_1 \oplus b_2 \).

How about multiplication? Encryptions are vectors, we cannot multiply them!
A secret key scheme

- First imagine we’re trying to construct a secret-key homomorphic encryption scheme based on LWE.
- Here’s a first attempt:
  - Shared secret key: $sk = s = (1, -s_0) \in \mathbb{Z}_{q+1}^n$, where $s_0 \in \mathbb{Z}_q^n$ is uniformly random.
  - $E_{sk}(b) = c = (\langle s_0, a \rangle + 2e + b, a)$ for uniformly random $a \in \mathbb{Z}_q^n$ and a small $e \in \mathbb{Z}_q$.
  - $D_{sk}(c) = [\langle s, c \rangle]_q \mod 2$.

- Clearly, under LWE, this is secure: $E_{sk}(0) \cong E_{sk}(1)$ since both are indistinguishable from a uniformly random vector in $\mathbb{Z}_q^{n+1}$.
- Additively homomorphically (somewhat): $E_{sk}(b_1) + E_{sk}(b_2)$ decrypts to $b_1 \oplus b_2$.
- How about multiplication? Encryptions are vectors, we cannot multiply them!
A secret key scheme

- First imagine we’re trying to construct a secret-key homomorphic encryption scheme based on LWE.
- Here’s a first attempt:
  - Shared secret key: \( sk = s = (1, -s_0) \in \mathbb{Z}_{q+1} \), where \( s_0 \in \mathbb{Z}_q \) is uniformly random.
  - \( E_{sk}(b) = c = (\langle s_0, a \rangle + 2e + b, a) \) for uniformly random \( a \in \mathbb{Z}_q^n \) and a small \( e \in \mathbb{Z}_q \).
  - \( D_{sk}(c) = [\langle s, c \rangle]_q \mod 2 \).
- Clearly, under LWE, this is secure: \( E_{sk}(0) \cong E_{sk}(1) \) since both are indistinguishable from a uniformly random vector in \( \mathbb{Z}_{q+1}^n \).
- Additively homomorphically (somewhat): \( E_{sk}(b_1) + E_{sk}(b_2) \) decrypts to \( b_1 \oplus b_2 \).
- How about multiplication? Encryptions are vectors, we cannot multiply them!
Outline

Breaking things with lattices
  Yesterday's game
  Howgrave-Graham on approximate GCDs

Fully homomorphic encryption from LWE
  Recap on LWE
  A secret key homomorphic scheme
  Achieving homomorphic multiplication
  Obtaining fully homomorphic encryption
Remark:

- In most previous FHE schemes, obtaining homomorphic operations was easy (ciphertexts were ring elements) and the hard part was to prove security.
- Here, security is easy; the hard part is to come up with a way to multiply ciphertexts.

One way to multiply vectors is tensor product:

To homomorphically multiply $c^{(1)}$ and $c^{(2)}$, publish:

$$c^* = c^{(1)} \otimes c^{(2)} = \left( c^{(1)}_i \cdot c^{(2)}_j \right)_{1 \leq i, j \leq n}$$

- We have $\langle s \otimes s, c^* \rangle = \langle s, c^{(1)} \rangle \cdot \langle s, c^{(2)} \rangle$, so we can decrypt (as long as the noise doesn’t get too large).
- Fine, but the new ciphertext $c^*$ is much larger (dimension $(n + 1)^2$) than the ones we started with!
Remark:

- In most previous FHE schemes, obtaining homomorphic operations was easy (ciphertexts were ring elements) and the hard part was to prove security.
- Here, security is easy; the hard part is to come up with a way to multiply ciphertexts.

One way to multiply vectors is tensor product:

- To homomorphically multiply $c^{(1)}$ and $c^{(2)}$, publish:

\[
\mathbf{c}^* = \mathbf{c}^{(1)} \otimes \mathbf{c}^{(2)} = \left( c_i^{(1)} \cdot c_j^{(2)} \right)_{1 \leq i,j \leq n}
\]

- We have $\langle s \otimes s, \mathbf{c}^* \rangle = \langle s, \mathbf{c}^{(1)} \rangle \cdot \langle s, \mathbf{c}^{(2)} \rangle$, so we can decrypt (as long as the noise doesn’t get too large).
- Fine, but the new ciphertext $\mathbf{c}^*$ is much larger (dimension $(n + 1)^2$) than the ones we started with!
Multiplication: reducing the size

- How do we convert $c^*$ to a ciphertext of the same length as what we started from?
  - The idea is key switching:
    - Publish "encryptions" $\sigma^*_{ij}$ of the components $s_i \cdot s_j$ of $s \otimes s$ under a new key $t$, i.e. vectors $\sigma^*_{ij}$ such that:
      $$\langle t, \sigma^*_{ij} \rangle = s_i \cdot s_j + 2e_{ij}$$
    - Let $c' = \sum_{ij} c^*_{ij} \sigma^*_{ij} \in \mathbb{Z}_q^{n+1}$.
    - We easily obtain:
      $$\langle s, c^{(1)} \rangle \cdot \langle s, c^{(2)} \rangle = \langle s \otimes s, c^* \rangle = \langle t, c' \rangle - 2 \sum_{i,j} c^*_{ij} e_{ij}$$
  - So under $c'$ decrypts under $t$ to the product $D_s(c^{(1)}) \cdot D_s(c^{(2)})$, provided that the blue sum is small. But it is not small!
  - Solution (rough idea): first decompose $c^*$ into bits, and apply the trick to the bit-decomposed extended ciphertext $c^{**}$. Since the $c^*_{ij}$’s are bits, the corresponding blue sum is small and we’re done.
Multiplication: reducing the size

- How do we convert $c^*$ to a ciphertext of the same length as what we started from?
- The idea is key switching:
  - Publish “encryptions” $\sigma_{ij}^*$ of the components $s_i \cdot s_j$ of $s \otimes s$ under a new key $t$, i.e. vectors $\sigma_{ij}^*$ such that:
    \[
    \langle t, \sigma_{i,j}^* \rangle = s_i \cdot s_j + 2e_{ij}
    \]
  - Let $c' = \sum_{ij} c_{ij}^* \sigma_{ij}^* \in \mathbb{Z}_q^{n+1}$.
  - We easily obtain:
    \[
    \langle s, c^{(1)} \rangle \cdot \langle s, c^{(2)} \rangle = \langle s \otimes s, c^* \rangle = \langle t, c' \rangle - 2 \sum_{i,j} c_{ij}^* e_{ij}
    \]
- So under $c'$ decrypts under $t$ to the product $D_s(c^{(1)}) \cdot D_s(c^{(2)})$, provided that the blue sum is small. But it is not small!
- Solution (rough idea): first decompose $c^*$ into bits, and apply the trick to the bit-decomposed extended ciphertext $c^{**}$. Since the $c_{ij}^{**}$’s are bits, the corresponding blue sum is small and we’re done.
Summing up

- We can publish vectors $\sigma_{ijk}^*$ that let you convert a bit-decomposed extended ciphertext $c^{**}$ to a ciphertext $c'$ of normal length under a new, independent key $t \in \mathbb{Z}_q^{n+1}$.

- This gives (somewhat) homomorphic multiplication:

$$D_t(c') = D_s(c^{(1)}) \cdot D_s(c^{(2)})$$

- Publishing that information doesn’t affect security, since under LWE, the vectors $\sigma_{ijk}^*$ are indistinguishable from random.

- Key switching works for any two keys, not just for multiplication: so publishing the vectors converting from the “null” key $0$ to $s$ turns the scheme to a public key scheme!

- This yields a leveled, somewhat homomorphic encryption scheme from LWE.
Summing up

- We can publish vectors $\sigma_{ijk}^*$ that let you convert a bit-decomposed extended ciphertext $c^{**}$ to a ciphertext $c'$ of normal length under a new, independent key $t \in \mathbb{Z}_{q}^{n+1}$.
- This gives (somewhat) homomorphic multiplication:
  $$D_t(c') = D_s(c^{(1)}) \cdot D_s(c^{(2)})$$
- Publishing that information doesn’t affect security, since under LWE, the vectors $\sigma_{ijk}^*$ are indistinguishable from random.
- Key switching works for any two keys, not just for multiplication: so publishing the vectors converting from the “null” key 0 to s turns the scheme to a public key scheme!
- This yields a leveled, somewhat homomorphic encryption scheme from LWE.
We can publish vectors $\sigma_{ijk}^*$ that let you convert a bit-decomposed extended ciphertext $c^{**}$ to a ciphertext $c'$ of normal length under a new, independent key $t \in \mathbb{Z}_q^{n+1}$.

This gives (somewhat) homomorphic multiplication:

$$D_t(c') = D_s(c^{(1)}) \cdot D_s(c^{(2)})$$

Publishing that information doesn’t affect security, since under LWE, the vectors $\sigma_{ijk}^*$ are indistinguishable from random.

Key switching works for any two keys, not just for multiplication: so publishing the vectors converting from the “null” key $0$ to $s$ turns the scheme to a public key scheme!

This yields a leveled, somewhat homomorphic encryption scheme from LWE.
Summing up

- We can publish vectors $\sigma_{ijk}^*$ that let you convert a bit-decomposed extended ciphertext $c^{**}$ to a ciphertext $c'$ of normal length under a new, independent key $t \in \mathbb{Z}_q^{n+1}$.

- This gives (somewhat) homomorphic multiplication:

$$D_t(c') = D_s(c^{(1)}) \cdot D_s(c^{(2)})$$

- Publishing that information doesn’t affect security, since under LWE, the vectors $\sigma_{ijk}^*$ are indistinguishable from random.

- Key switching works for any two keys, not just for multiplication: so publishing the vectors converting from the “null” key $0$ to $s$ turns the scheme to a public key scheme!

- This yields a leveled, somewhat homomorphic encryption scheme from LWE.
Summing up

- We can publish vectors $\sigma_{ijk}^*$ that let you convert a bit-decomposed extended ciphertext $c^{**}$ to a ciphertext $c'$ of normal length under a new, independent key $t \in \mathbb{Z}_q^{n+1}$.

- This gives (somewhat) homomorphic multiplication:

$$D_t(c') = D_s(c^{(1)}) \cdot D_s(c^{(2)})$$

- Publishing that information doesn't affect security, since under LWE, the vectors $\sigma_{ijk}^*$ are indistinguishable from random.

- Key switching works for any two keys, not just for multiplication: so publishing the vectors converting from the "null" key $0$ to $s$ turns the scheme to a public key scheme!

- This yields a leveled, somewhat homomorphic encryption scheme from LWE.
Outline

Breaking things with lattices
  Yesterday’s game
  Howgrave-Graham on approximate GCDs

Fully homomorphic encryption from LWE
  Recap on LWE
  A secret key homomorphic scheme
  Achieving homomorphic multiplication
  Obtaining fully homomorphic encryption
We discussed a technique (key switching) to convert a ciphertext under a long key $s \in \mathbb{Z}_q^N$ to an equivalent ciphertext under a short key $t \in \mathbb{Z}_q^n$, $n \ll N$.

A similar trick (modulus switching) lets you convert a ciphertext in $\mathbb{Z}_q^n$ to an equivalent ciphertext under a new key in $\mathbb{Z}_p^n$, $p \ll q$.

Application by Brakerski and Vaikuntanathan:

- Apply homomorphic operations over $\mathbb{Z}_q$.
- At the end, convert to $\mathbb{Z}_p$, $p \ll q$ to make the decryption circuit very shallow, and make Gentry’s bootstrapping technique (homomorphic evaluation of the decryption circuit) possible directly, without the former trick known as “squashing”, and without subset-sum assumptions.
FHE without squashing

- We discussed a technique (key switching) to convert a ciphertext under a long key $s \in \mathbb{Z}_q^N$ to an equivalent ciphertext under a short key $t \in \mathbb{Z}_q^n$, $n \ll N$.

- A similar trick (modulus switching) lets you convert a ciphertext in $\mathbb{Z}_q^n$ to an equivalent ciphertext under a new key in $\mathbb{Z}_p^n$, $p \ll q$.

- Application by Brakerski and Vaikuntanathan:
  - Apply homomorphic operations over $\mathbb{Z}_q$.
  - At the end, convert to $\mathbb{Z}_p$, $p \ll q$ to make the decryption circuit very shallow, and make Gentry’s bootstrapping technique (homomorphic evaluation of the decryption circuit) possible directly, without the former trick known as “squashing”, and without subset-sum assumptions.
FHE without squashing

We discussed a technique (key switching) to convert a ciphertext under a long key \( s \in \mathbb{Z}_q^N \) to an equivalent ciphertext under a short key \( t \in \mathbb{Z}_q^n, n \ll N \).

A similar trick (modulus switching) lets you convert a ciphertext in \( \mathbb{Z}_q^n \) to an equivalent ciphertext under a new key in \( \mathbb{Z}_p^n, p \ll q \).

Application by Brakerski and Vaikuntanathan:

- Apply homomorphic operations over \( \mathbb{Z}_q \).
- At the end, convert to \( \mathbb{Z}_p, p \ll q \) to make the decryption circuit very shallow, and make Gentry’s bootstrapping technique (homomorphic evaluation of the decryption circuit) possible directly, without the former trick known as “squashing”, and without subset-sum assumptions.
Leveled FHE without bootstrapping

- Alternate approach by Brakerski, Gentry and Vaikuntanathan:
  - Start from an initially large prime modulus, and apply modulus switching after each multiplication.
  - This makes noise size grow linearly instead of exponentially with circuit depth.
  - Hence, we can handle circuits of arbitrary (predetermined) polynomial size without bootstrapping.
  - Even with bootstrapping, we get much better performance than earlier.

- Yet another approach: leveled FHE without modulus switching.
  - Reduce ciphertext noise while still keeping the same modulus.
  - Possible if you put the message in the top bit of the ciphertext rather than the bottom bit (“scale-invariant scheme”), as in Vadim’s talk.
Leveled FHE without bootstrapping

- Alternate approach by Brakerski, Gentry and Vaikuntanathan:
  - Start from an initially large prime modulus, and apply modulus switching after each multiplication.
  - This makes noise size grow linearly instead of exponentially with circuit depth.
  - Hence, we can handle circuits of arbitrary (predetermined) polynomial size without bootstrapping.
  - Even with bootstrapping, we get much better performance than earlier.

- Yet another approach: leveled FHE without modulus switching.
  - Reduce ciphertext noise while still keeping the same modulus.
  - Possible if you put the message in the top bit of the ciphertext rather than the bottom bit ("scale-invariant scheme"), as in Vadim’s talk.
Thank you!