FHEW: Homomorphic Encryption Bootstrapping in less than a Second\textsuperscript{1}

Léo Ducas\textsuperscript{2}  Daniele Micciancio

UC San Diego

Charles River Crypto Day, April 2015

\textsuperscript{1}To appear in Eurocrypt 2015
\textsuperscript{2}Now at CWI
Outline

Introduction/Summary

The new NAND gate

Simpler Refreshing

Conclusion
The evolution of FHE

Fully Homomorphic Encryption has seen drastic changes since Gentry’s first proposal:

➤ [Rivest, Adleman, Dertouzos’78]: Open problem
➤ [Gentry’09]: ideal lattices, sparse subset-sum, squashing, etc.
➤ [Gentry, Halevi’11], [Brakerski, Vaikuntanathan’11]: no squash
➤ [Brakerski, Vaikuntanathan’11]: Subexponential LWE
➤ [Brakerski’12], [Alperin-Sheriff, Peiert’14]: (Polynomial) LWE
➤ Many more works improving efficiency, etc.
The evolution of FHE

Fully Homomorphic Encryption has seen drastic changes since Gentry’s first proposal:

► [Rivest,Adleman,Dertouzos’78]: Open problem
► [Gentry’09]: ideal lattices, sparse subset-sum, squashing, etc.
► [Gentry,Halevi’11],[Brakerski,Vaikuntanathan’11]: no squash
► [Brakerski,Vaikuntanathan’11]: Subexponential LWE
► [Brakerski’12],[Alperin-Sheriff,Peiert’14]: (Polynomial) LWE
► Many more works improving efficiency, etc.

Still, all schemes have a common ingredient:

**Key technique**

Gentry’s FHE bootstrapping
FHE Bootstrapping

All known FHE schemes are based on noisy encryption schemes:

- Decryption is possible only when noise is sufficiently small.
- Noise grows when computing on ciphertexts.
- After a while, no more operations can be performed.
FHE Bootstrapping

All known FHE schemes are based on noisy encryption schemes:
- Decryption is possible only when noise is sufficiently small.
- Noise grows when computing on ciphertexts.
- After a while, no more operations can be performed.

FHE Bootstrapping:
- Method to “reset” the noise level of a ciphertext
- Idea: homomorphically compute ciphertext decryption function on encrypted key

\[ \text{Dec}_\cdot(\text{Enc}_k(m)) \]
FHE Bootstrapping

All known FHE schemes are based on noisy encryption schemes:

- Decryption is possible only when noise is sufficiently small.
- Noise grows when computing on ciphertexts.
- After a while, no more operations can be performed.

FHE Bootstrapping:

- Method to “reset” the noise level of a ciphertext
- Idea: homomorphically compute ciphertext decryption function on encrypted key

\[
\begin{align*}
\text{Dec}(\cdot)(\text{Enc}_k(m)) &= \text{Dec}(\cdot)(\text{Enc}_k(m)) \\
\text{Enc}(k) &\rightarrow \text{Enc}(m)
\end{align*}
\]
The quality/noise of the output $c'$ depends on

1. the quality/noise of $\text{Enc}(k)$, which is a fresh ciphertext, and
2. the complexity of $\text{Dec}(\cdot)(c = \text{Enc}_k(m))$,

but not the quality/noise of $c$, as long as it decrypts.

**Lattice Cryptography**

**Basic homomorphic properties**

**Low-complexity decryption**

Still, even if $\text{Dec}(\cdot)(c)$ is efficient, bootstrapping is very costly because $\text{Dec}(\cdot)(c)$ needs to be computed homomorphically on an encrypted $\text{Enc}(k)$. 
The quality/noise of the output $c'$ depends on

- the quality/noise of $Enc(k)$, which is a fresh ciphertext, and
- the complexity of $Dec(\cdot)(c)$,
- but not the quality/noise of $c$, as long as it decrypts
FHE Bootstrapping (cont.)

\[ \text{Dec}(\cdot)(c = \text{Enc}_k(m)) \]

The quality/noise of the output \( c' \) depends on
- the quality/noise of \( \text{Enc}(k) \), which is a fresh ciphertext, and
- the complexity of \( \text{Dec}(\cdot)(c) \),
- but not the quality/noise of \( c \), as long as it decrypts

Lattice Cryptography
- Basic homomorphic properties
- Low-complexity decryption \( \text{Dec}(\cdot)(c) \)
FHE Bootstrapping (cont.)

\[ \text{Dec}(\cdot)(c = \text{Enc}_k(m)) \]

\[ \text{Enc}(k) \rightarrow \text{Dec}(\cdot)(c = \text{Enc}_k(m)) \rightarrow c' = \text{Enc}(m) \]

The quality/noise of the output \( c' \) depends on

- the quality/noise of \( \text{Enc}(k) \), which is a fresh ciphertext, and
- the complexity of \( \text{Dec}(\cdot)(c) \),
- but not the quality/noise of \( c \), as long as it decrypts

Lattice Cryptography

- Basic homomorphic properties
- Low-complexity decryption \( \text{Dec}(\cdot)(c) \)

Still, even if \( \text{Dec}(\cdot)(c) \) is efficient, bootstrapping is very costly because \( \text{Dec}(\cdot)(c) \) needs to be computed homomorphically on an encrypted \( \text{Enc}(k) \).
The Problem

FHE Bootstrapping/Refreshing is an expensive process:
- [Halevi, Shoup’14,’15] HElib: 6-30 mins
The Problem

FHE Bootstrapping/Refreshing is an expensive process:
- [Halevi, Shoup’14,’15] HElib: 6-30 mins

Mitigating the cost of bootstrapping (previous approaches):
- SIMD-FHE: Perform many refresh operations in parallel
- Noise control: allow more computation before refreshing
- HElib: Cost can be amortized over $\approx 1000$ binary ciphertext.
The Problem

FHE Bootstrapping/Refreshing is an expensive process:

- [Halevi, Shoup’14,’15] HElib: 6-30 mins

Mitigating the cost of bootstrapping (previous approaches):

- SIMD-FHE: Perform many refresh operations in parallel
- Noise control: allow more computation before refreshing
- HElib: Cost can be amortized over $\approx 1000$ binary ciphertexts.

**Question**

How fast can we refresh a single ciphertext?
Contributions

Question

How fast can we refresh for a single ciphertext?
Contributions

Question

How fast can we refresh for a single ciphertext?

We give a proof of concept solution in 0.6 seconds: amortized cost comparable to [HElib], but without the delay...

Two new techniques:

▶ a new, cheap NAND gate
▶ a simpler refreshing procedure using ring structure
The new NAND gate

**Base:** Start from LWE encryption with message space: $\mathbb{Z}_t$, $t \geq 2$.

**Idea:** Different message space for input ($t = 4$) and output ($t = 2$).
The new NAND gate

**Base:** Start from LWE encryption with message space: $\mathbb{Z}_t$, $t \geq 2$.

**Idea:** Different message space for input ($t = 4$) and output ($t = 2$).

**Advantages:**
- Cost of computing Homomorphic NAND is negligible (similar to a single private key cryptographic operation.)
- Excellent noise growth: $\epsilon$ grows only by a small constant factor.
- Substantially simplifies the task faced by the Refreshing procedure.
A simpler refreshing procedure


**Idea:** Implement arithmetic mod $q$ *in the exponent*
A simpler refreshing procedure


**Idea:** Implement arithmetic mod $q$ in the exponent

**Improvement over [AP14]:**
- Theoretical speed-up of $\tilde{\Omega}(\log^3 q)$
- Smaller final error.

Combined with the problem simplification brought by our cheap NAND computation, this results in bootstrapping cost $\approx 0.6$ second, at estimated $\approx 100$-bit security level.
Outline

Introduction/Summary

The new NAND gate

Simpler Refreshing

Conclusion
LWE and Symmetric Encryption

Definition (Learning with Errors)

- **For** a random secret \( s \in \mathbb{Z}_q^n \)
- **Given** many sample \((a, b = \langle a, s \rangle + e)\) where \( e \leftarrow \chi, \) small
- **Distinguish** the samples from uniformly random

LWE is as hard as worst-case lattice problems

\[
\text{Enc}_s(m \in \mathbb{Z}_2) = (a, b = \langle a, s \rangle + e + m \cdot q/2) \\
\text{Dec}_s(a, b) = \lfloor 2(b - \langle a, s \rangle)/q \rfloor
\]

Sets of encryptions of \( m \) with error \( e < E \) noted \( \text{LWE}_s(m, E) \).

**Correct decryption** ensured if \((a, b) \in \text{LWE}_s(\cdot, q/4)\)
LWE and Symmetric Encryption

Definition (Learning with Errors)

- For a random secret $s \in \mathbb{Z}_q^n$
- Given many sample $(a, b = \langle a, s \rangle + e)$ where $e \leftarrow \chi$, small
- Distinguish the samples from uniformly random

LWE is as hard as worst-case lattice problems

$$
\text{Enc}_s(m \in \mathbb{Z}_2) = (a, b = \langle a, s \rangle + e + m \cdot q/2) \\
\text{Dec}_s(a, b) = \lceil 2(b - \langle a, s \rangle)/q \rceil
$$

Sets of encryptions of $m$ with error $e < E$ noted $\text{LWE}_s(m, E)$.
Correct decryption ensured if $(a, b) \in \text{LWE}_s(\cdot, q/4)$
LWE and Symmetric Encryption

**Definition (Learning with Errors)**

- **For** a random secret \( s \in \mathbb{Z}_q^n \)
- **Given** many sample \((a, b = \langle a, s \rangle + e)\) where \( e \leftarrow \chi\), small
- **Distinguish** the samples from uniformly random

LWE is as hard as worst-case lattice problems

\[
\begin{align*}
\text{Enc}_s(m \in \mathbb{Z}_2) &= (a, b = \langle a, s \rangle + e + m \cdot q/2) \\
\text{Dec}_s(a, b) &= \left\lfloor 2(b - \langle a, s \rangle)/q \right\rfloor
\end{align*}
\]

Sets of encryptions of \( m \) with error \( e < E \) noted \( \text{LWE}_s(m, E) \).

**Correct decryption** ensured if \((a, b) \in \text{LWE}_s(\cdot, q/4)\)
Homomorphic Operation on LWE ciphertext

**Addition/XOR operation** as the sum of ciphertexts:

\[
\text{LWE}_s(m_1, e_1) \times \text{LWE}_s(m_2, e_2) \rightarrow \text{LWE}_s(m_1 \oplus m_2, e_1 + e_2)
\]
Homomorphic Operation on LWE ciphertext

**Addition/XOR operation** as the sum of ciphertexts:

\[ \text{LWE}_s(m_1, e_1) \times \text{LWE}_s(m_2, e_2) \rightarrow \text{LWE}_s(m_1 \oplus m_2, e_1 + e_2) \]

*(Traditional) (N)AND operation* as the tensor of ciphertexts:

\[ \text{LWE}_{s_1}(m_1, e_1) \times \text{LWE}_{s_2}(m_2, e_2) \rightarrow \text{LWE}_{s_1 \otimes s_2}(m_1 \land m_2, e_1 \cdot e_2) \]

⇒ FHE bootstrapping requires **strong Refreshing**:

\[ \text{LWE}_s(m, e) \rightarrow \text{LWE}_s(m, e'), \quad e' \ll e. \]

Techniques: Key Switching, Mod Switching, and Homomorphic Decryption.
LWE encryption with different message spaces

Idea: use an LWE sample as a mask

Enc_s(m) = (a, b = ⟨a, s⟩ + e + m · q/2)
Dec_s(a, b) = ⌊2(b − ⟨a, s⟩)/q⌋

m · q/2 + e

binary messages
LWE_2^2(m, q/4)
LWE encryption with different message spaces

Idea: use an LWE sample as a mask

\[
\text{Enc}_s(m) = (a, b = \langle a, s \rangle + e + m \cdot q/2) \\
\text{Dec}_s(a, b) = \left\lfloor 2(b - \langle a, s \rangle)/q \right\rfloor
\]

\[m \cdot \frac{q}{2} + e\]

binary messages
LWE_2^s(m, q/4)

\[m \cdot \frac{q}{4} + e\]

Messages in \(\mathbb{Z}_4\)
LWE_4^s(m, q/8)
LWE encryption with different message spaces

Idea: use an LWE sample as a mask

\[
\text{Enc}_s(m) = (a, b = \langle a, s \rangle + e + m \cdot q/2)
\]
\[
\text{Dec}_s(a, b) = \lceil 2(b - \langle a, s \rangle)/q \rceil
\]

\[
m \cdot \frac{q}{2} + e
\]

\[
m \cdot \frac{q}{4} + e
\]

\[
m \cdot \frac{q}{4} + e
\]

binary messages
LWE_s^2(m, q/4)

Messages in \( \mathbb{Z}_4 \)
LWE_s^4(m, q/8)

Smaller error
LWE_s^4(m, q/16)
A Cheap NAND gate

Idea, use: \( m_1 \land m_2 \iff m_1 + m_2 = 2 \mod 4 \).
Consider binary messages \( \{0, 1\} \) encrypted with \( t = 4 \):
A Cheap NAND gate

Idea, use: \( m_1 \land m_2 \iff m_1 + m_2 = 2 \mod 4 \).
Consider binary messages \( \{0, 1\} \) encrypted with \( t = 4 \):

\[
\begin{array}{cc}
\begin{array}{ccc}
3 & 2 & 1 \\
0 & & \\
\end{array} & + & \begin{array}{ccc}
3 & 2 & 1 \\
0 & & \\
\end{array} = \begin{array}{ccc}
3 & 2 & 1 \\
0 & & \\
\end{array}
\end{array}
\]
A Cheap NAND gate

Idea, use: \( m_1 \land m_2 \iff m_1 + m_2 = 2 \mod 4 \).
Consider binary messages \( \{0, 1\} \) encrypted with \( t = 4 \):

\[
\begin{array}{c}
3 & 2 & 1 \\
0 & 3 & 2
\end{array}
\begin{array}{c}
+ \\
= \\
\begin{array}{c}
3 & 2 & 1 \\
0 & 3 & 2
\end{array}
\end{array}
\]

Consider it as a ciphertext for \( t = 2 \) and rotate.
A Cheap NAND gate

Idea, use: \( m_1 \land m_2 \iff m_1 + m_2 = 2 \mod 4 \).
Consider binary messages \( \{0, 1\} \) encrypted with \( t = 4 \):

\[
\begin{array}{ccc}
3 & 2 & 1 \\
3 & 0 & 0
\end{array}
\begin{array}{ccc}
2 & 2 & 1 \\
0 & 0 & 0
\end{array}
= 
\begin{array}{ccc}
1 & 0 & 0 \\
1 & 0 & 0
\end{array}
\]

Consider it as a ciphertext for \( t = 2 \) and rotate.
A Cheap NAND gate

Idea, use: \( m_1 \land m_2 \iff m_1 + m_2 = 2 \mod 4 \).
Consider binary messages \( \{0, 1\} \) encrypted with \( t = 4 \):

\[
\begin{array}{c}
3 \quad 2 \quad 1 \\
\downarrow \quad \downarrow \quad \downarrow \\
0 \quad 0 \quad 1
\end{array}
+ \quad \begin{array}{c}
3 \quad 2 \quad 1 \\
\downarrow \quad \downarrow \quad \downarrow \\
0 \quad 0 \quad 1
\end{array}
= \begin{array}{c}
1 \\
\downarrow
\end{array}
\]

Consider it as a ciphertext for \( t = 2 \) and rotate.
A Cheap NAND gate

Idea, use: \( m_1 \land m_2 \iff m_1 + m_2 = 2 \mod 4 \).
Consider binary messages \( \{0, 1\} \) encrypted with \( t = 4 \):

\[
\begin{array}{ccc}
3 & 2 & 0 \\
\hline
3 & 2 & 1 \\
\hline
0 & 1 & 1
\end{array}
\]

Consider it as a ciphertext for \( t = 2 \) and rotate.

HomNAND:

\[
\text{LWE}_s^4(m_1, \frac{q}{16}) \times \text{LWE}_s^4(m_2, \frac{q}{16}) \rightarrow \text{LWE}_s^2(m_1 \overline{\land} m_2, \frac{q}{4})
\]

\[
(a_1, b_1), (a_2, b_2) \iff (a_1 + a_2, b_1 + b_2 + \frac{5q}{8})
\]
Lightweight Refreshing

We have HomNAND:

\[
\text{LWE}_s^4 \left( m_1, \frac{q}{16} \right) \times \text{LWE}_s^4 \left( m_2, \frac{q}{16} \right) \rightarrow \text{LWE}_s^2 \left( m_1 \bar{\land} m_2, \frac{q}{4} \right)
\]
Lightweight Refreshing

We have HomNAND:

\[
\text{LWE}_s^4 \left( m_1, \frac{q}{16} \right) \times \text{LWE}_s^4 \left( m_2, \frac{q}{16} \right) \rightarrow \text{LWE}_s^2 \left( m_1 \land m_2, \frac{q}{4} \right)
\]

To build an FHE we require a relaxed function LightRefresh:

\[
\text{LightRefresh} : \text{LWE}_s^2 (m, q/4) \rightarrow \text{LWE}_s^4 (m, q/16)
\]
Lightweight Refreshing

We have HomNAND:

$$LWE_4^s \left( m_1, \frac{q}{16} \right) \times LWE_4^s \left( m_2, \frac{q}{16} \right) \rightarrow LWE_2^s \left( m_1 \land m_2, \frac{q}{4} \right)$$

To build an FHE we require a relaxed function LightRefresh:

$$\text{LightRefresh} : LWE_2^s (m, q/4) \rightarrow LWE_4^s (m, q/16)$$

whereas previous works required:

$$\text{Refresh} : LWE_2^s (m, q/4) \rightarrow LWE_2^s (m, E), E \ll q.$$  

As usual, we will use Key Switching, Mod Switching, and Homomorphic Decryption.
Outline

Introduction/Summary

The new NAND gate

Simpler Refreshing

Conclusion
Decryption using an Accumulator

\[
\text{Dec}_s(a, b) = \text{msb} \left( b - \langle a, s \rangle \mod q \right) = \text{msb} \left( b - \sum_{i} a_i \cdot s_i \mod q \right)
\]

**Dec}_s(a, b):**

\[
\begin{align*}
\text{acc} & \leftarrow b \\
\text{for } i = 1 \text{ to } n:
\quad \text{acc} & \leftarrow \text{acc} - a_i \cdot s_i \mod q \\
\text{Return } \text{msb}(\text{acc})
\end{align*}
\]
Decryption using an Accumulator

Dec$_s$(a, b) = msb (b − ⟨a, s⟩ mod q) = msb \left( b - \sum_{i} a_i \cdot s_i \mod q \right)

Homomorphic decryption given $E'(s) = [E(a \cdot s_i) | i, a]$:

\[
E(acc) \leftarrow b \\
\text{for } i = 1 \text{ to } n: \\
E(acc) \leftarrow E(acc) - E(a_i \cdot s_i) \mod q \\
\text{Return LWE}(\text{msb}(acc))
\]

ACC = E(acc) holds an encrypted integer acc ∈ ℤ$_q$

The accumulator ACC should support the following operations:
Decryption using an Accumulator

\[
\text{Dec}_s(a, b) = \text{msb} \left( b - \langle a, s \rangle \mod q \right) = \text{msb} \left( b - \sum_i a_i \cdot s_i \mod q \right)
\]

Homomorphic decryption given \( E'(s) = [E(a \cdot s_i) \mid i, a] \):

- \( E(acc) \leftarrow b \)
- for \( i = 1 \) to \( n \):
  - \( E(acc) \leftarrow E(acc) - E(a_i \cdot s_i) \mod q \)
- Return \( \text{LWE}(\text{msb}(acc)) \)

\( ACC = E(acc) \) holds an encrypted integer \( acc \in \mathbb{Z}_q \)

The accumulator \( ACC \) should support the following operations:

- **Initialization:** \( ACC \leftarrow b \)
- **Addition** \( ACC \leftarrow ACC + c \) of a fresh ciphertext \( c = E(a \cdot s_i) \)
- **Extract encrypted MSB:** \( ACC \rightarrow \text{LWE}(\text{msb}(acc)) \)
Implementing the Accumulator

The framework of [AP14] based on [GSW13] \((E, +, \cdot)\):

- \(ACC = [E(a_{q-1}), \ldots, E(a_1), E(a_0)]\) with \(a_i = \delta_{i=acc} \in \{0, 1\}\)

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]

\[\]
Implementing the Accumulator

The framework of [AP14] based on [GSW13] \((E, +, \cdot)\):

- \(ACC = [E(a_{q-1}), \ldots, E(a_1), E(a_0)]\) with \(a_i = \delta_{i=\text{acc}} \in \{0, 1\}\)
- Increment \(ACC \leftarrow ACC + \text{Enc}(b), b \in \{0, 1\}\):
  \[ACC[i] \leftarrow ACC[i] \cdot (1 - E(b)) + ACC[i - 1] \cdot E(b)\]
Implementing the Accumulator

The framework of [AP14] based on [GSW13] \((E, +, \cdot)\):

- \(ACC = [E(a_{q-1}), \ldots, E(a_1), E(a_0)]\) with \(a_i = \delta_{i=acc} \in \{0, 1\}\)

- Increment \(ACC \leftarrow ACC + Enc(b), b \in \{0, 1\}\):
  \(ACC[i] \leftarrow ACC[i] \cdot (1 - E(b)) + ACC[i - 1] \cdot E(b)\)

- MSB extraction: \(MSB(ACC) = \sum_{q/2}^{q-1} Enc(a_i)\).
Implementing the Accumulator

The framework of [AP14] based on [GSW13] \((E, +, \cdot)\):

- \(ACC = [E(a_{q-1}), \ldots, E(a_1), E(a_0)]\) with \(a_i = \delta_{i=acc} \in \{0, 1\}\)
- Increment \(ACC \leftarrow ACC + Enc(b), b \in \{0, 1\}\):
  \(ACC[i] \leftarrow ACC[i] \cdot (1 - E(b)) + ACC[i-1] \cdot E(b)\)
- MSB extraction: \(MSB(ACC) = \sum_{q/2}^{q-1} Enc(a_i)\).
- Optimized using CRT and product of many small cyclic rings.
Implementing the Accumulator

The framework of [AP14] based on [GSW13] \((E, +, \cdot)\):

- \( ACC = [E(a_{q-1}), \ldots, E(a_1), E(a_0)] \) with \( a_i = \delta_{i=\text{acc}} \in \{0, 1\} \)
- Increment \( ACC \leftarrow ACC + Enc(b), \; b \in \{0, 1\} \):
  \[
  ACC[i] \leftarrow ACC[i] \cdot (1 - E(b)) + ACC[i - 1] \cdot E(b)
  \]
- MSB extraction: \( MSB(ACC) = \sum_{q/2}^{q-1} Enc(a_i) \).
- Optimized using CRT and product of many small cyclic rings.

We optimize this construction using the cyclotomic ring

\[
\mathcal{R} = \mathbb{Z}[X]/(X^N + 1).
\]

- Embed \( \mathbb{Z}_q \) in the group \((\{X^i\}_i, \cdot)\), of roots of unity, \( q = 2N \).
- \( ACC \) uses only a single ciphertext \( E(X^{\text{acc}}) \).
Ring version of [GSW13]

\[ Q = 2^k, \text{ Gadget matrix: } G = [I, 2I, 4I \ldots 2^{k-1}I]^t \in \mathbb{Z}_Q^{n+1 \times (n+1)k}. \]

\[ E_s(m) = [A, As + e] + m \cdot G \]

Dec\(_s\): extract an LWE\(_s\) ciphertext (last row) and decrypt.
Supports Add. and Mult. for small messages \( m \in \{-1, 0, 1\} \).
Ring version of [GSW13]

\[ Q = 2^k, \text{ Gadget matrix: } G = [l, 2l, 4l \ldots 2^{k-1}l]^t \in \mathbb{Z}_Q^{n+1 \times (n+1)k}. \]

\[ E_s(m) = [A, As + e] + m \cdot G \]

Dec_s: extract an LWE_s ciphertext (last row) and decrypt. Supports Add. and Mult. for **small messages** \( m \in \{-1, 0, 1\}. \)

Cyclotomic ring \( \mathcal{R} = \mathbb{Z}[X]/(X^N + 1), \quad 2N = q \) is a power of 2.

Generalized Gadget matrix: \( G = u \cdot [l, bl, b^2l \ldots b^{k-1}l]^t \in \mathcal{R}_Q^{2 \times 2k}. \)

\[ E_s(m \in \mathbb{Z}_q) = [a, a \cdot s + e] + X^m \cdot G \]

Supports addition for all message \( m \in \mathbb{Z}_q. \)
The group of roots of unity and msb

\[
\begin{align*}
\begin{array}{c|cccc|cccc}
\ m & 0 & 1 & \ldots & q/2 - 1 & q/2 & q/2 + 1 & \ldots & q - 1 \\
X^m & 1 & X & \ldots & X^{N-1} & -1 & -X & \ldots & -X^{N-1}
\end{array}
\end{align*}
\]

Take the vector representation of \( X^m \)

\[
\begin{align*}
\begin{array}{c|c|c|c|c|c|c|c|c|c}
\ x_m & \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} & \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix} & \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} & \begin{bmatrix} -1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} & \begin{bmatrix} 0 \\ -1 \\ \vdots \\ 0 \end{bmatrix} & \begin{bmatrix} 0 \\ 0 \\ \vdots \\ -1 \end{bmatrix} & \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} & \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}
\end{array}
\end{align*}
\]

Sum all the coordinates

\[
\begin{align*}
\langle 1, x_m \rangle & \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} & \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} & \begin{bmatrix} -1 \\ -1 \\ \vdots \\ -1 \end{bmatrix}
\end{align*}
\]

\[
\frac{\langle 1, x_m \rangle + 1}{2} = \frac{(-1)^{\text{msb}(m)} + 1}{2} = \text{msb}(m).
\]
Extracting $LWE_s^4(\text{msb}(m))$

Recall: $G = u \cdot [l, bl, b^2l \ldots b^{k-1}l]^t \in \mathcal{R}_Q^{2 \times 2^k}$ and

$$E_s(m \in \mathbb{Z}_q) = [a, a \cdot s + e] + X^m \cdot G$$
Extracting $\text{LWE}_s^4(\text{msb}(m))$

Recall: $G = u \cdot [1, b_1, b_2 1 \ldots b^{k-1} 1]^t \in \mathcal{R}_{Q}^{2 \times 2^k}$ and

$$E_s(m \in \mathbb{Z}_q) = [a, a \cdot s + e] + X^m \cdot G$$

Set $u = q/8$, take the 2\textsuperscript{nd} row, in vector representation:

$$C = \begin{bmatrix} A', A' \cdot s + e + \frac{Q}{8} \cdot x_m \end{bmatrix}$$
Extracting $\text{LWE}_s^4(\text{msb}(m))$

Recall: $G = u \cdot [1, b_1, b_21 \ldots b^{k-1}1]^t \in \mathcal{R}^{2 \times 2^k}_Q$ and

$$E_s(m \in \mathbb{Z}_q) = [a, a \cdot s + e] + X^m \cdot G$$

Set $u = q/8$, take the 2nd row, in vector representation:

$$C = \begin{bmatrix} A', A' \cdot s + e + \frac{Q}{8} \cdot x_m \end{bmatrix}$$

Sum all rows and add $q/8$:

$$1^t \cdot C = 1^t \cdot \begin{bmatrix} A', A' \cdot s + e + \frac{Q}{8} \cdot x_m + \frac{Q}{8} \end{bmatrix}$$

$$= \begin{bmatrix} a', a' \cdot s + e' + \frac{Q}{4} \cdot \text{msb}(m) \end{bmatrix}$$

Obtain an LWE encryption of msb($m$) with message space $\mathbb{Z}_4$. 
Improvements

Improvement over the bootstrapping of [AP14]:

- Generic $\tilde{\Omega}(n)$ speed-up from Ring structure
- An extra $\tilde{\Omega}(\log^3 q)$ speed-up by embedding
- Error after bootstrapping reduced by $O(\sqrt{n} \log n)$.

In addition to our new NAND gate, implementation becomes reasonable.
Outline

Introduction/Summary

The new NAND gate

Simpler Refreshing

Conclusion
The ciphertext cycle

\[ \text{LWE}_{n}^{4/q}(m_1, q/16) \]

\[ \text{LWE}_{n}^{4/q}(m_2, q/16) \]

NAND

\[ \text{ACC operations and msbTest} \]

\[ \text{LWE}_{n}^{2/q}(m, q/4) \]

\[ \text{LWE}_{n}^{4/Q} \left( m, \sigma \tilde{O}(N^{3/2}) \right) \]

Key Switch

\[ \text{LWE}_{n}^{4/4}(m, q/16) \]

Modulus Switch

\[ \text{LWE}_{s_1}^{4/q}(m, q/16) \]
Parameter Proposal

**Parameters.**
LWE parameters: \( n = 410 \quad q = 512. \)
Ring-GSW parameters: \( N = 1024 \quad Q = 2^{32} \).
Gadget Matrix: \( Q/8 \cdot [l, 2^{11} \cdot l, 2^{22} \cdot l] \)

**Key Size.**
- Bootstrapping Key Size: 846 MB
- Key Switching Key Size: +135 MB \( \leq 1 \text{GB} \)

**Running time.**
- Per NAND gate: 39,360 FFTs \( \approx 0.4 \text{ sec} \)

**Security.**
- Security of the LWE scheme \( \delta_1 = 1.0060 \)
- Security of the Ring-GSW scheme \( \delta_2 = 1.0060 \)
Proof of Concept Implementation

- Coded in 4 days·man
  room for implementation level optimization.
- Reasonably concise: \( \leq 600 \) lines of C++ code
  [HElib]: \( \approx 20,000 \) lines
- Using FFT\(^3\) over \( \mathbb{C} \) at double precision
  in dimension 2048 to obtain negacyclic-FFT.
  potentially slower than 32-bits NTT in dimension 1024.

**Result:** Homomorphic NAND & refreshing in 0.61 seconds
on a single standard 64-bit core at 3Ghz.

Comparable to the amortized cost of bootstrapping in [HElib].

\(^3\)FFTW library: The Fastest Fourier Transform in the West
Beyond binary gates

Replace msb by other membership testing function. Obtain:

- moderate fan-in symmetric gates
- majority, threshold gates, neurons
- multiple outputs gates

**Figure:** Full-adder in **one refresh** (3-inputs, 2-outputs)

\[
\begin{align*}
m_1 + m_2 + m_3 &\in \{1, 3, 5\} \\
m_1 \oplus m_2 \oplus m_3 &\\ \\
carry(m_1, m_2, m_3) &\in \{2, 3, 4\}
\end{align*}
\]
The Next Steps

- Generalize the construction to arbitrary cyclotomic: we are restricted to powers of 2
- Exploit previous work using tensor-CRT for coprimes $p, q$:

\[
\text{Roots of unity}(\mathbb{Z}[X]/\Phi_p(X)) \simeq \mathbb{Z}_p
\]
\[
\text{Roots of unity}(\mathbb{Z}[X]/\Phi_p(X) \otimes \mathbb{Z}[X]/\Phi_q(X)) \simeq \mathbb{Z}_p \times \mathbb{Z}_q \simeq \mathbb{Z}_{pq}
\]

- Obtain a full S-box for the price of 1 or 2 Refresh operation.
The Next Steps

▶ Generalize the construction to arbitrary cyclotomic:
   we are restricted to powers of 2
▶ Exploit previous work using tensor-CRT for coprimes $p, q$:

\[
\text{Roots of unity}(\mathbb{Z}[X]/\Phi_p(X)) \cong \mathbb{Z}_p
\]
\[
\text{Roots of unity}(\mathbb{Z}[X]/\Phi_p(X) \otimes \mathbb{Z}[X]/\Phi_q(X)) \cong \mathbb{Z}_p \times \mathbb{Z}_q \cong \mathbb{Z}_{pq}
\]
▶ Obtain a full S-box for the price of 1 or 2 Refresh operation.

Thank You