FHEW: Homomorphic Encryption Bootstrapping in less than a Second[1]

Léo Ducas[2] Daniele Micciancio

UC San Diego

May 2015

[1] Eurocrypt 2015
[2] Now at CWI

Outline

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.

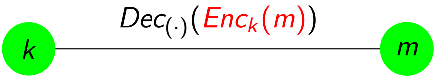
FHE Bootstrapping

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



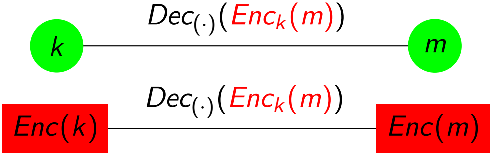
FHE Bootstrapping

All known FHE schemes are based on noisy encryption schemes:

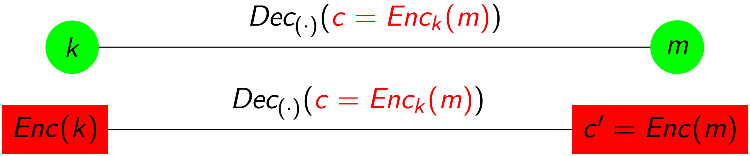
- Decryption is possible only when noise is sufficiently small.
- Noise grows when computing on ciphertexts.
- After a while, no more operations can be performed.

FHE Bootstrapping:

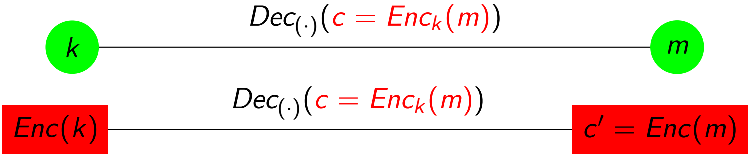
- Method to "reset" the noise level of a ciphertext
- Idea: homomorphically compute ciphertext decryption function on encrypted key



FHE Bootstrapping (cont.)



FHE Bootstrapping (cont.)



The quality/noise of the output c' depends on

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

FHE Bootstrapping (cont.)

$$k \xrightarrow{\quad Dec_{(\cdot)}(c = Enc_k(m)) \quad} m$$

$$Enc(k) \xrightarrow{\quad Dec_{(\cdot)}(c = Enc_k(m)) \quad} c' = Enc(m)$$

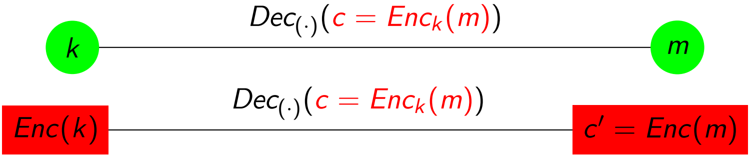
The quality/noise of the output c' depends on

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

Lattice Cryptography

- Basic homomorphic properties
- Low-complexity decryption $Dec_{(\cdot)}(c)$

FHE Bootstrapping (cont.)



The quality/noise of the output c' depends on

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

Lattice Cryptography

- Basic homomorphic properties
- Low-complexity decryption $Dec_{(\cdot)}(c)$

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

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 ≈ 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 ciphertext.

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: $\mathbb{Z}_t$, $t \geqslant 2$.

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

The new NAND gate

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

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: ϵ grows only by a small constant factor.
- Substantially simplifies the task faced by the Refreshing procedure.

A simpler refreshing procedure

Base: General approach of [Alperin-Sheriff,Peikert'14] + Ring variant of [Gentry,Sahai,Waters'13] Homomorphic encryption.

Idea: Implement arithmetic $\bmod q$ **in the exponent**

A simpler refreshing procedure

Base: General approach of [Alperin-Sheriff,Peikert'14] + Ring variant of [Gentry,Sahai,Waters'13] Homomorphic encryption.

Idea: Implement arithmetic $\bmod q$ **in the exponent**

Improvement over [AP14]:

- Theoretical speed-up of $\tilde{\Omega}(\log^3 q)$
- Smaller final error.

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

Outline

LWE and Symmetric Encryption

Definition (Learning with Errors)

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

LWE is as hard as worst-case lattice problems

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

Sets of encryptions of m with error $e < E$ noted $\mathrm{LWE}_{\mathbf{s}}(m, E)$.
Correct decryption ensured if $(\mathbf{a}, b) \in \mathrm{LWE}_{\mathbf{s}}(\,\cdot\,, q/4)$

LWE and Symmetric Encryption

Definition (Learning with Errors)

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

LWE is as hard as worst-case lattice problems

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

Sets of encryptions of m with error $e < E$ noted $\mathrm{LWE}_{\mathbf{s}}(m, E)$.
Correct decryption ensured if $(\mathbf{a}, b) \in \mathrm{LWE}_{\mathbf{s}}(\ \cdot\ , q/4)$

LWE and Symmetric Encryption

Definition (Learning with Errors)

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

LWE is as hard as worst-case lattice problems

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

Sets of encryptions of m with error $e < E$ noted $\mathrm{LWE}_{\mathbf{s}}(m, E)$.
Correct decryption ensured if $(\mathbf{a}, b) \in \mathrm{LWE}_{\mathbf{s}}(\,\cdot\,, q/4)$

Homomorphic Operation on LWE ciphertext

Addition/XOR operation as the sum of ciphertexts:

$$\mathsf{LWE}_{\mathbf{s}}(m_1, e_1) \times \mathsf{LWE}_{\mathbf{s}}(m_2, e_2) \rightarrow \mathsf{LWE}_{\mathbf{s}}(m_1 \oplus m_2, e_1 + e_2)$$

Homomorphic Operation on LWE ciphertext

Addition/XOR operation as the sum of ciphertexts:

$$\mathsf{LWE}_{\mathbf{s}}(m_1, e_1) \times \mathsf{LWE}_{\mathbf{s}}(m_2, e_2) \to \mathsf{LWE}_{\mathbf{s}}(m_1 \oplus m_2, e_1 + e_2)$$

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

$$\mathsf{LWE}_{\mathbf{s}_1}(m_1, e_1) \times \mathsf{LWE}_{\mathbf{s}_2}(m_2, e_2) \to \mathsf{LWE}_{\mathbf{s}_1 \otimes \mathbf{s}_2}(m_1 \wedge m_2, e_1 \cdot e_2)$$

$\Rightarrow$ FHE boostrapping requires **strong Refreshing** :

$$\mathsf{LWE}_{\mathbf{s}}(m, e) \to \mathsf{LWE}_{\mathbf{s}}(m, e'), \qquad e' \ll e.$$

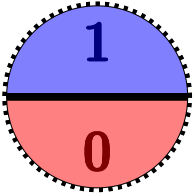
Techniques: Key Switching, Mod Switching, and Homomorphic Decryption.

LWE encryption with different message spaces

Idea: use an LWE sample as a mask

$$\mathsf{Enc}_{\mathbf{s}}(m) = (\mathbf{a}, b = \langle \mathbf{a}\,,\, \mathbf{s} \rangle + e + m \cdot q/2)$$
$$\mathsf{Dec}_{\mathbf{s}}(\mathbf{a}, b) = \lfloor 2(b - \langle \mathbf{a}\,,\, \mathbf{s} \rangle)/q \rceil$$

$m \cdot \frac{q}{2} + e$



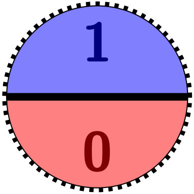
binary messages
$\mathsf{LWE}_{\mathbf{s}}^{2}(m, q/4)$

LWE encryption with different message spaces

Idea: use an LWE sample as a mask

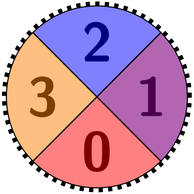
$$\mathsf{Enc}_{\mathbf{s}}(m) = (\mathbf{a}, b = \langle \mathbf{a}\,,\, \mathbf{s} \rangle + e + m \cdot q/2)$$
$$\mathsf{Dec}_{\mathbf{s}}(\mathbf{a}, b) = \lfloor 2(b - \langle \mathbf{a}\,,\, \mathbf{s} \rangle)/q \rceil$$

$m \cdot \frac{q}{2} + e$



binary messages
$\mathsf{LWE}^2_{\mathbf{s}}(m, q/4)$

$m \cdot \frac{q}{4} + e$

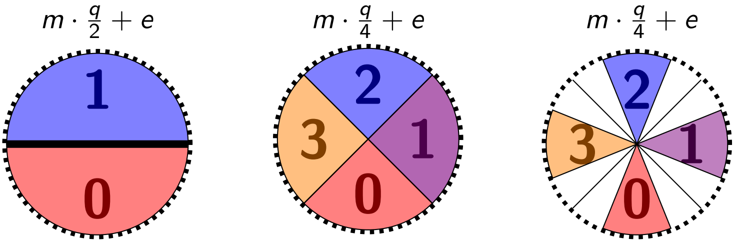


Messages in $\mathbb{Z}_4$
$\mathsf{LWE}^4_{\mathbf{s}}(m, q/8)$

LWE encryption with different message spaces

Idea: use an LWE sample as a mask

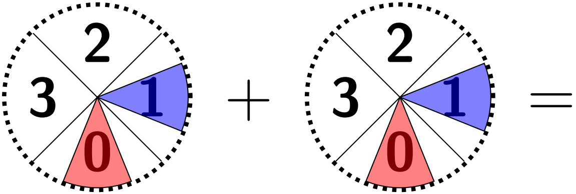
$$\mathsf{Enc}_{\mathbf{s}}(m) = (\mathbf{a}, b = \langle \mathbf{a}\,,\, \mathbf{s}\rangle + e + m \cdot q/2)$$
$$\mathsf{Dec}_{\mathbf{s}}(\mathbf{a}, b) = \lfloor 2(b - \langle \mathbf{a}\,,\, \mathbf{s}\rangle)/q \rceil$$



binary messages	Messages in $\mathbb{Z}_4$	Smaller error
$\mathsf{LWE}^2_{\mathbf{s}}(m, q/4)$	$\mathsf{LWE}^4_{\mathbf{s}}(m, q/8)$	$\mathsf{LWE}^4_{\mathbf{s}}(m, q/16)$

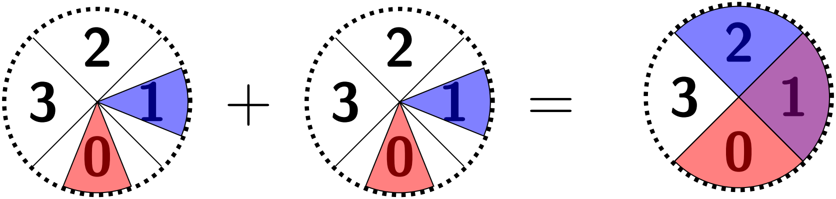
A Cheap NAND gate

Idea, use: $m_1 \wedge m_2 \Leftrightarrow m_1 + m_2 = \mathbf{2} \bmod 4$.
Consider binary messages $\{\mathbf{0}, \mathbf{1}\}$ encrypted with $t = 4$:



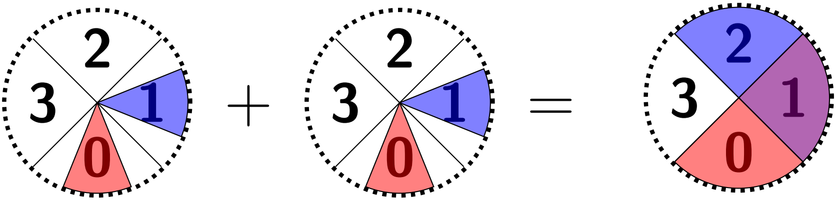
A Cheap NAND gate

Idea, use: $m_1 \wedge m_2 \Leftrightarrow m_1 + m_2 = \mathbf{2} \bmod 4$.
Consider binary messages $\{\mathbf{0}, \mathbf{1}\}$ encrypted with $t = 4$:



A Cheap NAND gate

Idea, use: $m_1 \wedge m_2 \Leftrightarrow m_1 + m_2 = \mathbf{2} \bmod 4$.
Consider binary messages $\{\mathbf{0}, \mathbf{1}\}$ encrypted with $t = 4$:

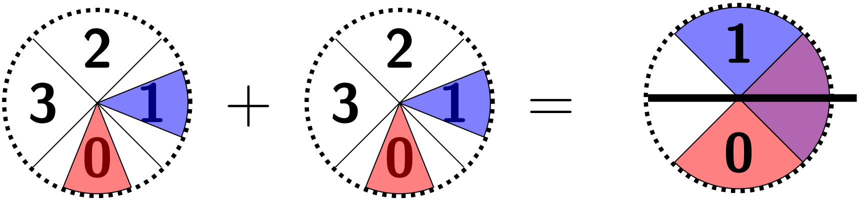


Consider it as a ciphertext for $t = 2$ and rotate.

A Cheap NAND gate

Idea, use: $m_1 \wedge m_2 \Leftrightarrow m_1 + m_2 = \mathbf{2} \bmod 4$.

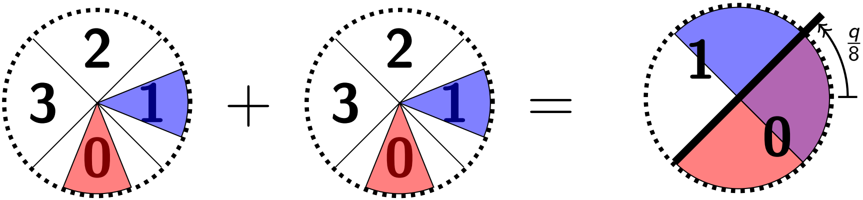
Consider binary messages $\{\mathbf{0}, \mathbf{1}\}$ encrypted with $t = 4$:



Consider it as a ciphertext for $t = 2$ and rotate.

A Cheap NAND gate

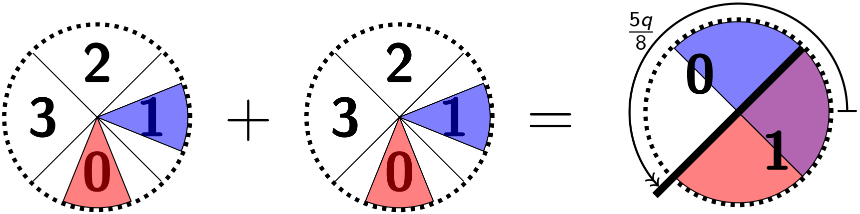
Idea, use: $m_1 \wedge m_2 \Leftrightarrow m_1 + m_2 = 2 \bmod 4$.
Consider binary messages $\{\mathbf{0}, \mathbf{1}\}$ encrypted with $t = 4$:



Consider it as a ciphertext for $t = 2$ and rotate.

A Cheap NAND gate

Idea, use: $m_1 \wedge m_2 \Leftrightarrow m_1 + m_2 = 2 \bmod 4$.
Consider binary messages $\{0, 1\}$ encrypted with $t = 4$:



Consider it as a ciphertext for $t = 2$ and rotate.

HomNAND:

$$\mathrm{LWE}^4_{\mathbf{s}}(m_1, \tfrac{q}{16}) \times \mathrm{LWE}^4_{\mathbf{s}}(m_2, \tfrac{q}{16}) \rightarrow \mathrm{LWE}^2_{\mathbf{s}}(m_1 \bar{\wedge} m_2, \tfrac{q}{4})$$

$$(\mathbf{a}_1, b_1) \quad , \quad (\mathbf{a}_2, b_2) \mapsto (\mathbf{a}_1 + \mathbf{a}_2, b_1 + b_2 + \tfrac{5q}{8})$$

Lightweight Refreshing

We have **HomNAND**:

$$\mathsf{LWE}^4_{\mathbf{s}}\left(m_1, \frac{q}{16}\right) \times \mathsf{LWE}^4_{\mathbf{s}}\left(m_2, \frac{q}{16}\right) \to \mathsf{LWE}^2_{\mathbf{s}}\left(m_1 \bar{\wedge} m_2, \frac{q}{4}\right)$$

Lightweight Refreshing

We have **HomNAND**:

$$\mathsf{LWE}^4_{\mathbf{s}}\left(m_1, \frac{q}{16}\right) \times \mathsf{LWE}^4_{\mathbf{s}}\left(m_2, \frac{q}{16}\right) \to \mathsf{LWE}^2_{\mathbf{s}}\left(m_1 \bar{\wedge}\, m_2, \frac{q}{4}\right)$$

To build an FHE we require a relaxed function **LightRefresh**:

$$\textbf{LightRefresh} : \mathsf{LWE}^2_{\mathbf{s}}(m, q/4) \to \mathsf{LWE}^4_{\mathbf{s}}(m, q/16)$$

Lightweight Refreshing

We have **HomNAND**:

$$\mathsf{LWE}^4_{\mathbf{s}}\left(m_1, \frac{q}{16}\right) \times \mathsf{LWE}^4_{\mathbf{s}}\left(m_2, \frac{q}{16}\right) \to \mathsf{LWE}^2_{\mathbf{s}}\left(m_1 \bar{\wedge}\ m_2, \frac{q}{4}\right)$$

To build an FHE we require a relaxed function **LightRefresh**:

$$\textbf{LightRefresh} : \mathsf{LWE}^2_{\mathbf{s}}(m, q/4) \to \mathsf{LWE}^4_{\mathbf{s}}(m, q/16)$$

whereas previous works required:

$$\textbf{Refresh} : \mathsf{LWE}^2_{\mathbf{s}}(m, q/4) \to \mathsf{LWE}^2_{\mathbf{s}}(m, E), E \ll q.$$

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

Outline

Decryption using an Accumulator

$$\mathrm{Dec}_{\mathbf{s}}(\mathbf{a}, b) = \mathrm{msb}\left(b - \langle \mathbf{a}\,,\,\mathbf{s}\rangle \bmod q\right) = \mathrm{msb}\left(b - \sum_i a_i \cdot s_i \bmod q\right)$$

$\mathrm{Dec}_{\mathbf{s}}(\mathbf{a}, b)$:

$acc \leftarrow b$
for $i = 1$ to n:
  $acc \leftarrow acc - a_i \cdot s_i \bmod q$
Return $\mathrm{msb}(acc)$

Decryption using an Accumulator

$$\mathrm{Dec}_{\mathbf{s}}(\mathbf{a}, b) = \mathrm{msb}\left(b - \langle \mathbf{a}\,,\, \mathbf{s}\rangle \bmod q\right) = \mathrm{msb}\left(b - \sum_i a_i \cdot s_i \bmod q\right)$$

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

$E(acc) \leftarrow b$
for $i = 1$ to n:
    $E(acc) \leftarrow E(acc) - E(a_i \cdot s_i) \bmod q$
Return LWE(msb(acc))

$ACC = E(acc)$ holds an encrypted integer $acc \in \mathbb{Z}_q$
The accumulator ACC should support the following operations:

Decryption using an Accumulator

$$\mathsf{Dec}_{\mathbf{s}}(\mathbf{a}, b) = \mathsf{msb}\,(b - \langle \mathbf{a}\,,\,\mathbf{s}\rangle \bmod q) = \mathsf{msb}\left(b - \sum_i a_i \cdot s_i \bmod q\right)$$

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

$E(acc) \leftarrow b$
for $i = 1$ to n:
    $E(acc) \leftarrow E(acc) - E(a_i \cdot s_i) \bmod q$
Return LWE(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: $ACC \rightarrow LWE(msb(acc))$

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

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 $a_i = \delta_{i=acc} \in \{0,1\}$
- 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 $E(X^{acc})$.

Ring version of [GSW13]

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

$$\mathsf{E}_{\mathbf{s}}(m) = [\mathbf{A}, \mathbf{As} + \mathbf{e}] + m \cdot \mathbf{G}$$

$\mathsf{Dec}_{\mathbf{s}}$: extract an $\mathsf{LWE}_{\mathbf{s}}$ ciphertext (last row) and decrypt.
Supports Add. and Mult. for **small messages** $m \in \{-1, 0, 1\}$.

Ring version of [GSW13]

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

$$\mathsf{E}_{\mathbf{s}}(m) = [\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e}] + m \cdot \mathbf{G}$$

$\mathsf{Dec}_{\mathbf{s}}$: extract an $\mathsf{LWE}_{\mathbf{s}}$ 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: $\mathbf{G} = u \cdot [\mathbf{I}, b\mathbf{I}, b^2\mathbf{I} \ldots b^{k-1}\mathbf{I}]^t \in \mathcal{R}_Q^{2 \times 2k}$.

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

Supports addition for all message $m \in \mathbb{Z}_q$.

The group of roots of unity and msb

$$
\begin{array}{c|cccc|cccc}
m & 0 & 1 & \dots & \frac{q}{2}-1 & \frac{q}{2} & \frac{q}{2}+1 & \dots & q-1 \\
X^m & 1 & X & \dots & X^{N-1} & -1 & -X & \dots & -X^{N-1}
\end{array}
$$

Take the vector representation of X^m

$$
\begin{array}{c|cccc|cccc}
\mathbf{x}_m & \begin{bmatrix}1\\0\\\vdots\\0\end{bmatrix} & \begin{bmatrix}0\\1\\\vdots\\0\end{bmatrix} & \ddots & \begin{bmatrix}0\\0\\\vdots\\1\end{bmatrix} & \begin{bmatrix}-1\\0\\\vdots\\0\end{bmatrix} & \begin{bmatrix}0\\-1\\\vdots\\0\end{bmatrix} & \ddots & \begin{bmatrix}0\\0\\\vdots\\-1\end{bmatrix}
\end{array}
$$

Sum all the coordinates

$$
\begin{array}{c|cccc|cccc}
\langle \mathbf{1}\,,\,\mathbf{x}_m\rangle & 1 & 1 & \dots & 1 & -1 & -1 & \dots & -1
\end{array}
$$

$$
\frac{\langle \mathbf{1}\,,\,\mathbf{x}_m\rangle + 1}{2} = \frac{(-1)^{\mathrm{msb}(m)} + 1}{2} = \mathrm{msb}(m).
$$

Extracting $\mathrm{LWE}^4_{\mathbf{s}}(\mathrm{msb}(m))$

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

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

Extracting $\mathrm{LWE}^4_{\mathbf{s}}(\mathrm{msb}(m))$

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

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

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

$$\mathbf{C} = \left[\mathbf{A}', \mathbf{A}' \cdot \mathbf{s} + \mathbf{e} + \frac{Q}{8} \cdot \mathbf{x}_m\right]$$

Extracting $\mathsf{LWE}^4_{\mathbf{s}}(\mathsf{msb}(m))$

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

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

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

$$\mathbf{C} = \left[\mathbf{A}', \mathbf{A}' \cdot \mathbf{s} + \mathbf{e} + \frac{Q}{8} \cdot \mathbf{x}_m\right]$$

Sum all rows and add $q/8$:

$$\begin{aligned} \mathbf{1}^t \cdot \mathbf{C} = \mathbf{1}^t \cdot & \left[\mathbf{A}', \mathbf{A}' \cdot \mathbf{s} + \mathbf{e} + \frac{Q}{8} \cdot \mathbf{x}_m + \frac{Q}{8}\right] \\ = \quad & \left[\mathbf{a}', \mathbf{a}' \cdot \mathbf{s} + e' + \frac{Q}{4} \cdot \mathsf{msb}(m)\right] \end{aligned}$$

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

Improvements

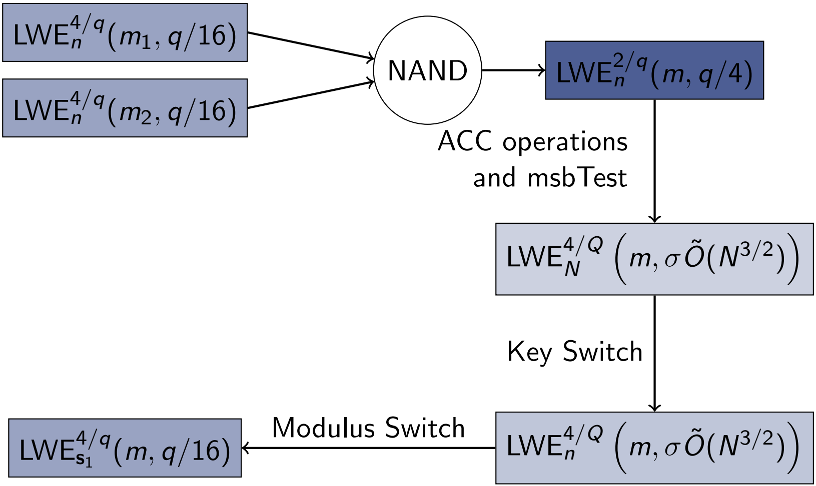
Improvement over the bootstrapping of [AP14]:

- 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

The ciphertext cycle



Parameter Proposal

Parameters.

LWE parameters:	$n = 410$	$q = 512.$
Ring-GSW parameters:	$N = 1024$	$Q = 2^{32}$.
Gadget Matrix:	$Q/8 \cdot [\mathbf{I}, 2^{11} \cdot \mathbf{I}, 2^{22} \cdot \mathbf{I}]$	

Key Size.

$$\left.\begin{array}{ll} \text{Bootstrapping Key Size} & 846\text{ MB} \\ \text{Key Switching Key Size:} & +135\text{ MB} \end{array}\right\} \leqslant 1\text{GB}$$

Running time.

Per NAND gate: $39,360$ FFTs ≈ 0.4 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: $\leqslant$ 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