Toward Automated Analysis and Prototyping of Homomorphic Encryption Schemes

Cyrielle FERON
Loïc LAGADEC
Vianney LAPOTRE
Plan

I – Introduction

II – PAnTHErS modeling

III – PAnTHErS analysis

IV – PAnTHErS usage

V – Results

VI – Conclusion & future work
I – Introduction

1. Homomorphic Encryption (HE)
2. Goals
Introduction – *Homomorphic Encryption (HE)*

Homomorphic Encryption principle
Introduction – *Homomorphic Encryption (HE)*

**Homomorphic Encryption principle**

**Advantages:**

- No decryption in the Cloud.
- Data are secured during the whole process (transfers and computations).
Introduction – Homomorphic Encryption (HE)

Homomorphic Encryption principle

Advantages:
• No decryption in the Cloud.
• Data are secured during the whole process (transfers and computations).

Challenges:
• Active research area ⇒ many HE schemes.
• Important complexity and memory consumption.
• Significant expansion factor.
Introduction – Goals

Develop a Prototyping and Analysis Tool for Homomorphic Encryption Schemes (PAnTHErS) to help improving research on HE schemes.
II – PAnTHERsS modeling

1. Atomic and Specific functions
2. HE basic functions
3. Interest
Atomic functions
(= one operation)

Examples:

**mult**
- Inputs: \(a, b\)
- \(c = a \times b\)
- Output: \(c\)

**add**
- Inputs: \(a, b\)
- \(c = a + b\)
- Output: \(c\)

**mod**
- Inputs: \(a, b\)
- \(c = a \mod b\)
- Output: \(c\)

**rand**
- Inputs: \(R, a, b\)
- \(c \leftarrow R^{a \times b}\)
- Output: \(c\)
### Atomic functions

**mult**

- **Inputs**: $a, b$
- **Output**: $c = a \times b$

**add**

- **Inputs**: $a, b$
- **Output**: $c = a + b$

**mod**

- **Inputs**: $a, b$
- **Output**: $c = a \% b$

### Specific functions

**Specific functions**

**Examples:**

**distriLWE**

- **Inputs**: $q, n, m, k$ : integers
- $s$ : vector of size $n$

**Example:**

\[
A = \text{rand}(R_q, m, n) \\
e = \text{rand}(\chi, m, 1) \\
e = \text{mult}(k, e) \\
b = \text{mult}(A, s) \\
b = \text{add}(e, b) \\
b = \text{mod}(b, q) \\
\text{Outputs}: b, A
\]
HE basic functions:

• Series of Atomic and/or Specific functions
• 5 HE basics: KeyGen, Enc, Dec, Add, Mult
PAnTHERs modeling – *HE basic functions*

**HE basic functions:**
- Series of Atomic and/or Specific functions
- 5 HE basics: KeyGen, Enc, Dec, Add, Mult

*Example:*

**FV**
- KeyGen
- Enc
- Dec
- Add
- Mult

**Dec**
- Inputs: $c$ : tuple of polynomials  
  $sk$ : secret key  
  $q, t$ : integers
- $m = \text{AddTimes}(c[0], c[1], sk)$
- $m = \text{mod}(m, q)$
- $m = \text{ChangeMod}(m, q, t)$
- Output: $m$
PAnTHErS modeling – *Interest*

SHIELD

GSW

YASHE

FV

F-NTRU
PAnTHERs modeling – *Interest*

Diagram showing the relationship between SHIELD, GSW, YASHE, FV, and F-NTRU.
PAnTHERs modeling – *Interest*

- **SHIELD**
- **GSW**
- **YASHE**
- **FV**

- **F-NTRU**
PAnTHERs modeling – *Interest*

SHIELD

GSW

YASHE

FV

F-NTRU
PAnTHERs modeling – *Interest*

- SHIELD
- GSW
- YASHE
- FV

F-NTRU
PAnTHERs modeling – *Interest*

- SHIELD
- GSW
- YASHE
- FV

- F-NTRU
PAnTHERs modeling – Interest

SHEILD

GSW

YASHE

FV

F-NTRU
III – PAnTHErS analysis

1. Complexity and memory cost
2. Atomics analysis
3. Specific and HE basic analysis
4. HE scheme analysis
PAnTHErS analysis – *Complexity and memory cost*

**Complexity**: table containing number of operations performed.

*Example:*

<table>
<thead>
<tr>
<th></th>
<th>MULT</th>
<th>ADD</th>
<th>DIV</th>
<th>MOD</th>
<th>RAND</th>
<th>ROUND</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT</td>
<td>$m \times d$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>POLY</td>
<td>$m \times n$</td>
<td>$m \times n$</td>
<td>0</td>
<td>$m$</td>
<td>$(n + 1).m$</td>
<td>0</td>
</tr>
</tbody>
</table>

**Memory**: table containing characteristics of parameters created.

*Example:*

<table>
<thead>
<tr>
<th>Name</th>
<th>Rows</th>
<th>Cols</th>
<th>Type</th>
<th>Degree</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>$n$</td>
<td>$m$</td>
<td>POLY</td>
<td>2048</td>
</tr>
<tr>
<td>c</td>
<td>$m$</td>
<td>1</td>
<td>POLY</td>
<td>2048</td>
</tr>
<tr>
<td>b</td>
<td>$m$</td>
<td>1</td>
<td>POLY</td>
<td>2048</td>
</tr>
</tbody>
</table>
Atomic functions

Example:

\[ c = a + b \]

Inputs: \( a, b \)

Output: \( c \)

Complexity.add

Inputs: \( a, b \)

Complexity[type(a)][ADD] += a.rows * a.cols

Output: \( c \)

Memory.add

Inputs: \( a, b \)

Memory += [c, a.type, a.rows, a.cols, a.degree]

Output: \( c \)
Specific functions

Example:

\textbf{distrILWE}

\begin{align*}
\text{Inputs} &: \quad q, n, m, k : \text{integers} \\
&\quad s : \text{vector of size } n
\end{align*}

\begin{align*}
A &= \text{rand}(R_q, m, n) \\
e &= \text{rand}(\chi, m, 1) \\
e &= \text{mult}(k, e) \\
b &= \text{mult}(A, s) \\
b &= \text{add}(e, b) \\
b &= \text{mod}(b, q)
\end{align*}

\text{Outputs} : \quad b, A
Specific functions

Example:

**distriLWE**

Inputs: $q, n, m, k : \text{integers}$
$s : \text{vector of size } n$

$A = \text{rand}(R_q, m, n)$
$e = \text{rand}(\chi, m, 1)$
$e = \text{mult}(k, e)$
$b = \text{mult}(A, s)$
$b = \text{add}(e, b)$
$b = \text{mod}(b, q)$

Outputs: $b, A$

**Complexity.distriLWE**

Inputs: $q, n, m, k : \text{integers}$
$s : \text{vector of size } n$

$A = \text{Complexity.rand}(R_q, m, n)$
$e = \text{Complexity.rand}(\chi, m, 1)$
$e = \text{Complexity.mult}(k, e)$
$b = \text{Complexity.mult}(A, s)$
$b = \text{Complexity.add}(e, b)$
$b = \text{Complexity.mod}(b, q)$

Outputs: $b, A$
HE basic functions

Example:

\textbf{FVDec}

\begin{itemize}
  \item \textit{Inputs}: \(c\) : tuple of polynomials
  \item \(sk\) : secret key
\end{itemize}

\begin{itemize}
  \item \(m = \text{AddTimes}(c[0], c[1], sk)\)
  \item \(m = \text{mod}(m, q)\)
  \item \(m = \text{ChangeMod}(m, q, t)\)
\end{itemize}

\textit{Output}: \(m\)
HE basic functions

Example:

\[
\begin{align*}
FVDec & : \\
\text{Inputs: } & c : \text{tuple of polynomials} \\
& sk : \text{secret key} \\
\text{Output: } & m \\
\end{align*}
\]

\[
\begin{align*}
m & = \text{AddTimes}(c[0], c[1], sk) \\
m & = \text{mod}(m, q) \\
m & = \text{ChangeMod}(m, q, t)
\end{align*}
\]

\[
\begin{align*}
\text{Memory. } FVDec & : \\
\text{Inputs: } & c : \text{tuple of polynomials} \\
& sk : \text{secret key} \\
\text{Output: } & m \\
\end{align*}
\]

\[
\begin{align*}
m & = \text{Memory. AddTimes}(c[0], c[1], sk) \\
m & = \text{Memory. mod}(m, q) \\
m & = \text{Memory. ChangeMod}(m, q, t)
\end{align*}
\]
 Execution of a HE scheme

Input parameters

HE Scheme

Keys, ciphertexts, plaintexts.
PAnTHERS analysis – *HE scheme analysis*

**Execution of a HE scheme**
- Input parameters
- **HE Scheme**
  - Keys, ciphertexts, plaintexts.

**Execution of HE scheme analysis**
- Input parameters
- **HE Scheme**
  - Complexity, memory cost.
IV – PAnTHERS usage
PAnTHERs usage

[Diagram showing PAnTHERs usage with connections to Atomics, functions, Analyses, Complexity, Memory, and HE Expert]
PAnTHErS usage
PAnTHERs usage
PAnTHERs usage
PAnTHERS usage
PAnTHERS usage
PAntHERS usage

Diagram showing the flow of data from Atomics, Specifics, HE Schemes, Functions, XML, XMI, HE Basics, and Analyses, leading to Parameters and a HE Expert.
PAnTHErS usage

Information feedback (csv)
PAnTHErS usage

Information feedback (csv)
V – Results

1. FV and YASHE complexity
2. FV and YASHE memory cost
3. FV and YASHE depth
Results – *FV and YASHE complexity*

\[ R = \mathbb{Z}[x]/\Phi_d(x), \ n = \varphi(d) \]
\[ R_q = R/qR \]

Integer base \( w \)
Plaintext space: \( R_t = R/tR \)

<table>
<thead>
<tr>
<th>( n )</th>
<th>1024</th>
<th>2048</th>
<th>4096</th>
<th>8192</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \max. \log_2(q) )</td>
<td>46</td>
<td>88</td>
<td>174</td>
<td>348</td>
</tr>
</tbody>
</table>

Results – *FV and YASHE complexity*

\[ R = \mathbb{Z}[x]/\Phi_d(x), \ n = \varphi(d) \]
\[ R_q = R/qR \]

**Fixed parameter**: \( \log_2(w) = 32 \)

**Plaintext space**: \( R_t = R/tR \)

<table>
<thead>
<tr>
<th>( n )</th>
<th>1024</th>
<th>2048</th>
<th>4096</th>
<th>8192</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{max. } \log_2(q) )</td>
<td>46</td>
<td>88</td>
<td>174</td>
<td>348</td>
</tr>
</tbody>
</table>

\[ R = \mathbb{Z}[x]/\Phi_d(x), \ n = \varphi(d) \]
\[ R_q = R/qR \]

**Fixed parameter**: \( \log_2(w) = 32 \)

**Plaintext space**: \( R_t = R/tR \)

Here, \( \omega = \log_2(w) \)
Results – *FV* and *YASHE* memory cost

\[ R = \mathbb{Z}[x]/\Phi_d(x), \ n = \varphi(d) \]
\[ R_q = R/qR \]

Integer base \( w \)

Plaintext space: \( R_t = R/tR \)

<table>
<thead>
<tr>
<th>( n )</th>
<th>1024</th>
<th>2048</th>
<th>4096</th>
<th>8192</th>
</tr>
</thead>
<tbody>
<tr>
<td>max. ( \log_2(q) )</td>
<td>46</td>
<td>88</td>
<td>174</td>
<td>348</td>
</tr>
</tbody>
</table>

Here, \( \omega = \log_2(w) \)

---

**KeyGen function**

**Mult function**
Results – *FV and YASHE depth*

\[
R = \mathbb{Z}[x]/\Phi_d(x), \quad n = \varphi(d)
\]

\[
R_q = R/qR
\]

Integer base \( w \)

Plaintext space : \( R_t = R/tR \)

\[
\log_2(\omega) = 32
\]

<table>
<thead>
<tr>
<th>( n )</th>
<th>1024</th>
<th>2048</th>
<th>4096</th>
<th>8192</th>
</tr>
</thead>
<tbody>
<tr>
<td>max. ( \log_2(q) )</td>
<td>46</td>
<td>88</td>
<td>174</td>
<td>348</td>
</tr>
</tbody>
</table>

*Fixed parameter: \( \log_2(w) = 32 \)*

*Here, \( \omega = \log_2(w) \)*
VI – Conclusion & future work
HE scheme modeling:
5 HE basic functions composed of Atomic and Specific functions.

\[ \text{Inputs: } c : \text{tuple of polynomials} \]
\[ sk : \text{secret key} \]
\[ q, t : \text{integers} \]

\[ m = \text{AddTimes}(c[0], c[1], sk) \]
\[ m = \text{mod}(m, q) \]
\[ m = \text{ChangeMod}(m, q, t) \]

Output: \[ m \]
HE scheme modeling: 5 HE basic functions composed of Atomic and Specific functions.

Filling the library implies faster modeling for new HE schemes.
Atomic, specific & HE basic functions analysis:
Atomic, specific & HE basic functions are linked to their complexity and memory cost analysis functions.

Complexity.add
- Inputs: a, b
- Complexity[type(a)][ADD] += a.rows * a.cols
- Output: c

Memory.add
- Inputs: a, b
- Memory += [c, a.type, a.rows, a.cols, a.degree]
- Output: c
Conclusion

Atomic, specific & HE basic functions analysis:
Atomic, specific & HE basic functions are linked to their complexity and memory cost analysis functions.

Automated generation of HE scheme analysis functions.
By *varying input parameters*, PAnTHErS provides memory, complexity and depth *results.*
By varying input parameters, PAnTHErS provides memory, complexity and depth results.

HE expert interprets PAnTHErS data...
By varying input parameters, PAnTHErS provides memory, complexity and depth results.

HE expert interprets PAnTHErS data...
...to select the scheme which fits better his application.
Future work
Future work
Future work
Thanks

Cyrielle FERON
Loïc LAGADEC
Vianney LAPOTRE

References: