Some security bounds for the DGHV scheme

F. Marinelli, R. Aragona, C. Marcolla, M. Sala

University of Trento

12 June 2014
Homomorphic encryption scheme

- traditional encryption scheme:
  \[ E = (\text{KeyGen}, \text{Encrypt}, \text{Decrypt}) \]

- homomorphic encryption scheme:
  \[ E = (\text{KeyGen}, \text{Encrypt}, \text{Decrypt}, \text{Evaluate}) \]

\[
\begin{array}{c}
m_1, \ldots, m_t \quad \xrightarrow{f} \quad f(m_1, \ldots, m_t) \\
\uparrow \quad \quad \uparrow \\
c_1, \ldots, c_t \quad \xrightarrow{} \quad c_f
\end{array}
\]
- traditional encryption scheme:
  \[ \mathcal{E} = (\text{KeyGen, Encrypt, Decrypt}) \]

- homomorphic encryption scheme:
  \[ \mathcal{E} = (\text{KeyGen, Encrypt, Decrypt, Evaluate}) \]

\[
\begin{array}{cccc}
m_1, \ldots, m_t & \stackrel{f}{\longrightarrow} & f(m_1, \ldots, m_t) \\
\uparrow & & \uparrow \\
c_1, \ldots, c_t & \longrightarrow & c_f
\end{array}
\]
1978 Rivest, Adleman and Dertouzos: first idea of homomorphic encryption with respect to both operations

- Partially homomorphic encryption:
  - RSA and El Gamal: homomorphic with respect to multiplication,
  - Goldwasser-Micali: homomorphic with respect to addition.

- 2009 Gentry: Fully homomorphic encryption
1978 Rivest, Adleman and Dertouzos: first idea of homomorphic encryption with respect to both operations

- Partially homomorphic encryption:
  - RSA and El Gamal: homomorphic with respect to multiplication,
  - Goldwasser-Micali: homomorphic with respect to addition.

2009 Gentry: Fully homomorphic encryption
1978 Rivest, Adleman and Dertouzos: first idea of homomorphic encryption with respect to both operations

Partially homomorphic encryption:
- RSA and El Gamal: homomorphic with respect to multiplication,
- Goldwasser-Micali: homomorphic with respect to addition.

2009 Gentry: Fully homomorphic encryption
Fully homomorphic encryption (FHE) scheme:

- is homomorphic with respect to both operations,
- performs an arbitrary number of operations,
- its evaluation algorithm outputs a compact ciphertext.

**Compactness:**
Given $\lambda$ the security parameter, there exists a polynomial $s = s(\lambda)$ such that the output length of the Evaluate algorithm is at most $s$ bits long.
Somewhat homomorphic encryption (SHE) scheme:

- is homomorphic with respect to both operations,
- performs a limited number of operations,
- the output of its evaluation algorithm is *NOT compact*.

Why does it perform only some operations? Each operation adds noise to the ciphertext, when the noise grows too much it becomes impossible to decrypt it.
Homomorphic encryption scheme

SHE $\xrightarrow{\text{squashing}}$ bootstrapping $\rightarrow$ FHE

- squashing: strategy for the reduction of ciphertext length
- bootstrapping: strategy for noise reduction
Three main families of SHE schemes are known:

- Gentry’s original scheme on ideal lattices,
- van Dijk, Gentry, Halevi and Vaikuntanathan’s (DGHV) scheme over the integers,
- Brakerski and Vaikuntanathan’s (BV) scheme based on the Learning with Errors (LWE) problem.
\( \mathcal{E} = (\text{KeyGen, Encrypt, Decrypt, Evaluate}) \)

**PARAMETERS:** (all depending on the security parameter \( \lambda \))

- \( \gamma \) is the bit-length of the integers in the public key.
- \( \eta \) is the bit-length of the secret key.
- \( \tau \) is the number of integers in the public key.
- \( \rho \) is the bit-length of the noise in \( \text{KeyGen} \).
- \( \rho' \) is the bit-length of the noise in \( \text{Encrypt} \).

For a specific \( p \in (2\mathbb{Z} + 1) \cap [2^{\eta-1}, 2^{\eta}) \) we use the following uniform distribution over integers:

\[
\mathcal{D}_{\gamma,\rho}(p) = \{ x = pq + r : q \xleftarrow{\$} \mathbb{Z} \cap [0, 2^{\gamma}/p), r \xleftarrow{\$} \mathbb{Z} \cap (-2^{\rho}, 2^{\rho}) \}
\]
SHE: DGHV scheme

$\mathcal{E} = (\text{KeyGen}, \text{Encrypt}, \text{Decrypt}, \text{Evaluate})$

**PARAMETERS:** (all depending on the security parameter $\lambda$)

- $\gamma$ is the bit-length of the integers in the public key.
- $\eta$ is the bit-length of the secret key.
- $\tau$ is the number of integers in the public key.
- $\rho$ is the bit-length of the noise in KeyGen.
- $\rho'$ is the bit-length of the noise in Encrypt.

For a specific $p \in (2\mathbb{Z} + 1) \cap [2^{\eta-1}, 2^\eta)$ we use the following uniform distribution over integers:

$$\mathcal{D}_{\gamma,\rho}(p) = \{x = pq+r : q \leftarrow \mathbb{Z} \cap [0, 2^{\gamma}/p), \ r \leftarrow \mathbb{Z} \cap (-2^{\rho}, 2^{\rho})\}$$
SHE: DGHV scheme

\[ \mathcal{E} = (\text{KeyGen}, \text{Encrypt}, \text{Decrypt}, \text{Evaluate}) \]

**PARAMETERS:** (all depending on the security parameter \( \lambda \))

- \( \gamma \) is the bit-length of the integers in the public key.
- \( \eta \) is the bit-length of the secret key.
- \( \tau \) is the number of integers in the public key.
- \( \rho \) is the bit-length of the noise in \( \text{KeyGen} \).
- \( \rho' \) is the bit-length of the noise in \( \text{Encrypt} \).

For a specific \( p \in (2\mathbb{Z} + 1) \cap [2^{\eta-1}, 2^{\eta}) \) we use the following uniform distribution over integers:

\[
D_{\gamma, \rho}(p) = \{ x = pq + r : q \xleftarrow{\$} \mathbb{Z} \cap [0, 2^\gamma/p), r \xleftarrow{\$} \mathbb{Z} \cap (-2^\rho, 2^\rho) \}
\]
\( \mathcal{E} = (\text{KeyGen}, \text{Encrypt}, \text{Decrypt}, \text{Evaluate}) \)

**KeyGen(\( \lambda \)) \rightarrow (sk, pk)\)**

- Sample \( p \leftarrow (2\mathbb{Z} + 1) \cap [2^{\eta - 1}, 2^\eta) \)
- For \( i = 0, \ldots, \tau \): sample \( x_i \leftarrow \mathcal{D}_{\gamma, \rho}(p) \), relabel so that \( x_0 \) is the largest. Restart until \( [x_0]_2 = 1 \) and \( [x_0]_p \) \( 2 \) = 0.
- Output: \( sk = p, pk = (x_0, x_1, \ldots, x_\tau) \).

**Encrypt(pk, m) \rightarrow c\)**

- Choose a random subset \( S \subseteq \{1, \ldots, \tau\} \).
- Choose a random \( r' \leftarrow (-2^\rho', 2^\rho') \).
- Output: \( c = [m + 2r' + 2 \sum_{i \in S} x_i]_{x_0} \).
\( E = (\text{KeyGen}, \text{Encrypt}, \text{Decrypt}, \text{Evaluate}) \)

**KeyGen(\( \lambda \)) \rightarrow (sk, pk)\)**

- Sample \( p \leftarrow (2\mathbb{Z} + 1) \cap [2^{\eta-1}, 2^\eta) \)
- For \( i = 0, \ldots, \tau \): sample \( x_i \leftarrow D_{\gamma,\rho}(p) \), relabel so that \( x_0 \) is the largest. Restart until \([x_0]_2 = 1\) and \([x_0]_p \) = 0.
- Output: \( sk = p, pk = (x_0, x_1, \ldots, x_\tau) \).

**Encrypt(pk, m) \rightarrow c**

- Choose a random subset \( S \subseteq \{1, \ldots, \tau\} \).
- Choose a random \( r' \leftarrow (-2^\rho', 2^\rho') \).
- Output: \( c = [m + 2r' + 2 \sum_{i \in S} x_i]_{x_0} \).
\[ \mathcal{E} = (\text{KeyGen}, \text{Encrypt}, \text{Decrypt}, \text{Evaluate}) \]

\textbf{KeyGen}(\lambda) \rightarrow (sk, pk)

- Sample \( p \leftarrow (2\mathbb{Z} + 1) \cap [2^{\eta-1}, 2^\eta) \)
- For \( i = 0, \ldots, \tau \): sample \( x_i \leftarrow D_{\gamma, \rho}(p) \), relabel so that \( x_0 \) is the largest. Restart until \( [x_0]_2 = 1 \) and \( [x_0]_p \) = 0.
- Output: \( sk = p, pk = (x_0, x_1, \ldots, x_\tau) \).

\textbf{Encrypt}(pk, m) \rightarrow c

- Choose a random subset \( S \subseteq \{1, \ldots, \tau\} \).
- Choose a random \( r' \leftarrow (-2^\rho', 2^\rho') \).
- Output: \( c = [m + 2r' + 2 \sum_{i \in S} x_i] x_0 \).
\( \mathcal{E} = (\text{KeyGen}, \text{Encrypt}, \text{Decrypt}, \text{Evaluate}) \)

**Decrypt** \((sk, c) \rightarrow m'\)
- Compute \(m' = \left[ [c]_p \right]_2 \).
- Output: \(m'\).

**Evaluate** \((pk, C, c_1, \ldots, c_t) \rightarrow c'\)
- Replace the XOR and the AND gates of \(C\) with addition and multiplication gates that operate over integers.
- Apply integer addition and integer multiplication gates to the ciphertexts.
- Output the resulting ciphertext \(c'\).
SHE: DGHV scheme

\[ E = (\text{KeyGen}, \text{Encrypt}, \text{Decrypt}, \text{Evaluate}) \]

**Decrypt(sk, c) \rightarrow m'**
- Compute \( m' = \left( \left[ c \right]_p \right)_2 \).
- Output: \( m' \).

**Evaluate(pk, C, c_1, ..., c_t) \rightarrow c'**
- Replace the XOR and the AND gates of \( C \) with addition and multiplication gates that operate over integers.
- Apply integer addition and integer multiplication gates to the ciphertexts.
- Output the resulting ciphertext \( c' \).
Let be:

- \((pk, sk) \leftarrow KeyGen(k)\),
- \(c \leftarrow Encrypt(pk, m)\).

The \text{Decrypt}(sk, c)\) is able to decrypt correctly \(c\), namely it outputs \(m' = m\), if \(\eta > \log_2(2^\rho' + \tau 2^\rho + 1) + 2\).

Where, we recall that:

- \(\eta\) is the bit-length of the secret key.
- \(\rho\) is the bit-length of the noise in \text{KeyGen}.
- \(\tau\) is the number of integers in the public key.
- \(\rho'\) is the bit-length of the noise in \text{Encrypt}.
Let be:

- \((pk, sk) \leftarrow KeyGen(\lambda)\),
- \(c_i \leftarrow Encrypt(pk, m_i)\), for \(i=1,\ldots,v\),
- \(c_a \leftarrow Evaluate(pk, C, c_1, \ldots, c_v)\).

The \(Decrypt(sk, c_a)\) is able to decrypt correctly \(c_a\), that is

\[
Decrypt(sk, c_a) = C(m_1, \ldots, m_v) = m_1 + \ldots + m_v \quad \text{if}
\]

\[
\eta > \log_2(2^{\rho'} + \tau 2^{\rho+1}) + 2 + \log_2(v)
\]

\[
\eta > bound(c_i) + \log_2(v)
\]
Let be:

- \((pk, sk) \leftarrow \text{KeyGen}(\lambda),\)
- \(c_i \leftarrow \text{Encrypt}(pk, m_i), \text{ for } i=1,\ldots,s,\)
- \(c_m \leftarrow \text{Evaluate}(pk, C, c_1, \ldots, c_s).\)

The \(\text{Decrypt}(sk, c_m)\) is able to decrypt correctly \(c_m\), that is:
\[
\text{Decrypt}(sk, c_m) = C(m_1, \ldots, m_s) = m_1 \cdot \ldots \cdot m_s \text{ if }
\]
\[
\eta > s(\log_2(2^{\rho'} + \tau 2^{\rho+1}) + 1) + 1
\]
\[
\eta > s(\text{bound}(c_i) - 1) + 1
\]
Let be:
- $C$ a binary circuit with $t$ inputs,
- $C'$ the associated integer circuit,
- $f(x_1, \ldots, x_t)$ the multivariate polynomial computed by $C'$,
- $d$ the degree of $f$.

If
\[ \eta \geq d \left[ \log_2(2^{\rho'} + \tau 2^{\rho+1}) + 1 \right] + 1 + \log |f|, \]

where $|f|$ is the sum of absolute values of the coefficients of $f$, then $\text{Decrypt}(sk, \text{Evaluate}(pk, C, c_1, \ldots, c_t)) = C(m_1, \ldots, m_t)$. 

F. Marinelli, R. Aragona, C. Marcolla, M. Sala Some security bounds for the DGHV scheme
SHE: DGHV scheme

general lemma

\[ \eta \geq d \left[ \log_2(2\rho' + \tau 2^\rho + 1) + 1 \right] + 1 + \log |f| \]

**Sketch of proof.**

- \(|a_0 + a_1 c + \ldots + a_d c^d| \leq |a_0 + a_1 + \ldots + a_d| \cdot |c^d| = |f| \cdot |c^d|,\)
- we want \(|f(c)| < p/2 \iff |f| \cdot |c^d| < p/2, \) where \(c = (m + 2r' + 2 \sum_{i \in S} x_i - kr_0)\)
- \(2^n > 2^{d+1}(2\rho' + \tau 2^\rho + 1)^d |f|,\)
- \(\eta > d \log_2(2\rho' + \tau 2^\rho + 1) + d + 1 + \log |f|.\)
bound given in the DGHV article:

\[ \eta \geq d(\rho' + 2) + 4 + \log |f|, \]

our bound:

\[ \eta \geq d \left[ \log_2(2^{\rho'} + \tau 2^{\rho'+1}) + 1 \right] + 1 + \log |f|. \]

For large \( \lambda \):

\[ \log_2(2^{\rho'} + \tau 2^{\rho'+1}) \approx \log_2(2^{\rho'}), \]

\[ \eta \geq d(\rho' + 1) + 1 + \log |f|, \]

if \( |f(c)| < p/8 \),

\[ \eta \geq d(\rho' + 1) + 4 + \log |f|. \]
**SHE: DGHV scheme**

**general lemma**

- **bound given in the DGHV article:**

  \[ \eta \geq d(\rho' + 2) + 4 + \log |f|, \]

- **our bound:**

  \[ \eta \geq d \left[ \log_2(2^{\rho'} + \tau 2^{\rho+1}) + 1 \right] + 1 + \log |f|. \]

- **For large \( \lambda \):**

  \[ \log_2(2^{\rho'} + \tau 2^{\rho+1}) \approx \log_2(2^{\rho'}), \]

  \[ \eta \geq d(\rho' + 1) + 1 + \log |f|, \]

- **if** \( |f(c)| < p/8 \), \( \eta \geq d(\rho' + 1) + 4 + \log |f|. \)
### SHE: DGHV Scheme Parameters

Some security bounds for the DGHV scheme

<table>
<thead>
<tr>
<th>Level of security</th>
<th>$\lambda$</th>
<th>$\rho$</th>
<th>$\rho'$</th>
<th>$\gamma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toy</td>
<td>32</td>
<td>32</td>
<td>64</td>
<td>33554432</td>
</tr>
<tr>
<td>Small</td>
<td>64</td>
<td>64</td>
<td>128</td>
<td>1073741824</td>
</tr>
<tr>
<td>Medium</td>
<td>80</td>
<td>80</td>
<td>160</td>
<td>3276800000</td>
</tr>
<tr>
<td>Large</td>
<td>128</td>
<td>128</td>
<td>256</td>
<td>34359738368</td>
</tr>
</tbody>
</table>
THANK YOU FOR YOUR ATTENTION!