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:

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

- homomorphic encryption scheme:

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

$$\begin{tikzcd}
m_1, \ldots, m_t \arrow[r, "f"] \arrow[d, leftrightarrow] & f(m_1, \ldots, m_t) \arrow[d, leftrightarrow] \\
c_1, \ldots, c_t \arrow[r] & c_f
\end{tikzcd}$$

Homomorphic encryption scheme

- traditional encryption scheme:

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

- homomorphic encryption scheme:

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

$$\begin{array}{ccc}
\hline
m_1, \ldots, m_t & \xrightarrow{\quad f \quad} & f(m_1, \ldots, m_t) \\
\updownarrow & & \updownarrow \\
c_1, \ldots, c_t & \textcolor{red}{\xrightarrow{\qquad\quad}} & c_f \\
\hline
\end{array}$$

Homomorphic encryption scheme

- 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

Homomorphic encryption scheme

- 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

Homomorphic encryption scheme

- 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

Homomorphic encryption scheme

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 λ 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.

Homomorphic encryption scheme

Somewhat homomorphic encryption (SHE) scheme:

- is homomorphic with respect to both operations,
- performs a limited number of operations,
- the output of its elvaluation 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

$$\text{SHE} \xrightarrow[\textit{bootstrapping}]{\textit{squashing}} \text{FHE}$$

- squashing: strategy for the reduction of ciphertext length
- bootstrapping: strategy for noise reduction

Homomorphic encryption scheme

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.

SHE: DGHV scheme

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

PARAMETERS: (all depending on the security parameter λ)

- γ is the bit-length of the integers in the public key.
- η is the bit-length of the secret key.
- τ is the number of integers in the public key.
- ρ is the bit-length of the noise in *KeyGen*.
- ρ' 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:

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

SHE: DGHV scheme

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

PARAMETERS: (all depending on the security parameter λ)

- γ is the bit-length of the integers in the public key.
- η is the bit-length of the secret key.
- τ is the number of integers in the public key.
- ρ is the bit-length of the noise in *KeyGen*.
- ρ' 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:

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

SHE: DGHV scheme

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

PARAMETERS: (all depending on the security parameter λ)

- γ is the bit-length of the integers in the public key.
- η is the bit-length of the secret key.
- τ is the number of integers in the public key.
- ρ is the bit-length of the noise in *KeyGen*.
- ρ' 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:

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

SHE: DGHV scheme

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

KeyGen(λ) $\longrightarrow$ (sk, pk)

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

Encrypt(pk, m) $\longrightarrow$ c

- Choose a random subset $S \subseteq \{1, ..., \tau\}$.
- Choose a random $r' \xleftarrow{\$} (-2^{\rho'}, 2^{\rho'})$.
- Output: $c = \left[m + 2r' + 2\sum_{i \in S} x_i\right]_{x_0}$.

SHE: DGHV scheme

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

KeyGen(λ) $\longrightarrow$ (sk, pk)

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

Encrypt(pk, m) $\longrightarrow$ c

- Choose a random subset $S \subseteq \{1, \ldots, \tau\}$.
- Choose a random $r' \xleftarrow{\$} (-2^{\rho'}, 2^{\rho'})$.
- Output: $c = \left[m + 2r' + 2\sum_{i \in S} x_i\right]_{x_0}$.

SHE: DGHV scheme

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

KeyGen(λ) $\longrightarrow$ (sk, pk)

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

Encrypt(pk, m) $\longrightarrow$ c

- Choose a random subset $S \subseteq \{1, ..., \tau\}$.
- Choose a random $r' \xleftarrow{\$} (-2^{\rho'}, 2^{\rho'})$.
- Output: $c = \left[m + 2r' + 2\sum_{i\in S} x_i\right]_{x_0}$.

SHE: DGHV scheme

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

Decrypt(sk, c) $\longrightarrow$ m'

- Compute $m' = \left[[c]_p \right]_2$.
- Output: m'.

Evaluate(pk, C, $c_1, ..., c_t$) $\longrightarrow$ 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

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

Decrypt(sk, c) $\longrightarrow$ m'

- Compute $m' = \left[[c]_p\right]_2$.
- Output: m'.

Evaluate(pk, C, $c_1, \ldots, c_t$) $\longrightarrow$ 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:

- $(\mathbf{pk}, sk) \longleftarrow KeyGen(k)$,
- $c \longleftarrow Encrypt(\mathbf{pk}, m)$.

The $Decrypt(sk, c)$ 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:
η is the bit-length of the secret key.
ρ is the bit-length of the noise in *KeyGen*.
τ is the number of integers in the public key.
ρ' is the bit-length of the noise in *Encrypt*.

Let be:

- $(\mathbf{pk}, sk) \longleftarrow KeyGen(\lambda)$,
- $c_i \longleftarrow Encrypt(\mathbf{pk}, m_i)$, for i=1,...,v,
- $c_a \longleftarrow Evaluate(\mathbf{pk}, C, c_1, ..., c_v)$.

The $Decrypt(sk, c_a)$ is able to decrypt correctly c_a, that is $Decrypt(sk, c_a) = C(m_1, ..., m_v) = m_1 + ... + m_v$ if

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

$$\eta > bound(c_i) + \log_2(v)$$

Let be:

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

The $Decrypt(sk, c_m)$ is able to decrypt correctly c_m, that is $Decrypt(sk, c_m) = C(m_1, ..., m_s) = m_1 \cdot ... \cdot m_s$ if

$$\eta > s(\log_2(2^{\rho'} + \tau 2^{\rho+1}) + 1) + 1$$

$$\eta > s(bound(c_i) - 1) + 1$$

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

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

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

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

Sketch of proof.

- $|a_0 + a_1 c + \ldots + a_d c^d| \leq |a_0 + a_1 + \ldots + a_d| \cdot |c^d| = |\mathbf{f}| \cdot |c^d|$,
- we want $|f(c)| < p/2 \Longrightarrow |\mathbf{f}||c^d| < p/2$, where $c = \left(m + 2r' + 2\sum_{i\in S} x_i - kr_0\right)$
- $2^\eta > 2^{d+1}(2^{\rho'} + \tau 2^{\rho+1})^d |\mathbf{f}|$,
- $\eta > d\log_2(2^{\rho'} + \tau 2^{\rho+1}) + d + 1 + \log|\mathbf{f}|$.

SHE: DGHV scheme

general lemma

- bound given in the DGHV article:

$$\eta \geq d(\rho' + 2) + 4 + \log |\mathbf{f}|,$$

- our bound:

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

- For large λ: $\log_2(2^{\rho'} + \tau 2^{\rho+1}) \approx \log_2(2^{\rho'})$,
- $\eta \geq d(\rho' + 1) + 1 + \log |\mathbf{f}|$,
- if $|f(c)| < p/8$, $\eta \geq d(\rho' + 1) + 4 + \log |\mathbf{f}|$.

- bound given in the DGHV article:

$$\eta \geq d(\rho' + 2) + 4 + \log|\mathbf{f}|,$$

- our bound:

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

- For large λ: $\log_2(2^{\rho'} + \tau 2^{\rho+1}) \approx \log_2(2^{\rho'})$,
- $\eta \geq d(\rho' + 1) + 1 + \log|\mathbf{f}|$,
- if $|f(c)| < p/8$, $\eta \geq d(\rho' + 1) + 4 + \log|\mathbf{f}|$.

SHE: DGHV scheme
parameters

Level of security	λ	ρ	ρ'	γ
Toy	32	32	64	33554432
Small	64	64	128	1073741824
Medium	80	80	160	3276800000
Large	128	128	256	34359738368

THANK YOU FOR YOUR ATTENTION!