Fully Homomorphic Encryption

Etienne Vareille

January 6, 2020

So far, we have seen:

- several PKE schemes (El Gamal, Dual Regev)
- LWE

Now, we are interested in the following question: can we delegate the computation of encrypted data to another machine? For instance, in the context of a high cost algorithms that we want to apply to some data, if the machine storing the data does not have sufficient computational power, we want to encrypt the data and make a distant server do the computation.

We want to preserve privacy. Ideally, we want to obtain the a scheme as presented in Figure 1 and Figure 2.

Definition 1 (HE)**.** An Homomorphic Encryption scheme for a class $\mathcal{C}$ of functions is a tuple (Gen, Enc, Dec, Eval) such that:

- (Gen, Enc, Dec) is a public-key encryption scheme.
- For all sequence of l messages $(m_1, ..., m_l) \in \{0,1\}^n$, for all function $f : \{0,1\}^n \to \{0,1\}$ in $\mathcal{C}$, we want

  $$\mathrm{Dec}(sk, \mathrm{Eval}(f, \mathrm{Enc}(pk, m_1), ..., \mathrm{Enc}(pk, m_l))) = f(m_1, ..., m_l)$$

  with overwhelming probability over pk, sk uniformly drawn by Gen, and over randomness of Enc.
- Security: standard IND-CPA security.

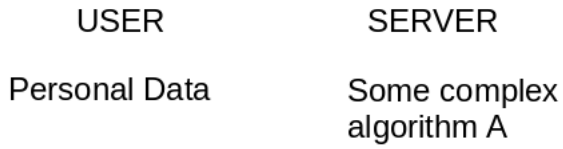

Figure 1: Situation of the data in the scheme

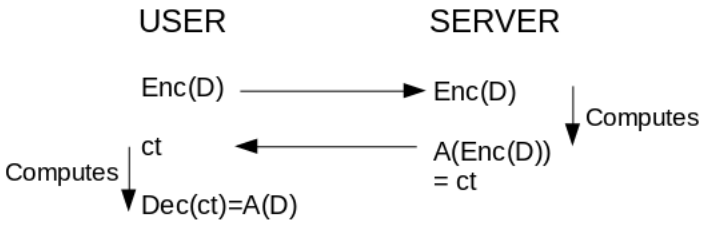

Figure 2: Communications in a FHE scheme

Remark. IND-CCA2 is not achievable. IND-CCA1 might be.

Definition 2. FHE An FHE scheme is a HE scheme for $\mathcal{C} = P/Poly$ the class of poly-size circuits.

1 Some HE schemes you already know

El-Gamal $ct_1 = (g^{r_1}, g^{r_2 s} m)$ and $ct_2 = (g^{r_2}, g^{r_2 s} m)$ with g^s the public key.
We can verify that it is multiplicatively homomorphic:
$(g^{r_1} g^{r_2}, g^{r_1 s} m_1 g^{r_2 s} m_2) = (g^{r_1 + r_2}, g^{(r_1+r_2)s} m_1 m_2)$ could be a valid cypher-text for $m_1 m_2$.

Dual Regev is additively homomorphic:
for $pk = A$, $\text{Enc}(pk, m_0) = As_0 + e_0 + (0, \lfloor q/2 \rfloor m_0)^T$ with $s_0 \longleftarrow^{\$} \mathbb{Z}_q^n$
for $pk = A$, $\text{Enc}(pk, m_1) = As_1 + e_1 + (0, \lfloor q/2 \rfloor m_1)^T$ with $s_1 \longleftarrow^{\$} \mathbb{Z}_q^n$
and $\text{Enc}(pk, m_0 + m_1) = A(s_0 + s_1) + (e_0 + e_1) + (0, \lfloor q/2 \rfloor (m_0 + m_1))^T$. A problem remains: the noise has increased but must be kept lesser than $q/4$.

2 FHE

2.1 Insecure FHE

An insecure (and non trivial) example:

- $pk = P \in \mathbb{Z}_q^{m \times n}$ such that $\text{leftKer}(P) \neq 0$
- $sk = s$ such that $s^T P = 0^n$
- $\text{Enc}(pk, m) = PR + mI_n$ where $R \longleftarrow^{\$} \mathbb{Z}_q^{n \times n}$
- $\text{Dec}(sk, ct) = sk \cdot ct = s^T(PR + mI_n) = ms^T$

This is fully homomorphic:

- Addition: $PR_1 + m_1 I_n + PR_2 + m_2 I_n = P(R_1 + R_2) + (m_1 + m_2) I_n$
- Multiplication: $(PR_1 + m_1 I_n)(PR_2 + m_2 I_n) = P(R_1 PR_2 + m_2 R_1 + m_1 R_2) + m_1 m_2 I_n$. Multiplying this by s^T on the left gives 0 for the P term and $s^T m_1 m_2$.

Still, any s in LeftKer(P) allows us to decrypt, and such s is easy to find.

2.2 LWE based FHE

LWE or "hard to find in almost the kernel"

Reminder: LWE is the problem consisting in finding s where only $(A, s^T A + e^T)$ is given.

Consider matrix $A = \begin{pmatrix} A' \\ s'^T A' + e^T \end{pmatrix}$. It is hard to find $s = \begin{pmatrix} s' \\ -1 \end{pmatrix}$ such that $s^T A = e^T$.

Thus, we use the following idea: replace in the previous insecure scheme P by A and s by $\begin{pmatrix} s' \\ -1 \end{pmatrix}$.

- pk=$\begin{pmatrix} A \\ s^T A + e^T \end{pmatrix}$ with $A \xleftarrow{\$} \mathbb{Z}_q^{n-1 \times n}$ and $s \xleftarrow{\$} \mathbb{Z}^{n-1}$
- sk=$\begin{pmatrix} s \\ -1 \end{pmatrix}$ such that $sk^T pk = -e^T$
- $\text{Enc}(pk, m) = AR + mI_n$ with $R \xleftarrow{\$} \{0,1\}^{n \times n}$
- $\text{Dec}(sk, ct) = sk^T \cdot ct = s^T \cdot (AR + mI_n) = -e^T R + ms^T$ with $-e^T R$ small since e and R are small.

Let us verify that it is very close to a FHE:

Addition $s^T(Enc_1 + Enc_2) = -e^T(R_1 + R_2) + s^T(m_1 + m_2)$ valid as $R_1 + R_2$ is still small.

Multiplication

$$\begin{aligned} ct_1 \times ct_2 &= (AR_1 + m_1 I_n)(AR_2 + m_2 I_n) \\ &= AR_1 AR_2 + m_2 AR_1 + m_1 AR_2 + m_1 m_2 I_n \\ s^T \times ct_1 \times ct_2 &= -e^T R_1 AR_2 - e^T R_2 m_1 - e^T R_1 m_2 + m_1 m_2 s^T \end{aligned}$$

Here we have a problem: the term $-e^T R_1 AR_2$ is not small since A is not small. The A blocks us when we want to decrypt.

There is a solution: the gadget matrix.

2.3 The Gadget matrix

We define the gadget matrix

$$G = \begin{pmatrix} 1 & 2 & 4 & \dots & 2^{\lfloor \log q \rfloor} & 0 & \dots & & & & & & & \\ 0 & 0 & 0 & \dots & & 1 & 2 & \dots & 2^{\lfloor \log q \rfloor} & 0 & \dots & & & \\ \vdots & & & & \vdots & & & & & \vdots & & & & \vdots \\ 0 & 0 & 0 & \dots & \dots & 0 & 0 & \dots & 0 & 0 & \dots & 1 & 2 & \dots & 2^{\lfloor \log q \rfloor} \end{pmatrix}$$

in $\mathbb{Z}_q^{n \times n\lfloor \log q \rfloor}$ (n lines, $n\lfloor \log q \rfloor$ columns).

For every $v \in \mathbb{Z}_q^n$, there is a unique $x \in \{0,1\}^{n\lfloor \log q \rfloor}$ such that $Gx = v$.

In particular, we can define G^{-1} the inverse function of G. But be careful: this function is not a matrix as it is not linear. It is just the binary decomposition of numbers in $\mathbb{Z}_q$.

Property. We have:

- $G^{-1} \cdot G = Id$
- G^{-1} maps into $\{0,1\}^{n\lfloor \log q \rfloor}$ which contains "small elements".

Thus, we transform the previous LWE based scheme to use G.

- pk=$\begin{pmatrix} A \\ s^T A + e^T \end{pmatrix}$ with $A \xleftarrow{\$} \mathbb{Z}_q^{n-1 \times n}$ and $s \xleftarrow{\$} \mathbb{Z}^{n-1}$
- sk=$\begin{pmatrix} s \\ -1 \end{pmatrix}$ such that $sk^T pk = -e^T$
- $\text{Enc}(pk, m) = AR + mG$ with $R \xleftarrow{\$} \{0,1\}^{n \times n}$
- $\text{Dec}(sk, ct) = sk^T \cdot ct = s^T \cdot (AR + mG) = -e^T R + ms^T G$ with $-e^T R$ small since e and R are small.

Once again, we must verify that it is indeed a FHE:

Addition $s^T(Enc_1 + Enc_2) = -e^T(R_1 + R_2) + s^T(m_1 + m_2)G$ valid as $R_1 + R_2$ is still small.

Multiplication Note that since the outputs of Enc are in $\mathbb{Z}_q^{n \times n\lfloor \log q \rfloor}$. Thus, multiplying the two outputs does not make sense. Thus, we find an operation that amounts to multiplying the messages: applying G^{-1} to ct_2. Then, the dimensions match.

$$\begin{aligned} ct_1 \times G^{-1} ct_2 &= (AR_1 + m_1 I_n) G^{-1} (AR_2 + m_2 I_n) \\ &= AR_1 G^{-1}(AR_2 + m_2 G) + m_1 G G^{-1}(AR_2 + m_2 G) \\ &= A(R_1 G^{-1}(AR_2 + m_2 G) + m_1 R_2) + m_1 m_2 G \\ s^T \times ct_1 \times G^{-1} ct_2 &= s^T A(...) + m_1 m_2 s^T G \qquad = -e^T(...) m_1 m_2 s^T G \end{aligned}$$

The term that multiplies $-e^T$ is small due to the application of G^{-1} to $(AR_2 + m_2 G)$. Thus, this is a valid encoding.

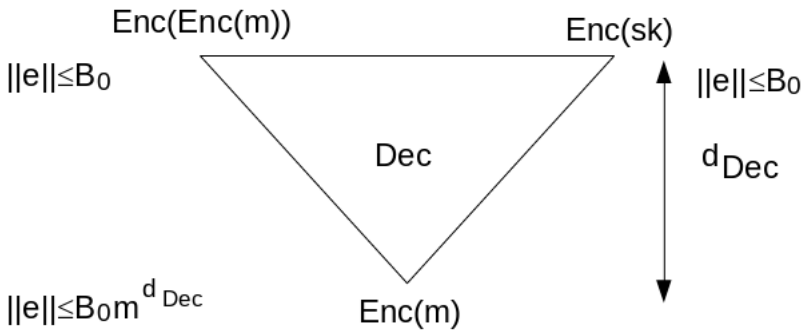

Figure 3: Bootstrapping

2.4 Noise Growth

Noise get doubled when adding, and is multiplied by approximately $n\lfloor \log q \rfloor$ when multiplying. Denote $m = n\lfloor \log q \rfloor$ and consider a circuit $\mathcal{C}$ of depth d, with gates denoting addition or multiplication. The homomorphic evaluation of $\mathcal{C}$ leads to an error bounded by $B_0 m^d$ with B_0 being the initial bound on the noise.

Conclusion so far: Choosing parameters appropriately can allow us to do any computation of given bounded depth. This is called "leveled fully homomorphic encryption" or "somewhat fully homomorphic encryption". Any depth can be achieved, but the parameters must be chosen accordingly.

3 Bootstrapping, or how to reduce the noise [Gentry 2008]

A trivial solution is to decode to reduce or remove the noise during the computation. But we do not have access to sk during the computation, and communication with a party that has sk is not allowed.

Idea: Reveal an encryption of sk. We can now run Dec homomorphically on any ciphertext.

Start with Enc(sk) and Enc(Enc(m)) for some ciphertext ct. We end up with Enc(m).

We process as in Figure 3.

If we can do one more operation, i.e. Decrypt with noise $||e|| \leq B_0 m^{d_{Dec}+1}$, we have a FHE.

Formally, we do not have security based on LWE, since some information about sk might have been revealed. We speak of security of FHE under *circular LWE*.