Encrypted Computation from Lattice

Rick

2021 年 3 月 13 日

1 Introduction

Hoeteck Wee's Chinese name is 黄和德, at least according to his WeChat account.

In this two-part talk, he first generalized the third generation of FHE into a rather "universal" lemma, then in the following part he showed how to tweak different parameters and variable meanings to get various advanced cryptographic objects. Examples include the following appliations.

表 1: Application of Today's Main Lemma

Applications	A_f	$H_{f,x}$	Reference
ABE	KeyGen	Decryption	[BGGHNSVV]
Fully Homomorphic Signature	Verify	Homomorphic Sign	[GVW]
Constrained PRFs	Normal Eval	Constrained Eval	[BV]

2 Generalized FHE

In this section Hoeteck proposed a generalized "formula" from fully homomorphic encryption.

2.1 Privacy in Big Data

Traditionally encryption protects privacy, but utility is lost meanwhile. Our goal here is to do secure computation – reconcile between privacy and utility.

Hoeteck proposed three notions of encrypted computation, and there is one unifying equation underlying these three notations.

2.2 Fully Homomorphic Encryption

It suffices to get SKE version of FHE, since we can include encryption of all necessary information to homomorphically evaluate the encryption circuit in the public key to mimic PKE. This will simply notation a bit.

Security sematic security.

Functionality $\mathsf{Enc}(\mathsf{sk},x) \to \mathsf{Enc}(\mathsf{sk},f(x))$

One of the cryptographic breakthrough, as hoeteck commented, is FHE for circuits from LWE.

First construct a "insecure" FHE in the sense that it is rather easy to make it secure. And of course we are familiar with this it is unnacessary to include it into our note.

Use approximate eigenvectors in the sense that:

$$\underbrace{t}_{\text{secret key}} \cdot \underbrace{A_i}_{\text{ct.}} \approx x_i \cdot t$$

From LWE assumption, A_i is pseudorandom and hides message. But multiplication fails here since small error times random matrix over $\mathbb{Z}_q^{n\times n}$ is big.

If we can reduce the norm of next matrix, then homomorphic multiplication is viable. So the requirement now is:

- matrix A hides x
- matrix A is small

Now we move into details a bit:

$$t = \begin{bmatrix} s & -1 \end{bmatrix}$$

where s is the LWE secret.

Now we have

$$\begin{bmatrix} s & -1 \end{bmatrix} \cdot \begin{bmatrix} B \\ sB+e \end{bmatrix} \approx 0$$

Where approx. is up to LWE noise. (B-bounded maybe a good idea)

Now we let

$$\underbrace{\begin{bmatrix} s & -1 \end{bmatrix}}_{\text{secret key}} \cdot \underbrace{\begin{bmatrix} B \\ sB+e \end{bmatrix} + x_i \cdot I}_{\text{ct.}} \approx x_i \cdot t$$

This satisfies sematices security, but the problem is big ciphertext. So we need to make it small. And now introducing the famous technique called "flattening" that Dalao's ususally practice.

We call this transform G^{-1}. A matrix $M \in \mathbb{Z}_q^{n\times n}$ transforms into $G^{-1}(M) \in \mathbb{Z}_2^{n\log q\times n}$. This is done by first picking the LSB matrix, then the next bit, and so on, till the MSB. In this way, not only the matrix is binary, but it left-multiplies the **gadget matrix** will get the original matrix.

$$G \in \mathbb{Z}_2^{n \times n \log q} = \begin{bmatrix} I & 2I & \dots & q/2I \end{bmatrix}$$

$$G \cdot G^{-1}(M) = M$$

Now we use this technique to make A small.

Now we use

$$\underbrace{\begin{bmatrix} s & -1 \end{bmatrix}}_{t} \cdot G \cdot \underbrace{G^{-1}\left(\begin{bmatrix} B \\ sB+e \end{bmatrix} + x_i \cdot I\right)}_{\text{ct.}} \approx x_i \cdot t$$

But still not good enough since the original approximate eigenvector equation is lost: one G on the LHS, and no G on the RHS. We can fix this problem however, by widening ciphertext. In particular:

$$\underbrace{\begin{bmatrix} s & -1 \end{bmatrix} \cdot G}_{t} \cdot \underbrace{G^{-1}\left(\begin{bmatrix} B \\ sB+e \end{bmatrix} + x_i \cdot G\right)}_{\text{ct.}} \approx x_i \cdot t$$

And that is just fine, since by LWE assumption, we just increase the LWE samples from n to $n \log q$, and it does not affect security at all. Now the two requirements are met simutaneously.

"But this is basically the entire construction that underlies one of the cryptographic breakthoughs in our generation" – Hoeteck Wee.

2.3 Noise Control

Two ideas in the literature in controlling small noise.

2.3.1 First Idea

- $G^{-1}(M_1) \cdot G^{-1}(M_2)$ means noise eventually grows from small to small$^{\deg(f)}$. Too bad.
- Instead, We do $G^{-1}(M_1 \cdot G^{-1}(M_2))$ which generates the same result by left multiplying G. Eventually this would imply noise growth from small to small$^{\log \deg f}$. But he did not explain how to do it. Indeed, he admitted that he suddenly begin "cheating". By ensuring slow growth, one can also rely on stronger noise, which means stronger LWE assumption.

2.3.2 Second Idea

In the circuit model, noise follows the pattern:

$$\text{small}_{\text{out}} = n \cdot \text{small}_{\text{prev}}$$
$$\text{small}_{\text{output}} = n^{\text{depth}} \cdot \text{small}_{\text{input}}$$

But in branching program, since every multiplication has at least one operand from the input, noise will only grow additionally. Here we have

$$\text{small}_{\text{out}} = \text{small}_{\text{prev}} + n \cdot \text{small}_{\text{input}}$$
$$\text{small}_{\text{output}} = n \cdot \text{length} \cdot \text{small}_{\text{input}}$$

And so we have the following comparison. Circuit of depth $O(\log n)$ is a subset of branching program of length poly. But our analysis suggests that circuit has $n^{\log n}$ noise blow-up, while branching program has poly noise blow-up. And so by using BP for log-depth circuits, we can use larger noise, and hence polynomial (rather than sub-exponential) hardness assumption in FHE and other circumstances ([BV14,AP14,GVW13]).

2.4 Generalizing Eigenvector Lemma

So far we focused on the property that matrices that share the same eigenvector will have "homomorphism" so that we can perform computation on eigenvalues with matrices alone. Hoeteck now generalizes it to any vectors whether they share eigenvectors or not.

2.4.1 Proposition (Lemma II)

$\forall A_i$, $\forall x$, $\forall f$, $\exists H_{f,x}$

$$\begin{bmatrix} A_1 - x_1 I & \dots & A_n - x_n I \end{bmatrix} \cdot H_{f,x} = A_f - f(x) I$$

First observe that Lemma 2 (generalized version) implies Lemma 1. By left multiplying t we have the requied form. (but Lemma 1 is so trivial you do have to prove it this way).

2.4.2 Proving Lemma 2:

First consider addition. Notice how we only have to come up with matrix H. In this case $H = \begin{bmatrix} I \\ I \end{bmatrix}$, easy.

Then consider multipliction. Notice $H_{f,x}$ is the last quantified variable, which means it can depends on x and f. We can then design H to be

$$H = \begin{bmatrix} A_2 \\ x_1 I \end{bmatrix}.$$

By induction, we can conclude the proof.

But notice how this is insecure, since no noise is introduce. We need to add noise. Introducing asteristics – eigenvectors, revisited*, lemma 2*, etc.

2.4.3 Proposition (Lemma II*):

$\forall A_i$, $\forall x$, $\forall f$, $\exists$**small**$H_{f,x}$

$$\begin{bmatrix} A_1 - x_1 G & \dots & A_n - x_n G \end{bmatrix} \cdot H_{f,x} = A_f - f(x)G$$

Notice how a wider A_i will fail the original way of generating A_f since it is impossible to multiplying two rectangular matrices.

Also small H will ensure cascade operations. This allows more sophiscated constructions in cryptography.

2.4.4 Corollary 1.

Note how this would imply the claim that

$$t \cdot A_i \approx x_i t G$$

implies

$$t \cdot A_f \approx f(x) t G.$$

Since small noise times small matrix H is still small, we get effect as good as zero.

2.4.5 Corollary 2.

$$(s \begin{bmatrix} A_1 - x_1 G & \dots & A_n - x_n G \end{bmatrix} + e) \cdot H_{f,x} \approx s \cdot (A_f - f(x)G)$$

2.4.6 Proving Lemma 2*.

Once again, addition is easy to handle, H is two identity matrices stacked together.

The original way of handling multiplication would fail since not only entries are big, we cannot simply multiply two rectangular matrices.

We solve this by

$$\begin{bmatrix} A_1 - x_1 G & A_2 - x_2 G \end{bmatrix} \cdot \begin{bmatrix} G^{-1}(A_2) \\ x_1 I \end{bmatrix} = A_1 G^{-1}(A_2) - x_1 x_2 G$$

Notice how $A_f = A_1 G^{-1}(A_2)$.

2.4.7 Improvement?

This is more of a conceptual improvement, since in some sense allows more constructions, and it considers branching program model which has better noise management than circuit model.

3 Application

In this section Hoeteck introduced ABE and Laconic Function Evaluation as two examples of the generalized lemma in the last section. Here ABE has two technical routes, where the second one uses the aforementioned lemma.

3.1 Attribute-Based Encryption

ABE has the setting that sender want to encrypt message M with input f. Each receiver has property x's and they cannot decrypt unless $f(x) = 0$. Here f seems to be refered to as "access policy", seems legit.

So it should be clear that complexity of f should be the key measurement in effectiveness of ABE. Before 2013, the literature can only achieve shallow circuits, but after 2013, we have ABE for all circuits from LWE.

Hoeteck gives in my opinion, a most fitting example for ABE – dating website. Suppose users are registered on a dating website and they want to share profiles. But only those satisfying their criteria will be able to view their personal information. In this example, key distribution as well as key management are dealt by a central website. These simplifications allows us to focus on how to implement such a access policy.

3.1.1 First Method

This is a line of work starting from [GVW13], improved by [BGGH,NSVV14] later on.

The basic idea is simple: assign random key to each attribute and use them as OTP. One obvious drawback is mix-and-match attack (not collude resilient).

This can be improved however, by replacing random string by random function. This is the key idea in [GVW13]. By using random function, we can use it multiple times by using different indices.

So now we have random funcitons e.g. ϕ_{cs}, ϕ_{phd}, etc.

Creating key by picking random input, and then evaluate the function on the input corresponding to the user's attribute. e.g. $\phi_{cs}(s)$, $\phi_{phd}(s)$ with random s for a cs phd.

Hoeteck did not mention much detail about encryption. This process is a bit tricky since the encryption process should not rely on randomness used in key generation, and thus it any evaluation on satisfying attribute functions should be able to decrypt.

Now naive mix and match should fail since they are from different users, and therefore unlikely to be on the same point of evalutaion.

More generally this idea is sufficient for all collusions. It also works for general circuits.

3.1.2 Second Method

This method uses previous Lemma. In this part of the presentation, Hoeteck uses a "dual" version of ABE functionality. In particular, sender now holds a string x while receiver holds function f. Receiver can decrypt $[M]$ iff. $f(x) = 0$. For now f can be considered as a polynomial. In general, he commented, this does not affect the resulting scheme.

Public Key is A_1, \$..., A_n, P uniformly random.

Encryption:

$s \leftarrow \$ \mathbb{Z}_q^n$, output $s\begin{bmatrix} A_1 - x_1 G & \dots & A_n - x_n G \end{bmatrix} + e$, $sP + M$

Secret key is a short vector such that $A_f \cdot \mathsf{sk}_f = P$

Correctness:

If $f(x) = 0$, receiver learns M.

Multiply both side by H, we have

$$(s\begin{bmatrix} A_1 - x_1 G & \dots & A_n - x_n G \end{bmatrix} + e) \cdot H_{f,x} \approx s(A_f - f(x)G)$$

Since $f(x) = 0$ we can proceed by right multiplying sk_f to get sP.

Sadly, Hoeteck states that this scheme is yet provably secure. We need to tweak it a bit further. This is done by adding an additional vector in ciphertext. Now we have:

- Public Key: $A, A_1, \ldots, A_n, P$
- Encryption: $s\begin{bmatrix}A_1 - x_1 G & \ldots & A_n - x_n G\end{bmatrix} + e,\ sP + M,\ sA + e'$
- Secret Key: $\begin{bmatrix}A & A_f\end{bmatrix} \cdot \mathsf{sk}_f = P$

Proof of security: M is hidden given ciphertext, sk_f, f s.t. $f(x) \neq 0$.

My understanding is that A_f ensures utility while A ensures security. If we decompose sP by

$$\underbrace{s}_{\text{hidden}} \cdot \underbrace{\begin{bmatrix}A & A_f\end{bmatrix} \cdot \mathsf{sk}_f}_{\text{no entropy}}$$

Since A_f is correlated with previous randomness, A is fresh, we can use A as public randomness in GL-theorem to extract pseudorandom bits. Hoeteck explained this in more "proof-oriented" manner.

First observe by LWE assumption, if only A, P are public, $sP + M$ and $sA + e'$ are pseudorandom. So now we need to simulate other information $\bar{A} = \begin{bmatrix}A_1 - x_1 G & \ldots & A_n - x_n G\end{bmatrix}$ and $sk_f : \begin{bmatrix}A & A_f\end{bmatrix} \cdot \mathsf{sk}_f = P$. He just briefly mentioned we can do this by setting $\bar{A} = AR$ where R is a small matrix. In this way, we can use $(sA + e') \cdot R$ to simulate $s\bar{A} + e$. Since R and e are both small, noise will not blow up too much.

Moreover, sk_f can be simulated as

$$\mathsf{sk}_f = \begin{bmatrix}-RH_{f,x} \\ I\end{bmatrix} \cdot G^{-1}(1/f(x)P)$$

Basically that is how its proved.

3.2 Laconic Function Evaluation

References: [QWW18,CDGGMP17]

Settings:

Alice the user has x while Bob the big company has function f (potentially very complicated). Bob sends in the first round a digest to Alice, then in the second round, Alice send ciphertext back to zBob, allowing Bob to learn $f(x)$.

Requirements:

Security hides x, a semi-honest Bob learns only $f(x)$ and nothing else.

Efficiency cost is independent of f, computation and communication

Naive FHE will require three rounds, rather than two, to finish this task.
Solution using Lemma 2:

- digest $= A_1, \ldots, A_n, A_f$ (notice that f is not protected)
- ciphertext $\approx s\left[A_1 - x_1 G \quad \ldots \quad A_n - x_n G\right], sA_f$

Notice how this will not work since $H_{f,x}$ depends on x so we will circumvent this problem by setting the function $\hat{f}$ to be homomorphic evaluation of f. In this way, input to $\hat{f}$ are encrypted x_i which can be disclosed, so now H can be computed. Another point I think is that s should also be the FHE secret key, since what finally gets is $s\hat{f}(x)G$, which is in the form of secret key dot product with ciphertext, exactly the form of decryption circuit in most FHE schemes.

4 Conclusion

Lattice based encrypted computation has two paradigms. The first way is the lemma introduced today. The second way is in [GGH15,CC17,GKW17,WZ17,CVW18].