The CKKS Cryptosystem
Part 3: Final Words

Introduction to Homomorphic Cryptosystems - Lecture 5

HOW TO WORK WITH THE ERROR

What about the error?

- CKKS security is based on adding artificial errors ((R)LWE)
- Executing operations on the ciphertext increases the error
- This makes CKKS a levelled encryption schema
  - Every ciphertext has a certain level and doing operations on the ciphertext lowers the level
  - When we reach the lowest level, no more operations are possible (otherwise we would not be able to recover the results)
- **But:** To meet the requirements of a FHE schema, we need to be able to do unlimited number of operations
- How do we do this in CKKS?
  - (Rescaling)
  - Bootstrapping

Rescaling

Note
We omitted the scaling step during encoding and decoding, but you should know, that this step exists and that it forces us to have a rescaling operation.

Scaling
To minimize the error introduced in the rounding step of the encoding, a scaling Δ is applied:

$$x = \mathrm{Enc}(\mathrm{Ecd}(v \odot \Delta))$$

This means, that when multiplying two scaled numbers, we also multiply the scale:

$$\mathrm{Dcd}(\mathrm{Dec}(\mathrm{Mul}(x, y))) = \mathrm{Dcd}\left(\mathrm{Dec}\left(\mathrm{Mul}\left(\mathrm{Enc}\left(\mathrm{Ecd}(v \odot \Delta)\right), \mathrm{Enc}\left(\mathrm{Ecd}(v' \odot \Delta)\right)\right)\right)\right) \approx v \odot v' \odot \Delta^2$$

Rescaling
The rescaling operation can be applied to the ciphertext after the multiplication to keep the scale constant.
As it's a division by a certain value, it also helps to reduce the noise, but at the same time decreases the modulus.

Addition?
Adding two scaled values does not change the scaling factor

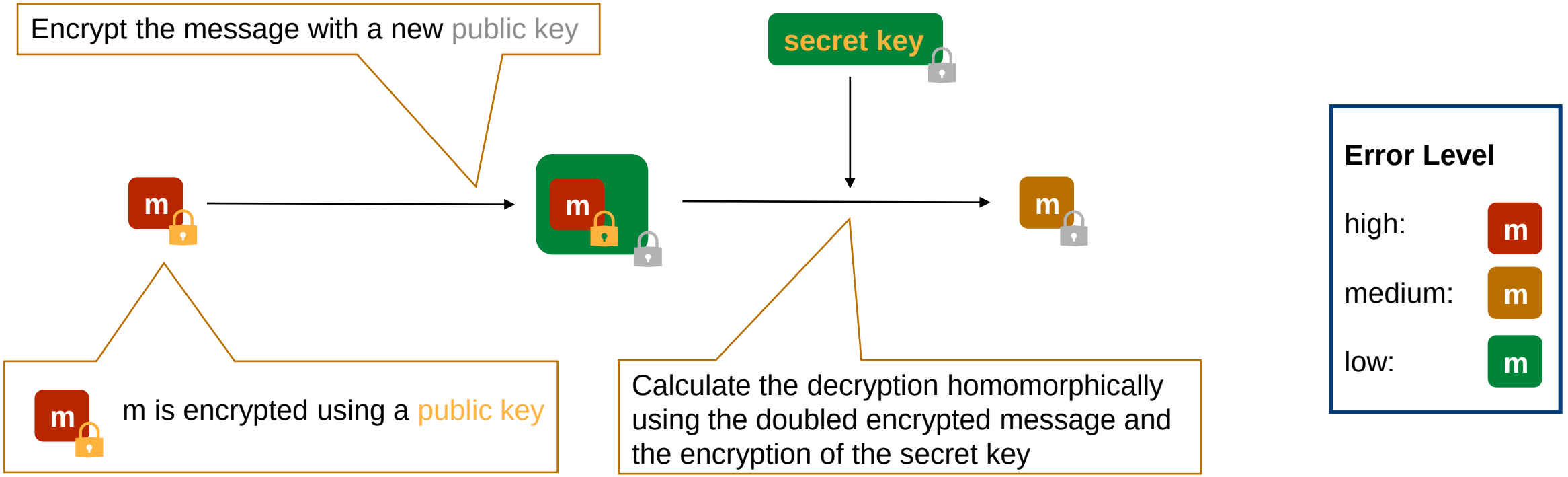
Bootstrapping

Goal

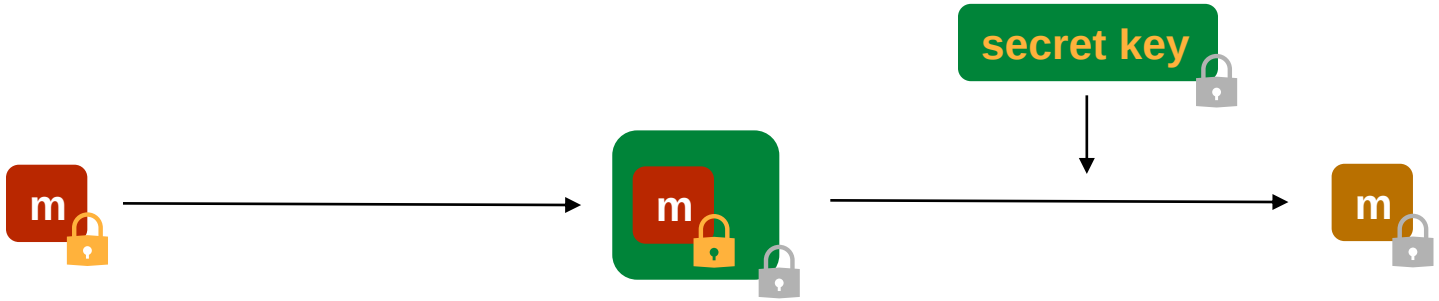
Reduce the error in the ciphertext, thus allowing an infinite number of operations.

Idea

We can evaluate the Dec Operation homomorphically



Bootstrapping



Error Level

high: m

medium: m

low: m

Notes

- The owner of the data must provide a bootstrapping key pair for every bootstrap operation and generate them beforehand
- It is usually unclear how many bootstrap operations are necessary
- The bootstrap operation itself also increases the error
- Calculating the decryption homomorphically is not straightforward (especially in CKKS)
- Bootstrapping introduces a huge performance overhead

HOW TO SET SECURITY PARAMETERS

Security Parameters

These are the same q_L and n as from the last lectures!

- Before doing any calculations, we set certain security parameters
- Most importantly we have to choose a modulus $q_L \in \mathbb{N}$ and a ring dimension $n \in \{2^k \mid k \in \mathbb{N}\}$
- As the notation suggests the modulus depends on the multiplicative depth L (and the scaling factor)
  - If we want to do multiple multiplications in a row we need a bigger modulus
  - **But:** A bigger modulus results in increased computation time
- Based on the modulus and the desired level of security we can choose the ring dimension

Security Parameters

- Libraries like OpenFHE choose the parameters based on the Homomorphic Encryption Standard (https://homomorphicencryption.org/standard/)
- They provide combinations of n and q which achieve different levels of security

A security level is typically expressed in “bits of security”. n-bit security means, that the attacker would have to perform 2^n operations to break it.

n	security level	logq	uSVP	dec	dual
1024	128	27	131.6	160.2	138.7
	192	19	193.0	259.5	207.7
	256	14	265.6	406.4	293.8
2048	128	54	129.7	144.4	134.2
	192	37	197.5	233.0	207.8
	256	29	259.1	321.7	273.5
4096	128	109	128.1	134.9	129.9
	192	75	194.7	212.2	198.5
	256	58	260.4	292.6	270.1
8192	128	218	128.5	131.5	129.2
	192	152	[illegible]		
	256	118	256.7	273.0	[illegible]
16384	128	438	128.1	129.9	129.0
	192	305	192.1	196.2	193.2
	256	237	256.9	264.2	259.8
32768	128	881	128.5	129.1	128.5
	192	611	192.7	194.2	193.7
	256	476	256.4	260.2	258.2

These are currently known attacks on the (R)LWE with their according security level

After choosing a multiplicative depth L, calculating our modulus q and choosing a security level we can use this table to get the ring dimension.

Taken from
http://homomorphicencryption.org/wp-content/uploads/2018/11/HomomorphicEncryptionStandardv1.1.pdf

LIMITATIONS OF CKKS

Addition and Multiplication are not enough!

- CKKS (and other FHE encryption schemes) only allow addition and multiplication on the ciphertexts
- In the real world we need more then that!

Example: Division

Exercise:
How can we implement the division with addition and multiplication?

Addition and Multiplication are not enough!

- CKKS (and other FHE encryption schemes) only allow addition and multiplication on the ciphertexts
- In the real world we need more then that!

Example: Division

What is the problem here?

```
#integer division a/b
def div(a,b):
  result = 0
  while(a > 0):
    a = a - b
    result++
  return result

div(15,5) -> 3
```

Answer:
We cannot calculate this homomorphically!
How can we know, that a number is bigger than another number?

We need other methods to implement more advanced functions like division, square root etc.

Summary – What did we learn today?

Error Management

To be a FHE scheme CKKS (and others) implement functions that preserve the level (noise) of the ciphertext.

Bootstrapping is a method to reduce the noise, and it works by applying the decryption homomorphically.

Security Parameters

The security parameters are chosen according to the desired level of security

CKKS Limitations

Implementing advanced functions is not straightforward