Cloud Security: Part 4

Gaurav S. Kasbekar
Dept. of Electrical Engineering
IIT Bombay

NPTEL

References

- W. Stallings, "*Cryptography and Network Security*", 8th edition, Pearson Education, 2023
- C. Kaufman, R. Perlman, M. Speciner, R. Perlner, "*Network Security: Private Communication in a Public World*", Pearson Education, 3rd edition, 2023

NPTEL

Fully Homomorphic Encryption (FHE)

Fully Homomorphic Encryption (FHE)

- With homomorphic encryption, the data can be encrypted (in a special way), computation can be done on the encrypted data, and the result is the encrypted answer
- So, computing on the plaintext data will yield the same answer as computing on the encrypted data and decrypting the result
- An application of homomorphic encryption is to store one's encrypted data in a public cloud and do computations on the encrypted data, without needing to trust the cloud not to leak one's data
- Note that any computation can be done with a circuit consisting of $\wedge$ (AND) and $\oplus$ (XOR)
- Hence, to achieve a homomorphic encryption scheme, we need to use some mathematics that allows both the above operations on encrypted data
- Next, we discuss a simple homomorphic encryption scheme

Simple FHE Scheme

- Most FHE schemes involve complicated mathematics
- To gain some intuition, we discuss one scheme that is easy to understand
- It is not practical (e.g., the encrypted data would expand by a factor of about a billion)
- The scheme requires encryption to be done on each bit
- So it is necessary to know how to encrypt 0 and how to encrypt 1
- The private key is a large odd integer, say n
- The scheme performs ordinary integer arithmetic, and not modular arithmetic

Simple FHE Scheme (contd.)

- To encrypt a bit if you know n:
  - Choose some very large multiple of n
  - Add or subtract a relatively small even number (which we will call “noise”)
    - We call the value at this point a noisy multiple of n
    - If you are encrypting a 0, you are now done
    - Encrypted bit of form: $c = kn \pm e$
  - If the bit to be encrypted is a 1, add or subtract 1
    - Encrypted bit of form: $c = kn \pm e \pm 1$
- In order to allow someone who does not know n to encrypt:
  - We create a public key that is a list of encryptions of 0
  - To encrypt a 0, one would add together a randomly chosen subset of the encryptions of 0
    - E.g., if $c_1 = k_1 n + e_1$, $c_2 = k_2 n + e_2$, and $c_3 = k_3 n + e_3$, then $c_1 + c_2 + c_3 = (k_1 + k_2 + k_3)n + (e_1 + e_2 + e_3)$
  - To encrypt a 1, they would do the same thing, but add or subtract 1 at the end
- Only someone who knows n will be able to decrypt:
  - To decrypt a value x, find the nearest multiple of n
  - If the difference between x and the multiple of n is even, x decrypts to a 0
  - If the difference is odd, x decrypts to a 1

Simple FHE Scheme (contd.)

- Adding two encrypted bits results in the ⊕ (XOR) of the two plaintext bits
  - ❑ If $c_1 = k_1 n + e_1$ and $c_2 = k_2 n + e_2$, then $c_1 + c_2 = (k_1 + k_2)n + (e_1 + e_2)$
  - ❑ If $c_1 = k_1 n + e_1$ and $c_2 = k_2 n + e_2 + 1$, then $c_1 + c_2 = (k_1 + k_2)n + (e_1 + e_2) + 1$
  - ❑ If $c_1 = k_1 n + e_1 + 1$ and $c_2 = k_2 n + e_2 + 1$, then $c_1 + c_2 = (k_1 + k_2)n + (e_1 + e_2 + 2)$
- Multiplying two encrypted bits results in the ∧ (AND) of the two bits
  - ❑ If $c_1 = k_1 n + e_1$ and $c_2 = k_2 n + e_2$, then $c_1 \times c_2 = (k_1 k_2)n^2 + (k_1 e_2 + k_2 e_1)n + (e_1 e_2)$
  - ❑ If $c_1 = k_1 n + e_1$ and $c_2 = k_2 n + e_2 + 1$, then $c_1 \times c_2 = (k_1 k_2)n^2 + (k_1 e_2 + k_2 e_1 + k_1)n + (e_1 e_2 + e_1)$
  - ❑ If $c_1 = k_1 n + e_1 + 1$ and $c_2 = k_2 n + e_2 + 1$, then $c_1 \times c_2 = (k_1 k_2)n^2 + (k_1 e_2 + k_2 e_1 + k_1 + k_2)n + (e_1 e_2 + e_1 + e_2) + 1$
- Adding or subtracting 1 to an encrypted bit computes the NOT of the encrypted bit

Simple FHE Scheme (contd.)

- Note that the noise increases each time two encrypted bits are added or multiplied
- If the noise ever gets bigger than $n/2$, decryption will no longer be guaranteed to produce the right answer
- So, we can do additions and multiplications, but we have to avoid letting the noise get bigger than $n/2$
- Another limitation of the scheme is that the size of the encrypted bit doubles each time a multiplication is performed
  - ❑ If you multiply two billion-bit numbers together, you get a two-billion bit number
  - ❑ After some multiplications, the numbers (which were large while starting), get extremely large
- For the above reasons, this scheme is not practical
- There are more practical homomorphic schemes proposed in the research literature, and this is an area of active research