

I2C homeworks

1.

An example of MPC usage is when different employees want to know their average wage without letting others know what is their real wage. So they can take their wage and split this data into shares (the number of shares has to be the same as the number of parties). So they have created n shares that summed up will give the real value but seen alone they are totally random numbers. These shares are then sent to the other parties (only one per party) and when they want to know the average wage they send all their shares to a MPC computer that sums up all the shares and divides the result by the number of parties.

A second example is used for data recovery on internet package. This uses the Reed Solomon technique explained more in detail on answer 6 and this lets the receiver to retrieve a message that has been split in n parts (due to internet frame limitation) even if $n - k$ parts are lost where k are the minimum number of parts that has to arrive to reconstruct the message.

2.

Homomorphic is a property for some encryption methods that guarantees to make computation on the cypher-text without before decipher it.

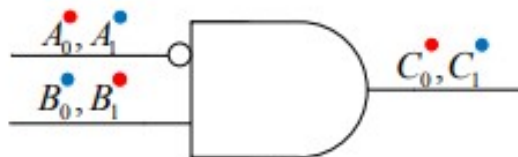
The RSA is homomorphic because $RSA(m) = m^e \bmod n$ so we can say for messages m_1, m_2 that

$$RSA(m_1) * RSA(m_2) = m_1^e * m_2^e \bmod n = (m_1 * m_2)^e \bmod n = RSA(m_1 * m_2)$$

Considering CAESAR cipher instead we can just simply consider that concatenating a new string ciphered with the same key won't change the decryption of the first message and will for sure do not change the decryption for the message attached. The change of length of the message does not consist in any problem because CAESAR cipher deciphers a message letter by letter without keeping track for before computations. So if I $ENC_3(A) \parallel ENC_3(B)$ I will get DE, the same as $ENC_3(A \parallel B)$.

3.

There is a way to reduce the usage of rows by using a method called "the garbled row reduction 3 ciphertexts" also known as GRR3. Assuming an and gate structured like the image.



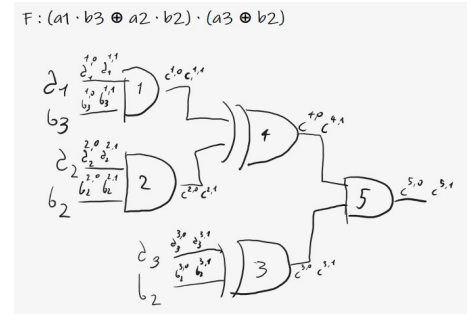
The only way the circuit returns a 1 will be with A_0, B_1 every other permutation would return C_0 . So as we have to choose a value for C_0 and C_1 we can save one row by not choosing those 2 numbers randomly but choosing C_1 as the value that encrypted with A_0, B_1 returns a string composed by all zeros. With this we don't have to send all the 4 rows cause the receiver can assume that if he receives 3 rows

he is able to retrieve the values for the cases $A_0 B_0$, $A_1 B_1$ and $A_1 B_0$ and can assume that in the case $A_0 B_1$ the C_1 is a string composed by zeros.

We can use the same approach for the OR gate just considering that the operation $A_0 \text{ OR } B_0$ is the only one giving C_0 , so we can choose a C_0 that is a string of zeros and send only 3 rows letting the receiver know that when he has A_0 and B_0 he has to assume the value is the string of zeros.

4.

For Yao having 5 gates and assuming we are have at most 20 rows to be sent from A to B, and the OT's used are 3. we will have a total of 20 values used as keys and 2 results for the circuit.



5.

Considering the logics to achieve a sum are an XOR for the sum and an AND for the carry we will for sure use as communication cost at maximum 8 rows (as for 2 logic ports) we can reduce it to 7 using the reduction on the AND gate as specified on answer 3. We can also consider to use as table

$$a_0 + b_0 = c_0$$

$$a_0 + b_1 = c_1$$

$$a_0 + b_2 = c_2$$

$$a_1 + b_0 = c_1$$

$$a_1 + b_1 = c_2$$

$$a_1 + b_2 = c_0$$

$$a_2 + b_0 = c_2$$

$$a_2 + b_1 = c_0$$

$$a_2 + b_2 = c_1$$

With this table we can easily garble the whole function ignoring the logical circuit behind.

6.

A secret share in a MPC consists in splitting a data through the parties letting them know only a piece of it. Then if anyone wants to know what is the complete data has to ask any other party to their piece. There pieces are called shares and any member of the party has only one. A famous technique is the Reed Solomon technique. This method uses an annotation (n, k) where n are the number of parties (so the number of times the data is split) and k is the number of parties required to collaborate to get the information. The algorithm uses a geometrical solution assigning the secret message s as a point $(0, s)$ and considering the k generates $k-1$ random points. With all those points I can compute a function that generates points and those (except for the $(0, s)$) are shared one per party. So only if more or equal than k parties collaborate the secret is readable.

Secret sharing is useful as a standalone method because it can be used when you want to have multiple permissions to do something, an example is (as in some American films) the launch of missiles handled using multiple keys of codes to authorize the launch. In an MPC the reason secret sharing is used is because we want to agree on a key without sharing personal data from any party.

8.

Algorithm [\[edit \]](#)

As Wikipedia says.

The scheme works as follows:

Key generation [\[edit \]](#)

1. Choose two large prime numbers p and q randomly and independently of each other such that $\gcd(pq, (p-1)(q-1)) = 1$. This property is assured if both primes are of equal length.^[1]
2. Compute $n = pq$ and $\lambda = \text{lcm}(p-1, q-1)$. lcm means Least Common Multiple.
3. Select random integer g where $g \in \mathbb{Z}_{n^2}^*$
4. Ensure n divides the order of g by checking the existence of the following modular multiplicative inverse:
$$\mu = (L(g^\lambda \bmod n^2))^{-1} \bmod n,$$

where function L is defined as $L(x) = \frac{x-1}{n}$.

Note that the notation $\frac{a}{b}$ does not denote the modular multiplication of a times the modular multiplicative inverse of b but rather the quotient of a divided by b , i.e., the largest integer value $v \geq 0$ to satisfy the relation $a \geq vb$.

- The public (encryption) key is (n, g) .
- The private (decryption) key is (λ, μ) .


If using p, q of equivalent length, a simpler variant of the above key generation steps would be to set $g = n + 1$, $\lambda = \varphi(n)$, and $\mu = \varphi(n)^{-1} \bmod n$, where $\varphi(n) = (p-1)(q-1)$.^[1]

Encryption [\[edit \]](#)

1. Let m be a message to be encrypted where $0 \leq m < n$
2. Select random r where $0 < r < n$ and $r \in \mathbb{Z}_n^*$ (i.e., ensure $\gcd(r, n) = 1$)
3. Compute ciphertext as: $c = g^m \cdot r^n \bmod n^2$

Decryption [\[edit \]](#)

1. Let c be the ciphertext to decrypt, where $c \in \mathbb{Z}_{n^2}^*$
2. Compute the plaintext message as: $m = L(c^\lambda \bmod n^2) \cdot \mu \bmod n$

As the original [paper](#)  points out, decryption is "essentially one exponentiation modulo n^2 ."

Homomorphic properties [\[edit \]](#)

A notable feature of the Paillier cryptosystem is its homomorphic properties along with its non-deterministic encryption (see Electronic voting in Applications for usage). As the encryption function is additively homomorphic, the following identities can be described:

• Homomorphic addition of plaintexts

The product of two ciphertexts will decrypt to the sum of their corresponding plaintexts,

$$D(E(m_1, r_1) \cdot E(m_2, r_2) \bmod n^2) = m_1 + m_2 \bmod n.$$

The product of a ciphertext with a plaintext raising g will decrypt to the sum of the corresponding plaintexts,

$$D(E(m_1, r_1) \cdot g^{m_2} \bmod n^2) = m_1 + m_2 \bmod n.$$

• Homomorphic multiplication of plaintexts

A ciphertext raised to the power of a plaintext will decrypt to the product of the two plaintexts,

$$\begin{aligned} D(E(m_1, r_1)^{m_2} \bmod n^2) &= m_1 m_2 \bmod n, \\ D(E(m_2, r_2)^{m_1} \bmod n^2) &= m_1 m_2 \bmod n. \end{aligned}$$

More generally, a ciphertext raised to a constant k will decrypt to the product of the plaintext and the constant,

$$D(E(m_1, r_1)^k \bmod n^2) = k m_1 \bmod n.$$

However, given the Paillier encryptions of two messages there is no known way to compute an encryption of the product of these messages without knowing the private key.