C/C++ implementation of a cryptographic protocol
Using Paillier’s Homomorphic Cryptographic Primitive

Objective
• Implement a protocol using a modern cryptographic primitive
• It is useful in electronic banking and voting
• C to implement the Paillier Cryptography
• C++ to implement the protocol using NSPR

The Paillier’s Cryptosystem
• Asymmetric cryptographic system created by Paillier in 1999
• Additive homomorphic property
  • Add two plaintexts with public key and their respective ciphertexts

The Protocol

Alice and Bob are both wealthy people and they want to ascertain who is wealthier without disclosing the amount of wealth they have. We are assuming both parties are semi-honest: they provide true information, but will not hesitate to snoop information that is not for them to know.

Alice

1. Creates public key (pub_key) and private key (prv_key) as well as calculating ciphertext for Alice’s wealth $E(-A)$

**ACTIONS :**
prv_key, pub_key = paillier_key_gen
$E(-A) = enc(pub_key, Alice\_wealth)$

2. Uses pub_key to calculate ciphertext for Bob’s wealth $E(A)$.

3. Uses the public key and private key previously calculated to get $X(A-B)$.

**ACTIONS :**
Use the sign of $X(A-B)$ to determine who is richer

**If positive, Alice is poorer.**
**If negative, Alice is richer.**
**If zero, they are equally wealthy**

Bob

2. Uses pub_key to calculate ciphertext for Bob’s wealth $E(B)$.

**ACTIONS :**
Uses pub_key to calculate the homomorphic multiplication of $E(-A)$ and $E(B)$, which is just $E(A - B)$.

Uses pub_key to multiply $E(A - B)$ by a random integer, $X$, to get $E(X(A-B))$

**ACTIONS :**
$E(B) = enc(pub_key, Bob\_wealth)$
$E(A-B) = mul(pub_key, E(B), E(-A))$
$E(X(A-B)) = pow(pub_key, E(A-B), X); where X is random integer.$

4. Receive the result and determine Bob’s relative wealth.

Alice and Bob now know who is wealthier, but can never determine the other’s wealth. Save for the case, where they have the same exact wealth.