Partially Homomorphic Cryptography

$$
\begin{tikzcd}
& m_1 \arrow[dl] \arrow[r, leftrightarrow] & E(m_1) \arrow[dr] & \\
m_1 \oplus m_2 & & & E(m_1) \otimes E(m_2) \equiv E(m_1 \oplus m_2) \\
& m_2 \arrow[ul] \arrow[r, leftrightarrow] & E(m_2) \arrow[ur] &
\end{tikzcd}
$$

Homomorphic Cryptography

- Homomorphic encryption is a form of encryption that allows computations to be carried out on ciphertexts
- The encrypted result which, when decrypted, matches the result of certain operations performed on the plaintext
- Fully homomorphic encryption algorithms allow several operations, such as addition, multiplication, and multiplication by a scalar
- It may also allow more complicated functions

Homomorphic Cryptography

- For example, given $c_1 = E(m_1)$ and $c_2 = E(m_2)$, and a scalar α, we may be able perform certain operations on ciphertexts c_1 and c_2, and obtain the ciphertexts that are encryptions of:

$$\begin{array}{rcccl} c_1 \oplus c_2 & = & E(m_1) \oplus E(m_2) & = & E(m_1 + m_2) \\ c_1 \odot c_2 & = & E(m_1) \odot E(m_2) & = & E(m_1 \cdot m_2) \\ \alpha \otimes c_1 & = & \alpha \otimes E(m_1) & = & E(\alpha \cdot m_1) \end{array}$$

- Other operations such as $m_1 < m_2$ are also very useful
- A fully homomorphic encryption function allows the computation of any function $g(\ )$ over the plaintext with the help of another function $f(\ )$ over the ciphertext

$$c = E(m) \quad \Rightarrow \quad f(c) = E(g(m))$$

Partially Homomorphic PKC

- Many public-key cryptographic functions are partially homomorphic, that is, they allow a subset of ciphertext operations
- For example, consider that $c = E(m)$ is the RSA function, i.e., $c = E(m) = m^e \pmod{n}$, where (e, n) are the RSA public (or private) exponent and the modulus
- Given $c_1 = E(m_1)$ and $c_2 = E(m_2)$, we can perform modular multiplications

$$
\begin{aligned}
c_1 \cdot c_2 &= E(m_1) \cdot E(m_2) \pmod{n} \\
&= m_1^e \cdot m_2^e \pmod{n} \\
&= (m_1 \cdot m_2)^e \pmod{n} \\
&= E(m_1 \cdot m_2)
\end{aligned}
$$

- Therefore, the RSA encryption is **multiplicatively** homomorphic

Additively Homomorphic RSA?

- Is the RSA encryption is **additively** homomorphic?
- Unfortunately, the encrypted text $E(m_1 + m_2)$ cannot be easily obtained using the ciphertexts $c_1 = E(m_1)$ and $c_2 = E(m_2)$

$$
\begin{aligned}
E(m_1 + m_2) &= (m_1 + m_2)^e \pmod{n} \\
&\stackrel{?}{=} m_1^e \oplus m_2^e \pmod{n}
\end{aligned}
$$

- There does not seem to exist a simple operation $\oplus$ so that when we apply $c_1 \oplus c_2$, we could obtain $E(m_1 + m_2)$
- The RSA encryption does not seem to be additively homomorphic

Multiplicatively Homomorphic ElGamal

- Several PKC algorithms are also multiplicatively homomorphic: ElGamal, Goldwasser-Micali, Benaloh, and Paillier
- In ElGamal, the encryption of m_1 is obtained as the pair (c_{11}, c_{12})

$$\begin{aligned} c_{11} &= g^{r_1} \pmod{p} \\ c_{12} &= m_1 \cdot y^{r_1} \pmod{p} \end{aligned}$$

- Similarly, the encryption of m_2 is obtained as the pair (c_{21}, c_{22})

$$\begin{aligned} c_{21} &= g^{r_2} \pmod{p} \\ c_{22} &= m_2 \cdot y^{r_2} \pmod{p} \end{aligned}$$

such that the random numbers r_1 and r_2 are (obviously) different

Multiplicatively Homomorphic ElGamal

- The pairwise product of the ciphertext pairs would give

$$\begin{aligned} c_{11} \cdot c_{21} &= g^{r_1} \cdot g^{r_2} \pmod{p} \\ &= g^{r_1 + r_2} \pmod{p} \end{aligned}$$

$$\begin{aligned} c_{12} \cdot c_{22} &= (m_1 \cdot y^{r_1}) \cdot (m_2 \cdot y^{r_2}) \pmod{p} \\ &= m_1 \cdot m_2 \cdot y^{r_1 + r_2} \pmod{p} \end{aligned}$$

- Therefore, we conclude that the product pair

$$(c_{11} \cdot c_{21} \bmod p,\ c_{12} \cdot c_{22} \bmod p)$$

is the encryption of the product of the plaintexts

$$m_1 \cdot m_2 \pmod{p}$$

with the random number $r_1 + r_2$

Additively Homomorphic ElGamal

- It turns out ElGamal algorithm can be made additively homomorphic by making a small change in the method
- The additively homomorphic ElGamal encryption is also based on the large prime p and the primitive root g mod p
- The private key is $x \in \mathcal{Z}_p^*$ and the public key $y = g^x \pmod{p}$
- **Encryption** of the plaintext $m \in \mathcal{Z}_p^*$
- The sender generates a random number r and computes

$$
\begin{aligned}
c_1 &= g^r \pmod{p} \\
c_2 &= g^m \cdot y^r \pmod{p}
\end{aligned}
$$

- The encryption of m is the ciphertext pair $E(m) = (c_1, c_2)$

Additively Homomorphic ElGamal

- **Decryption** of the ciphertext (c_1, c_2)
- The receiver has the private key x and computes u_1 and u_2 as

$$
\begin{aligned}
u_1 &= c_1^x = (g^r)^x = (g^x)^r = y^r \pmod{p} \\
u_2 &= u_1^{-1} \cdot c_2 = y^{-r} \cdot (g^m \cdot y^r) = g^m \pmod{p}
\end{aligned}
$$

  which are found as $u_1 = y^r$ and $u_2 = g^m$
- However, the legitimate owner of the private cannot decrypt the ciphertext pair (c_1, c_2) to obtain m
- In order to find m from $u_2 = g^m$, the receiver needs to solve a DLP, which is an intractable problem

Additively Homomorphic ElGamal

- If we ignore the problem that the legitimate owner of the private key cannot decrypt the ciphertext pair (c_1, c_2) to obtain m, we can check to see that this new version of ElGamal is additively homomorphic
- **Element addition $\oplus$ operation:** Given $E(m) = (c_1, c_2)$ and $E(m') = (c'_1, c'_2)$ for arbitrary plaintexts m and m', we compute $E(m) \oplus E(m') = (c''_1, c''_2)$ using

$$\begin{aligned} c''_1 &= c_1 \cdot c'_1 \pmod{p} \\ c''_2 &= c_2 \cdot c'_2 \pmod{p} \end{aligned}$$

- Then, it is easy to see that the ciphertext pair (c''_1, c''_2) is the encryption of $E(m + m')$

Element Addition

- We have indeed obtained

$$\begin{aligned}
c_1 &= g^r \pmod{p} \\
c_2 &= g^m \cdot y^r \pmod{p} \\
c_1' &= g^{r'} \pmod{p} \\
c_2' &= g^{m'} \cdot y^{r'} \pmod{p} \\
c_1'' &= g^r \cdot g^{r'} = g^{r+r'} \pmod{p} \\
c_2'' &= (g^m \cdot y^r) \cdot (g^{m'} \cdot y^{r'}) = g^{m+m'} \cdot y^{r+r'} \pmod{p}
\end{aligned}$$

- In other words, the pair (c_1'', c_2'') is the encryption of $m + m'$ with the random number $r + r'$

$$E(m) \oplus E(m) = E(m + m') = (c_1'', c_2'') = (g^{r''}, g^{m+m'} \cdot y^{r''})$$

Scalar-by-Element Multiplication

- Given an arbitrary scalar α and the element $E(m) = (c_1, c_2)$, the scalar-by-element multiplication requires the computation of

$$\begin{aligned} c_1' &= c_1^{\alpha} \pmod{p} \\ c_2' &= c_2^{\alpha} \pmod{p} \end{aligned} \tag{1}$$

- This gives $E(\alpha\, m)$ since

$$\begin{aligned} c_1' &= c_1^{\alpha} = g^{r\,\alpha} = g^{\alpha r} \pmod{p} \\ c_2' &= c_2^{\alpha} = (g^{m} \cdot y^{r})^{\alpha} = g^{\alpha\, m} \cdot y^{\alpha\, r} \pmod{p} \end{aligned}$$

- In other words, the pair (c_1', c_2') is the encryption of $\alpha\, m$ since, for some random $r' = \alpha\, r$, we have

$$E(\alpha\, m) = (c_1', c_2') = (g^{r'}, g^{\alpha\, m} \cdot y^{r'})$$

Additively Homomorphic ElGamal

- Therefore, we see that this new definition of the ElGamal encryption algorithm has two basic properties of additively homomorphic functions: Element Addition and Scalar-by-Element Multiplication
- However, we also know that given a pair of ciphertext (c_1, c_2), the legitimate owner of the private key can only obtain $g^m \pmod{p}$ by decryption, but cannot obtain m
- However, interestingly, while the legitimate user cannot decrypt (c_1, c_2) to obtain m, she can discover whether the plaintext is zero: $m = 0$
- In other words, if $m = 0$, the legitimate owner of the private key can decrypt it

Decryption of Zero in Additively Homomorphic ElGamal

- **Encryption of $m = 0$:** The sender generates a random number r and computes the ciphertext:

$$\begin{aligned} c_1 &= g^r \pmod{p} \\ c_2 &= g^0 \cdot y^r = y^r \pmod{p} \end{aligned}$$

- **Decryption for $m = 0$:** The receiver computes u_1 and u_2

$$\begin{aligned} u_1 &= c_1^x = y^r \pmod{p} \\ u_2 &= u_1^{-1} \cdot c_2 = y^{-r} \cdot y^r = 1 \pmod{p} \end{aligned}$$

- The receiver decides whether $m = 0$ by checking if $u_2 = 1$

$$u_2 = 1 \Rightarrow m = 0$$

Decryption of Zero in Additively Homomorphic ElGamal

- The decryption of zero property allows one to check for equality of two plaintexts whose ciphertexts are given
- This operation however requires access to the decryption key
- It is important to realize that ElGamal encryption algorithm is randomized: every encryption (even, of the same plaintext) involves a different random number r
- Therefore, the encryption of the same m will produce a different ciphertext pair at each encryption
- Given $m = m'$ with $E(m) = (c_1, c_2)$ and $E(m') = (c_1', c_2')$, the cipher text pairs are not equal $c_1 \neq c_1'$ and $c_2 \neq c_2'$

Equality Checking Operation

- Given $E(m) = (c_1, c_2)$ and $E(\hat{m}) = (\hat{c}_1, \hat{c}_2)$, the equality checking determines if $m = \hat{m}$
- **First:** we perform scalar-by-element multiplication of $E(\hat{m})$ using the scalar $\alpha = -1$ to obtain $E(-\hat{m}) = (c_1', c_2')$

$$\begin{aligned} c_1' &= \hat{c}_1^{\alpha} = \hat{c}_1^{-1} \pmod{p} \\ c_2' &= \hat{c}_2^{\alpha} = \hat{c}_2^{-1} \pmod{p} \end{aligned}$$

- This gives $E(-\hat{m}) = (c_1', c_2')$

Equality Checking Operation

- **Second:** we perform the element addition operation $E(m) + E(-\hat{m})$ on the given ciphertext pairs, and

$$\begin{aligned} c_1'' &= c_1 \cdot c_1' \pmod{p} \\ c_2'' &= c_2 \cdot c_2' \pmod{p} \end{aligned} \tag{2}$$

- Therefore, we obtain the ciphertext of the encrypted sum $E(m - \hat{m}) = (c_1'', c_2'')$
- Now, if $m = \hat{m}$, this ciphertext pair must be the encryption of 0

Additively Homomorphic ElGamal

- **Third:** the legitimate user performs Decryption of Zero operation
- This is accomplished using the operation steps:

$$
\begin{aligned}
u_1 &= (c_1'')^x \pmod p \\
u_2 &= u_1^{-1} \cdot c_2'' \pmod p \\
u_2 = 1 &\Rightarrow m = \hat{m}
\end{aligned}
$$

- First and Second steps are accomplished without having access to the private key, and the resulting pair (c_1'', c_2'') is sent to the owner of the private key, who performs Decryption of Zero operation and decides if $m = \hat{m}$
- This property allows secure search