Use the aforementioned CPA security and secure MAC to design a provably CCA-secure encryption scheme

To obtain a CPA-secure encryption scheme using PRF, Let F be a PRF. First, one needs to define a private-key encryption scheme for messages of length n.
The encryption scheme is a collection of three algorithms, namely, key generation algorithm (Gen), encryption algorithm (Enc) and decryption algorithm (Dec). The Gen gives key to both the sender and receiver, the Enc with key, plain text and local randomness gives the ciphertext as the output, and Dec takes the ciphertext and key to output the message.
For the CPA-secure encryption :

1. Gen : on input 1^n, choose k <- $\{0,1\}^n$ uniformly at random and output it as the key
2. Enc : on input a key where k is belonging to $\{0,1\}^n$ and a message 'm' belonging to $\{0,1\}^n$, choose r <- $\{0,1\}^n$ uniformly at random and output the ciphertext
   $c := \langle r, F_k(r) \oplus m \rangle$
3. Dec : on input a key 'k' belonging to $\{0,1\}^n$ and a ciphertext $c = \langle r,s \rangle$, output the plaintext message
   $m := F_k(r) \oplus s$

By following the above mentioned instructions, a CPA-secure encryption scheme from any pseudorandom function can be obtained.

A MAC is Message Authentication Codes. The components of the authentication protocol involves :

1. A key generation algorithm that returns a secret key 'k'
2. A MAC generating algorithm that returns a tag for a given message 'm' where the tag 't' = $MAC_k(m)$
3. A verification algorithm that returns a bit b = $Verify_k(m_1, t_1)$, given a message m_1 and a tag t_1.
4. If the message is not modified then with high probability, the value of b is true otherwise false.

A MAC(Gen, MAC, Verify) is secure if for all probabilistic polynomial-time adversaries A:
Pr[MAC-Game(n) = 1] <= negl(n)

To build a provably secure CCA-secure private-key encryption scheme(Gen', Enc', Dec'), the following instructions will be helpful.
Let a private-key encryption scheme be (Gen_E, Enc, Dec) and a message authentication code be (Gen_M, Mac, Vrfy)

1. Gen' : on input 1^n, run $Gen_E(1^n)$ and $Gen_M(1^n)$ to obtain keys k_1, k_2, respectively.

2. Enc' : on input a key (k_1,k_2) and a plaintext message m, compute c <- Enc_{k_1} (m) and t <- Mac_{k_2}(c) and output the ciphertext <c,t>
3. Dec' : on input a key (k_1, k_2) and a ciphertext <c,t>, first check whether $Vrfy_{k\_2}$(c,t) = 1. If yes, then output Dec_{k_1}(c).