Simple Asymmetric Encryption Algorithm

Project

Design and implement a simple asymmetric encryption system that executes the following functions:

1. **Key-pair generation**. Given the (secret) private key S_K (random number) in the range 1 to $p-1$, generate the corresponding public key
$$P_K = p - S_k$$

2. **Encryption**. Given a plaintext byte $P[i]$ encrypt it (i.e., generate the corresponding ciphertext byte $C[i]$) according to the following encryption law:
$$C[i] = (P[i] + P_K) \, mod \, p$$

3. **Decryption**. Given a ciphertext byte $C[i]$ decrypt it (i.e., generate the corresponding plaintext byte $P[i]$) according to the following decryption law:
$$P[i] = (C[i] + S_K) \, mod \, p$$

where:

$C[i]$	is the 8-bit ASCII code of the ith character of ciphertext;
$P[i]$	is the 8-bit ASCII code of the ith character of plaintext;
P_K	is the public key;
S_K	is the (secret) private key;
p	is the modulo equal to 223.

Figure 1 shows how the simple asymmetric encryption algorithm works.

Referring to figure 1, Walt (left side) and Jesse (right side) want to communicate using the simple asymmetric encryption scheme proposed above:

1. Walt generates his key pair ($P_{K,W}$, $S_{K,W}$);
2. Jesse generates his key pair ($P_{K,J}$, $S_{K,J}$);
3. Walt and Jesse exchange their public keys;
4. Walt encrypts each character of the plaintext $P_W[i]$ to obtain the cyphertext $C_W[i]$ using the Jesse's public key of $P_{K,J}$. Then, he sends the encrypted message to Jesse;
5. Jesse decrypts the received message using his secret private key $S_{K,J}$;
6. Jesse encrypts each character of the plaintext $P_j[i]$ (different to $P_K[i]$) to obtain the cyphertext $C_J[i]$ using the Walt's public key $P_{K,W}$. Then, he sends the encrypted message to Walt;
7. Walt decrypts the received message using his secret private key $S_{K,W}$;

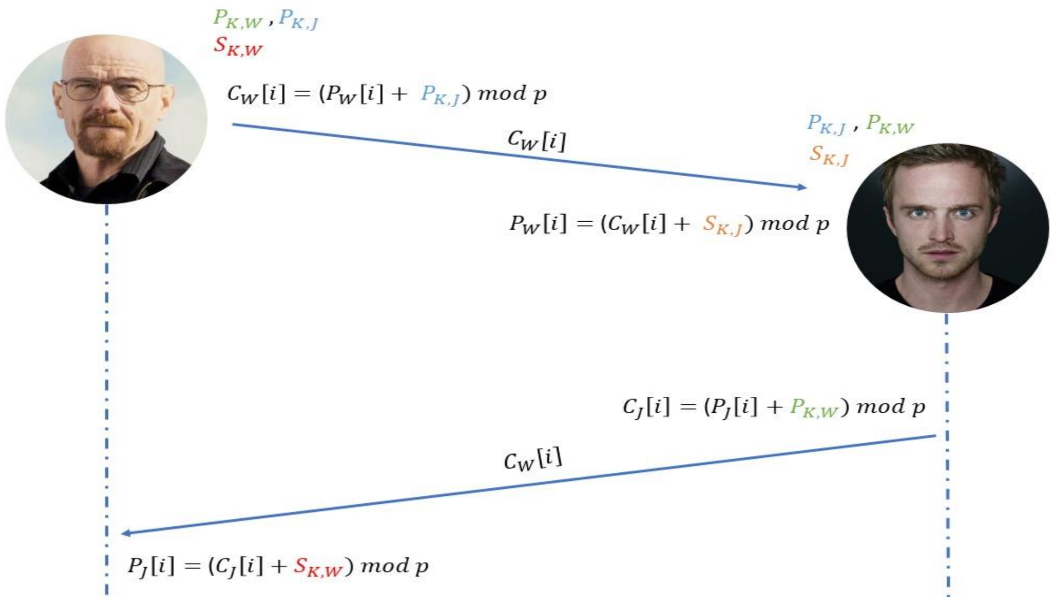
$P_{K,W}$, $P_{K,J}$
$S_{K,W}$
$C_W[i] = (P_W[i] + P_{K,J}) \ mod \ p$
$C_W[i]$
$P_{K,J}$, $P_{K,W}$
$S_{K,J}$
$P_W[i] = (C_W[i] + S_{K,J}) \ mod \ p$
$C_J[i] = (P_J[i] + P_{K,W}) \ mod \ p$
$C_W[i]$
$P_J[i] = (C_J[i] + S_{K,W}) \ mod \ p$

Figure 1: Simple asymmetric encryption scheme

Additional design specifications

- The module shall support all the 3 functions, but it must execute one function at time;
- The module shall have an asynchronous active-low reset port;
- The module interface shall include input flags to be driven as it follows: 1’b1, when the corresponding input data on the corresponding input port is valid and stable (i.e. it can be used by the internal module logic), 1’b0, otherwise.
- The module interface shall include an output flag to be driven as it follows: 1’b1, when the corresponding output data on the corresponding output port is ready and stable (i.e., external modules can read and use it), 1’b0, otherwise.