Simple Asymmetric Encryption Algorithm v2

Referent

Email stefano.dimatteo@phd.unipi.it
Teams s.dimatteo1@studenti.unipi.it

Project

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

1. Key-pair generation: Given a secret key S_K (generated at random) comprised between *1* and (*p-1),* generate the corresponding Public key P_K:

$$P_K = (S_k + q)\ mod\ p$$

2. Encryption: Given each character of the plaintext (only 8-bit ASCII characters representing lower- case letters only) encrypt it following the encryption law:

$$C[i] = (P[i] - P_K)\ mod\ p$$

3. Decryption: Given each character of the cyphertext decrypt it following the decryption law:

$$P[i] = (C[i] + S_K + q)\ mod\ p$$

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 key;
p	is the modulo equal to 227;
q	is a parameter equal to 225.

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 Public key of Jesse $P_{K,J}$. Then, he sends the encrypted message to Jesse;
5. Jesse decrypts the received message using his Secret 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 Public key of Walt $P_{K,W}$. Then, he sends the encrypted message to Walt;
7. Walt decrypts the received message using his Secret key $S_{K,W}$;

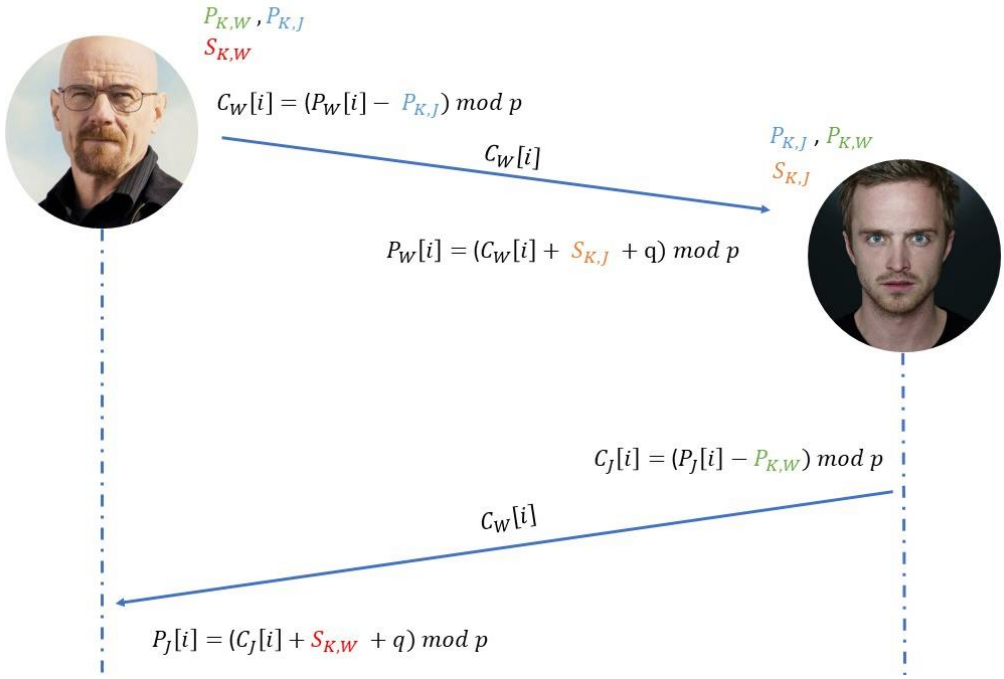
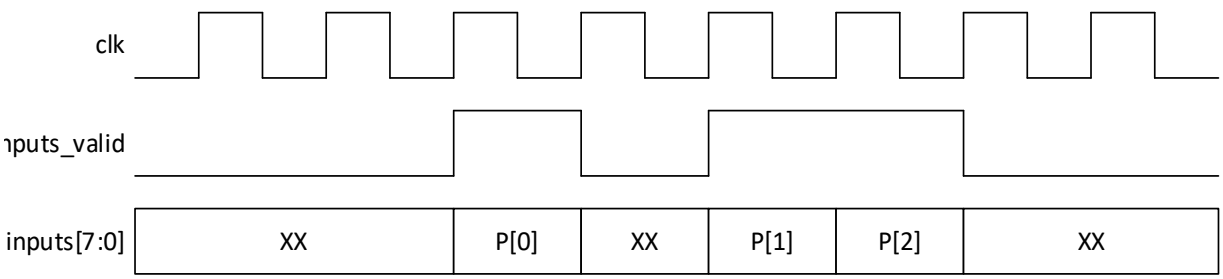


Figure 2: Simple asymmetric encryption scheme

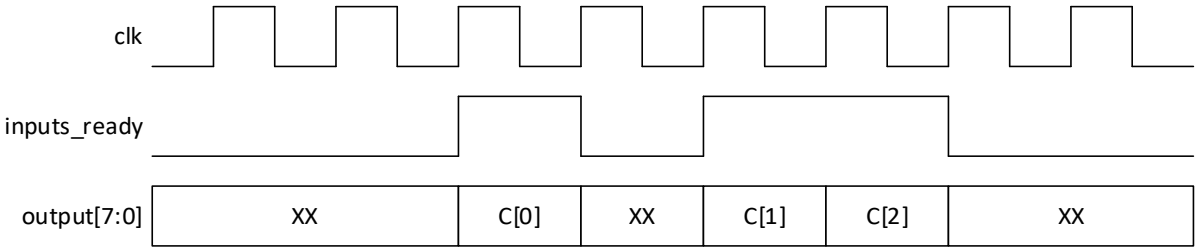
Additional design specifications

- The simple asymmetric encryption system shall execute one of the supported functions at time (Key pair generation, encryption, decryption) in one clock cycle;
- The simple asymmetric encryption system shall have an asynchronous active-low reset port;
- The plaintext character can be only 8-bit ASCII character representing lower- case letters only;
- The Secret key shall be comprised between *1* and (*p-1);*
- The simple asymmetric encryption system shall feature an input port which has to be asserted when the inputs are valid (*ptxt_valid* port): 1'b1, when inputs are valid and have to be consumed by the simple asymmetric encryption system, 1'b0, otherwise; the following waveform is expected at input interface of the simple asymmetric encryption system;



- The simple asymmetric encryption system shall feature an output port which is asserted when the output is available at the corresponding output port (*ctxt_ready* port): 1'b1, when output is valid and

has to be consumed by the logic resources linked to the simple asymmetric encryption system, 1'b0, otherwise; this flag shall be kept to logic 1 at most for one clock cycle; the following waveform is expected at the output interface of the simple asymmetric encryption system;



Hints

- Develop a model (C, Python or SystemVerilg) to test the functionality of the simple asymmetric encryption system and to generate the test vectors for the RTL simulation;
- Develop an RTL design of the simple asymmetric encryption system that executes key pair generation, encryption and decryption functions;
- Develop a testbench that instantiate the RTL design and simulate the communication scheme reported in Figure 1: for each plaintext message P, store the corresponding ciphertext C into proper object (file or string), then decrypt C and store the corresponding plaintext $P^{'}$ into proper object (file or string); compare P' with P and check they match (character by character or using string methods).