Assignment

Prof. G.Srinivasaraghavan

Date: April 5, 2024	**Submission Deadline**: April 30, 2024, Midnight	**Max Marks**: 30

Implement the Integer version of the Reed-Solomon Error Correcting code discussed in the class (Ref Section 4.6.2 of the textbook by V.Shoup). Your program should take three inputs :

1. A 'message' a to be transmitted as an (large) integer not larger than a given bound M.
2. Bound μ on the maximum fraction of units that can get corrupted during transmission. Consider every integer transmitted as part of the protocol as one 'unit' of transmission.
3. One must be able to 'transmit' multiple messages a with the same μ, M (characteristics of the transmission medium and the message domain).

The skeleton on the next page shows how your implementation is expected to work.

You are not expected to implement big-integer arithmetic from the scratch. The choice of programming language (preferably Python or C++) is left to your discretion. Recommended libraries for big-integer arithmetic:

- `https://github.com/rgroshanrg/bigint` for C++
- `https://github.com/aleaxit/gmpy` or `https://www.sympy.org/en/index.html` for Python

Specifically, you need to implement the following supporting functions:

1. The binary version of the EGCD (as discussed in the class; refer Exercise 4.10 of the textbook by V.Shoup). Treat division/multiplication by 2 as a binary shift.
2. CRT using your EGCD implementation above.
3. Implement the full Miller-Rabin algorithm (MR2 as discussed in the class). Refer Page 309 in the textbook by V.Shoup.
4. The primitives you are allowed to use from one of the libraries suggested: (i) squaring a given integer, (ii) generating a random integer in a given range, (iii) integer multiplication, division by 2, (iv) arbitrary integer addition/subtraction.
5. Use the above to complete the implementation of the four functions in the skeleton given in the next page.

Note: The implementation must be carried out in pairs.

```
/* Globals Known to both the Sender, the Transmitter (simulating the
   transmission medium) and the Receiver                                  */
$M \equiv$ large integer bound on the message
$0 < \mu < 1 \equiv$ corruption fraction
$k \equiv$ Number of 'residues' to be transmitted
$p_1, \ldots, p_k \equiv$ (Small) Primes
/* ----------------------------------------------                         */
def GlobalSetup($\mu, M$):
    /* To be implemented                                                  */
    Choose $k$
    Choose $k$ primes, $p_1, \ldots, p_k$
end
def ReedSolomonSend($a$):
    /* To be implemented                                                  */
    Compute $a_i \equiv \quad \mod p_i, \forall 1 \leq i \leq k$
    Transmit($a_1, \ldots, a_k$)
end
def Transmit($a_1, \ldots, a_k$):
    /* To be implemented                                                  */
    Choose $l$ at random from the range $[0, \mu.k]$
    Pick $l$ indexes $\mathcal{I} \equiv (I_1, \ldots, I_l)$ at random from the range $[0, k]$
    Construct $b_i$ as follows
    for $i = 1, \ldots, k$ do
        if $i \in \mathcal{I}$ then
            $b_i \leftarrow$ Random integer (different from $a_i$) chosen from $[0, (p_i - 1)]$
        else
            $b_i = a_i$
        end
    end
    return $(b_1, \ldots, b_k)$
end
def ReedSolomonReceive():
    /* To be implemented                                                  */
    $b_1, \ldots, b_k \leftarrow$ From Transmit
    Recover the original message $a$ from $b_1, \ldots, b_k$
    if Recovery Failed then
        return ERROR
    else
        return $a$
    end
end
```