LAB 2: BREAKING MERKLE-HELLMAN CRYPTOSYSTEM

An encryption scheme proposed proposed by Merkle-Hellman in 1978 [1] was shortly after broken by Adi Shamir [2]. Your task will be to implement the attack and break concrete instances. Let us start with a description of the encryption scheme.

Definition 1 (Superincreasing sequence)**.** Superincreasing sequence is a tuple $\mathbf{r} = (r_1, \ldots, r_n)$ of positive numbers such that

$$r_i > 2r_{i-1}, \quad 2 \leq i < n.$$

From the definition it follows that $r_k > r_{k-1} + \ldots r_1$ for all $2 \leq k \leq n$. For a superincreasing sequence, the following knapsack problem has an efficient solution: given $S \in \mathbb{Z}$ such that there exists $\mathbf{b} \in \{0,1\}^n$ with the property

$$S = \sum_{i=1}^{n} b_i r_i,$$

find $\mathbf{b}$. Such $\mathbf{b}$ can be found via the following algorithm

Algorithm 2 Solving superincreasing knapsack

Input: $\mathbf{r} = (r_1, \ldots, r_n), S \in \mathbb{Z}$
Output: $\mathbf{b}$ s.t. $S = \sum_{i=1}^{n} b_i r_i$

$\mathbf{b} = \mathbf{0}$
for i from n to 1 **do**
  if $S > r_i$ **then**
    $b_i = 1$
    $S = S - r_i$
  end if
end for
return $\mathbf{b}$

Let n be a public parameter. Key generation function KeyGen, encryption function Enc, and decryption function Dec are defined as follows.

$\mathsf{KeyGen}(n)$.

1. Generate a random superincreasing sequence $\mathbf{r}$ (the precise method of 'random' is irrelevant for this discussion).
2. Choose A, q such that $q > r_n$ and $\gcd(A, q) = 1$.
3. Compute the sequence $M_i = Ar_i \mod q$
4. Set $\mathrm{pk} = M, \mathrm{sk} = (A^{-1} \mod q, q, \mathbf{r})$.

$\mathsf{Enc}(\mathrm{pk}, \mathbf{m} \in \{0,1\}^n)$.

1. Return ciphertext $c = \sum_{i=1}^{n} M_i m_i \in \mathbb{Z}$

$\mathsf{Dec}(\mathrm{sk}, c)$.

1. Compute $c' = cA^{-1} \mod q$

2. Using Algorithm 1 with input $c', \mathbf{r}$ find $\mathbf{m}' \in \{0,1\}^n$.

You can convince yourself in the correctness of the scheme. There is no need to convince yourself in its security since now you'll see an efficient attack on it.

Shamir's attack recovers message $\mathbf{m}$ from the knowledge of the corresponding ciphertext and the public key. Consider the lattice generated by the *rows* of the following $(n+1) \times (n+1)$ matrix

$$B = \begin{pmatrix} 2 & 0 & 0 & \dots & 0 & M_1 \\ 0 & 2 & 0 & \dots & 0 & M_2 \\ 0 & 0 & 2 & \dots & 0 & M_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ -1 & -1 & -1 & \dots & -1 & -c \end{pmatrix}$$

Note that if $\mathbf{m}$ is the message for the ciphertext c, i.e., $c = \sum_{i=1}^{n} M_i m_i$, then $L(B)$ contains $\mathbf{t} = (2m_1 - 1, 2m_2 - 1, 2m_3 - 1, \dots, 0) \in \mathbb{Z}^{n+1}$. For a binary $\mathbf{m}$, this vector is the shortest of $L(B)$ with high probability and, for many interesting parameters, can be efficiently found via LLL.

Task

1. Download script merkle − hellman.sage. There you'll find implementations of KeyGen, Enc, Dec. You do not need to modify them.
2. Implement Shamir's attack in the function attack().
3. Check the correctness of your implementation by running

   sage -t merkle_hellman.sage

   All three tests should pass.

References

[1] Ralph Merkle and Martin Hellman, *Hiding Information and Signatures in Trapdoor Knapsacks*. IEEE Trans. Information Theory. 1978

[2] Adi Shamir, *A Polynomial Time Algorithm for Breaking the Basic Merkle-Hellman Cryptosystem*. CRYPTO. 1982