Paillier cryptosystem

- homomorphic properties
  addition: \(D(E(m_1, r_1) \times E(m_2, r_2) \mod n^2) = m_1 + m_2 \mod n\)
  multiplication: \(D(E(m_1, r_1)^{m_2} \mod n^2) = m_1m_2 \mod n\)

- exponentiation involving keys and messages
  \(p_k = (n, g); s_k = (\lambda, \mu); c = E(m, r); r \text{ random}\)
  base, exp \(\in\) int: \(r^n, r^\lambda, e^n, c^\lambda\)
  base \(\in\) int, exp \(\in\) float: \(g^m, \mu^m, c^m\)
  base \(\in\) float, exp \(\in\) int: -
  base, exp \(\in\) float: -

- problem simplification and error analysis
  approximation: \(m \in\) float \(\rightarrow\) \(m' \in\) int (by truncation)
  \(m' \leq m < m' \rightarrow\) worst case: \(m = m' + 1\)
  addition (+) and multiplication (*)
  \(a = a' + 1, b = b' + 1\)
  \(error_+ = (a + b) - (a' + b') = (a' + 1 + b' + 1) - (a' + b') = 2\)
  \(error_* = (ab) - (a'b') = (a'b' + a' + b' + 1) - (a'b') = a' + b' + 1\)
  \(\%error_+ = \frac{2}{q+r} \times 100\%\)
  \(\%error_* = \frac{a' + b' + 1}{ab} \times 100\% = \frac{a + b - 1}{ab} \times 100\%\)

<table>
<thead>
<tr>
<th>(O(a))</th>
<th>(O(b))</th>
<th>(O(%error_+))</th>
<th>(O(%error_*))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100%</td>
<td>100%</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>10%</td>
<td>20%</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>1%</td>
<td>2%</td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>0.1%</td>
<td>0.2%</td>
<td></td>
</tr>
<tr>
<td>10000</td>
<td>0.01%</td>
<td>0.02%</td>
<td></td>
</tr>
</tbody>
</table>

Paillier algorithm implementation
I finished my implementation of the Paillier cryptosystem using Manas’ suggestions. It works for positive/negative integers and for floating point numbers, which are truncated before processing.