Simulation Steps and Matrix:

In my simulation to check Salsa20 and ChaCha20 performance, I have simulated the results using the below steps.

1. I have used 7 different types of data to check its performance.
   1. Small int (ex: 12345)
   2. Big Int (ex: 123456578999999\*\*\*\*\*\*\*\*\*)
   3. Float Int (ex: 123456578.999999\*\*\*\*\*\*\*\*\*)
   4. Small String Value (30 chars)
   5. Big String Value (1000 chars)
   6. Small File data (250 kb)
   7. Big File data (50 mb)
2. Apart from Salsa20, ChaCha20 I have worked on adding other ciphers to process this data. List of ciphers used for this experiment.
   1. Salsa20
   2. ChaCha20
   3. RC4 (ARC4)
   4. AES
   5. (AES-CTR) – stream cipher implementation of AES.
3. I have gathered matrix of
   1. Memory consumption (after encryption) in kb
   2. Time takes between encryption and decryption in milli sec.
   3. Memory diff between original data and encrypted data.
4. I have run each data type with each cipher nearly 50 times, for example I have sampled the recordings when the same small int with salsa20 for 50 times, I graphs will display the performance for the above mentioned metrix.
5. All the code/sampled data results has been pushed into github
   1. <https://github.com/manogna1997/CS698>

Observation:

1. Big int Time Difference between Memory encryption and Decryption

Chart, histogram

Description automatically generated

1. Big File memory Difference

Chart, background pattern, bar chart

Description automatically generated

1. Big File Time Difference between encryption and decryption

Chart, bar chart, histogram

Description automatically generated

1. Small file memory difference

Chart, background pattern, bar chart

Description automatically generated

1. Small file time difference between encryption and decryption

Chart, bar chart

Description automatically generated

1. Small int memory difference

Chart, histogram

Description automatically generated

1. Small int time difference between encryption and decryption

Chart, histogram

Description automatically generated