# Trapdoor based Searchable Encryption Scheme

Sunitha Narayan (snaray18/snaray19)

12 December 2019

Code available @ https://github.com/sunithan29/Searchable-Symmetric-Encryption

# Cryptographic primitives used

- AES Encryption
  - EBC mode
  - CBC mode with IV
- Why we use CBC over EBC for the implementation
- Trapdoor: A  trapdoor is generated for every keyword which is in turn used in the search algorithm

# SSE Scheme Implementation

- Keygen: Key generation algorithm is run by the user to set up the scheme. It takes as input a security parameter k, and outputs a master key MK.
- Trapdoor: Generated by the client using master key MK and keyword, w as the input, and outputs the trapdoor, t of keyword, w.
- BuildIndex: The input is the document D as the input file that is to be encrypted
- SearchIndex: Now we take input as the trapdoor T for word w and the index I for the search query.

# SSE Scheme Construction

- For each unique word, compute
  - A trapdoor
  - A codeword for w
  - Insert the codeword into the index.
  - Output the encrypted index file

- BuildIndex: The input is the document D as the input file that is to be encrypted
  - Phase I: input_index: The input file is encrypted using the masterkey and index is generated
  - Phase II: Generate trapdoor: Uses masterkey and keyword as the input, and output the trapdoor of keyword.

- Search Query: Used to search the encrypted input file for specific keywords that are saved as trapdoors

# Demo

```
sn@sn:~/Desktop/Advanced_Crypto/sse$ python3 build_index.py

        Phase I: Building Index

Please input the file/directory where the master key is stored:  masterkey

Successfully added master key.

An index file will be generated for the input file.
        Please input the file to be encrypted:  input.csv
        please input the file where the keywords are listed:  keyword

The time taken to generate the input_index.csv file is
3.9344851970672607
None

        Phase II: Trapdoor Generation for a keyword

Please input the keyword you want to search for in the encrypted input file:  Month

A trapdoor for the keyword is successfully generated.
        The trapdoor is saved as keyword_trapdoor.csv file

b'\x87\xc3)8\xfe\xd7\xfe\xc1\x81\x06\xc2\xe5\xe0YB\xf4\x13\x8b.x\xec"\xcb\x0b\x0fjF\t\xe3/\x1f\xf3'
Finished
```

# Demo

Searchable Encryption

```
sn@sn:~/Desktop/Advanced_Crypto/sse$ python3 sse_query.py

Please input the index file you want to search:  input_index.csv

Please input the file where trapdoor is stored:  keyword_trapdoor.csv
The identifiers that contain the keyword are:
[2053, 2059, 17, 2075, 27, 2083, 2090, 48, 2098, 2102, 2109, 62, 2112, 67, 2117, 2118, 2136, 97, 2146, 2157, 111, 2160, 2162, 118, 119, 2171, 125, 127,
 2179, 2183, 2185, 141, 2194, 149, 152, 2201, 156, 2209, 2224, 2232, 186, 2234, 2236, 196, 199, 2259, 219, 2292, 250, 271, 273, 2322, 2326, 280, 284, 2
332, 292, 2348, 2349, 2353, 2364, 2367, 320, 2372, 325, 326, 327, 2391, 345, 350, 2399, 2400, 2405, 2406, 2407, 371, 2424, 378, 380, 2428, 385, 2435, 3
90, 2458, 410, 424, 429, 449, 453, 2512, 468, 2519, 2523, 484, 485, 495, 498, 501, 508, 2557, 2558, 513, 2573, 2574, 526, 2578, 531, 2594, 549, 2602, 5
58, 560, 561, 567, 2619, 2624, 577, 2634, 2639, 2643, 2647, 2651, 608, 619, 622, 2671, 2681, 2685, 2687, 2692, 2707, 2712, 665, 683, 2732, 690, 691, 27
38, 2742, 2751, 2752, 2755, 710, 714, 715, 2762, 727, 732, 737, 738, 740, 2794, 752, 754, 757, 2806, 2815, 2820, 774, 784, 2834, 2838, 791, 793, 2842,
2850, 2855, 2862, 815, 826, 2885, 2888, 2890, 843, 848, 849, 854, 861, 2912, 866, 867, 870, 2918, 2922, 886, 887, 2937, 890, 892, 2941, 2942, 2946, 904
, 915, 2966, 2967, 2976, 929, 940, 945, 2997, 949, 3001, 3003, 978, 985, 991, 3045, 3046, 3051, 1007, 1009, 1022, 1034, 1038, 3092, 1046, 3095, 1051, 3
102, 1062, 1077, 3126, 1080, 1092, 1095, 1096, 3149, 1102, 1111, 1116, 1131, 3181, 1143, 1151, 1156, 1158, 1162, 1163, 3214, 1169, 1171, 1179, 3234, 11
87, 1196, 3244, 1201, 1207, 1217, 1222, 1223, 1227, 1235, 1242, 3295, 3296, 1250, 3304, 1262, 3316, 1268, 1275, 3330, 3337, 1290, 1303, 3354, 3363, 336
9, 3378, 1339, 3393, 3395, 3402, 1355, 3405, 1361, 1363, 1370, 3424, 1380, 3429, 1401, 3457, 1410, 3460, 1417, 1418, 1426, 1433, 1434, 1435, 3486, 3492
, 3499, 1455, 3514, 3526, 1480, 1482, 1485, 1486, 1489, 1493, 1494, 3545, 3557, 3564, 3589, 1541, 1543, 3596, 1558, 1572, 3623, 3628, 1586, 3635, 3636,
 1588, 1591, 3641, 3642, 3644, 3650, 3652, 1610, 3668, 3672, 1637, 3695, 1651, 3705, 3706, 3709, 1673, 3721, 3723, 1691, 3739, 1695, 3747, 3766, 3768,
1722, 3776, 3780, 3782, 3786, 3792, 1745, 1747, 1755, 1761, 1762, 3816, 1771, 3824, 1784, 1805, 1806, 3864, 3865, 1821, 3872, 1829, 3885, 1840, 3894, 3
905, 1859, 3910, 3914, 1878, 3931, 3934, 3941, 1896, 1899, 3948, 1902, 3951, 3955, 3962, 1924, 1925, 3984, 3993, 3994, 3995, 3996, 3999, 1981, 1983, 19
87, 2015, 2037, 2038, 2040]
```

# Extensions

- Locating Words Within Documents. Instead of using an index for every document, we can divide documents into chunks and create indexes for each chunk. With this modification, words can be located with chunk size granularity within a document.
- Implement sse using bloom filter choosing suitable Filter Parameters and small array size to make it efficient.
- Extend this prototype to support search in multi-user settings.
- Use other AES encryption modes that provide better security and privacy like CBC-MAC, CMAC, HMAC, and GMAC