

Лабораторная работа № 7

Элементы криптографии.
Однократное гаммирование

Форис Анастасия Дмитриевна

10.2024

Российский университет дружбы народов, Москва, Россия

Освоить на практике применение режима однократного гаммирования.

Я начала процесс с импорта необходимых библиотек, неотъемлемых для криптографических операций, которые последовали. Впоследствии я разработала специализированную функцию, способную выполнять операции побитового XOR на двух строках, облегчая процессы шифрования и дешифрования. Исходный текст, служащий оригинальным незашифрованным текстом, играл ключевую роль в последующих шагах. Ключ, тщательно созданный для соответствия длине исходного текста, генерировался для внесения дополнительного уровня безопасности в механизм шифрования.

Применяя ранее разработанную функцию, я легко получила шифротекст, предполагая знание как исходного текста, так и созданного ключа. Этот шифротекст, представляющий собой зашифрованную форму оригинального текста, служил важным посредником в криптографических процедурах. Кроме того, я продемонстрировала обратимость процесса, используя ту же функцию для получения исходного текста при наличии информации как о шифротексте, так и о ключе.

Интригующим образом, криптографический процесс не ограничивался лишь шифрованием и дешифрованием, но также распространялся на восстановление ключа. Снова воспользовавшись той же функцией, я успешно извлекла ключ, обладая информацией как о исходном тексте, так и соответствующем шифротексте. Этот всесторонний подход к криптографическим операциям подчеркивает взаимосвязанную природу шифрования, дешифрования и управления ключами в обеспечении целостности и конфиденциальности чувствительной информации.

Выполнение лабораторной работы 4

```

100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000

```

Рис. 1: Приложение, реализующее режим однократного гаммирования

```

10  def __init__(self, x, y):
11      self.x = x
12      self.y = y
13
14  def __str__(self):
15      return f"Point({self.x}, {self.y})"
16
17  def __repr__(self):
18      return f"Point({self.x}, {self.y})"
19
20  def __add__(self, other):
21      return Point(self.x + other.x, self.y + other.y)
22
23  def __sub__(self, other):
24      return Point(self.x - other.x, self.y - other.y)
25
26  def __mul__(self, other):
27      return Point(self.x * other.x, self.y * other.y)
28
29  def __div__(self, other):
30      return Point(self.x / other.x, self.y / other.y)
31
32  def __eq__(self, other):
33      return self.x == other.x and self.y == other.y
34
35  def __neq__(self, other):
36      return not self.__eq__(other)
37
38  def __lt__(self, other):
39      return self.x < other.x and self.y < other.y
40
41  def __gt__(self, other):
42      return self.x > other.x and self.y > other.y
43
44  def __le__(self, other):
45      return self.x <= other.x and self.y <= other.y
46
47  def __ge__(self, other):
48      return self.x >= other.x and self.y >= other.y
49
50  def __hash__(self):
51      return hash((self.x, self.y))
52
53  def __getitem__(self, index):
54      return self.x if index == 0 else self.y
55
56  def __setitem__(self, index, value):
57      if index == 0:
58          self.x = value
59      else:
60          self.y = value
61
62  def __delitem__(self, index):
63      if index == 0:
64          self.x = None
65      else:
66          self.y = None
67
68  def __contains__(self, item):
69      return item in (self.x, self.y)
70
71  def __iter__(self):
72      return iter([self.x, self.y])
73
74  def __reversed__(self):
75      return iter([self.y, self.x])
76
77  def __len__(self):
78      return 2
79
80  def __sizeof__(self):
81      return 16
82
83  def __dir__(self):
84      return ['x', 'y', '__add__', '__sub__', '__mul__', '__div__', '__eq__', '__neq__', '__lt__', '__gt__', '__le__', '__ge__', '__hash__', '__getitem__', '__setitem__', '__delitem__', '__contains__', '__iter__', '__reversed__', '__len__', '__sizeof__', '__dir__']
85
86  def __copy__(self):
87      return Point(self.x, self.y)
88
89  def __deepcopy__(self, memo):
90      return Point(self.x, self.y)
91
92  def __getattr__(self, name):
93      if name in ('x', 'y'):
94          return getattr(self, name)
95      else:
96          raise AttributeError(f"'Point' object has no attribute '{name}'")
97
98  def __setattr__(self, name, value):
99      if name in ('x', 'y'):
100         setattr(self, name, value)
101     else:
102         raise AttributeError(f"'Point' object has no attribute '{name}'")
103
104  def __delattr__(self, name):
105      if name in ('x', 'y'):
106         delattr(self, name)
107     else:
108         raise AttributeError(f"'Point' object has no attribute '{name}'")
109
110  def __call__(self):
111      return self
112
113  def __bool__(self):
114      return True
115
116  def __format__(self, format_spec):
117      return f"Point({self.x}, {self.y})"
118
119  def __round__(self):
120      return Point(round(self.x), round(self.y))
121
122  def __trunc__(self):
123      return Point(int(self.x), int(self.y))
124
125  def __floor__(self):
126      return Point(math.floor(self.x), math.floor(self.y))
127
128  def __ceil__(self):
129      return Point(math.ceil(self.x), math.ceil(self.y))
130
131  def __abs__(self):
132      return Point(abs(self.x), abs(self.y))
133
134  def __neg__(self):
135      return Point(-self.x, -self.y)
136
137  def __pos__(self):
138      return Point(self.x, self.y)
139
140  def __invert__(self):
141      return Point(~self.x, ~self.y)
142
143  def __complex__(self):
144      return complex(self.x, self.y)
145
146  def __float__(self):
147      return float(self.x + self.y * 1j)
148
149  def __int__(self):
150      return int(self.x + self.y * 1j)
151
152  def __complex___.self):
153      return complex(self.x, self.y)
154
155  def __float___.self):
156      return float(self.x + self.y * 1j)
157
158  def __int___.self):
159      return int(self.x + self.y * 1j)
160
161  def __complex___.self):
162      return complex(self.x, self.y)
163
164  def __float___.self):
165      return float(self.x + self.y * 1j)
166
167  def __int___.self):
168      return int(self.x + self.y * 1j)
169
170  def __complex___.self):
171      return complex(self.x, self.y)
172
173  def __float___.self):
174      return float(self.x + self.y * 1j)
175
176  def __int___.self):
177      return int(self.x + self.y * 1j)
178
179  def __complex___.self):
180      return complex(self.x, self.y)
181
182  def __float___.self):
183      return float(self.x + self.y * 1j)
184
185  def __int___.self):
186      return int(self.x + self.y * 1j)
187
188  def __complex___.self):
189      return complex(self.x, self.y)
190
191  def __float___.self):
192      return float(self.x + self.y * 1j)
193
194  def __int___.self):
195      return int(self.x + self.y * 1j)
196
197  def __complex___.self):
198      return complex(self.x, self.y)
199
200  def __float___.self):
201      return float(self.x + self.y * 1j)
202
203  def __int___.self):
204      return int(self.x + self.y * 1j)
205
206  def __complex___.self):
207      return complex(self.x, self.y)
208
209  def __float___.self):
210      return float(self.x + self.y * 1j)
211
212  def __int___.self):
213      return int(self.x + self.y * 1j)
214
215  def __complex___.self):
216      return complex(self.x, self.y)
217
218  def __float___.self):
219      return float(self.x + self.y * 1j)
220
221  def __int___.self):
222      return int(self.x + self.y * 1j)
223
224  def __complex___.self):
225      return complex(self.x, self.y)
226
227  def __float___.self):
228      return float(self.x + self.y * 1j)
229
230  def __int___.self):
231      return int(self.x + self.y * 1j)
232
233  def __complex___.self):
234      return complex(self.x, self.y)
235
236  def __float___.self):
237      return float(self.x + self.y * 1j)
238
239  def __int___.self):
240      return int(self.x + self.y * 1j)
241
242  def __complex___.self):
243      return complex(self.x, self.y)
244
245  def __float___.self):
246      return float(self.x + self.y * 1j)
247
248  def __int___.self):
249      return int(self.x + self.y * 1j)
250
251  def __complex___.self):
252      return complex(self.x, self.y)
253
254  def __float___.self):
255      return float(self.x + self.y * 1j)
256
257  def __int___.self):
258      return int(self.x + self.y * 1j)
259
260  def __complex___.self):
261      return complex(self.x, self.y)
262
263  def __float___.self):
264      return float(self.x + self.y * 1j)
265
266  def __int___.self):
267      return int(self.x + self.y * 1j)
268
269  def __complex___.self):
270      return complex(self.x, self.y)
271
272  def __float___.self):
273      return float(self.x + self.y * 1j)
274
275  def __int___.self):
276      return int(self.x + self.y * 1j)
277
278  def __complex___.self):
279      return complex(self.x, self.y)
280
281  def __float___.self):
282      return float(self.x + self.y * 1j)
283
284  def __int___.self):
285      return int(self.x + self.y * 1j)
286
287  def __complex___.self):
288      return complex(self.x, self.y)
289
290  def __float___.self):
291      return float(self.x + self.y * 1j)
292
293  def __int___.self):
294      return int(self.x + self.y * 1j)
295
296  def __complex___.self):
297      return complex(self.x, self.y)
298
299  def __float___.self):
300      return float(self.x + self.y * 1j)
301
302  def __int___.self):
303      return int(self.x + self.y * 1j)
304
305  def __complex___.self):
306      return complex(self.x, self.y)
307
308  def __float___.self):
309      return float(self.x + self.y * 1j)
310
311  def __int___.self):
312      return int(self.x + self.y * 1j)
313
314  def __complex___.self):
315      return complex(self.x, self.y)
316
317  def __float___.self):
318      return float(self.x + self.y * 1j)
319
320  def __int___.self):
321      return int(self.x + self.y * 1j)
322
323  def __complex___.self):
324      return complex(self.x, self.y)
325
326  def __float___.self):
327      return float(self.x + self.y * 1j)
328
329  def __int___.self):
330      return int(self.x + self.y * 1j)
331
332  def __complex___.self):
333      return complex(self.x, self.y)
334
335  def __float___.self):
336      return float(self.x + self.y * 1j)
337
338  def __int___.self):
339      return int(self.x + self.y * 1j)
340
341  def __complex___.self):
342      return complex(self.x, self.y)
343
344  def __float___.self):
345      return float(self.x + self.y * 1j)
346
347  def __int___.self):
348      return int(self.x + self.y * 1j)
349
350  def __complex___.self):
351      return complex(self.x, self.y)
352
353  def __float___.self):
354      return float(self.x + self.y * 1j)
355
356  def __int___.self):
357      return int(self.x + self.y * 1j)
358
359  def __complex___.self):
360      return complex(self.x, self.y)
361
362  def __float___.self):
363      return float(self.x + self.y * 1j)
364
365  def __int___.self):
366      return int(self.x + self.y * 1j)
367
368  def __complex___.self):
369      return complex(self.x, self.y)
370
371  def __float___.self):
372      return float(self.x + self.y * 1j)
373
374  def __int___.self):
375      return int(self.x + self.y * 1j)
376
377  def __complex___.self):
378      return complex(self.x, self.y)
379
380  def __float___.self):
381      return float(self.x + self.y * 1j)
382
383  def __int___.self):
384      return int(self.x + self.y * 1j)
385
386  def __complex___.self):
387      return complex(self.x, self.y)
388
389  def __float___.self):
390      return float(self.x + self.y * 1j)
391
392  def __int___.self):
393      return int(self.x + self.y * 1j)
394
395  def __complex___.self):
396      return complex(self.x, self.y)
397
398  def __float___.self):
399      return float(self.x + self.y * 1j)
400
401 
```

Рис. 2: Код

Вывод



В ходе выполнения данной лабораторной работы я освоила на практике применение режима однократного гаммирования.