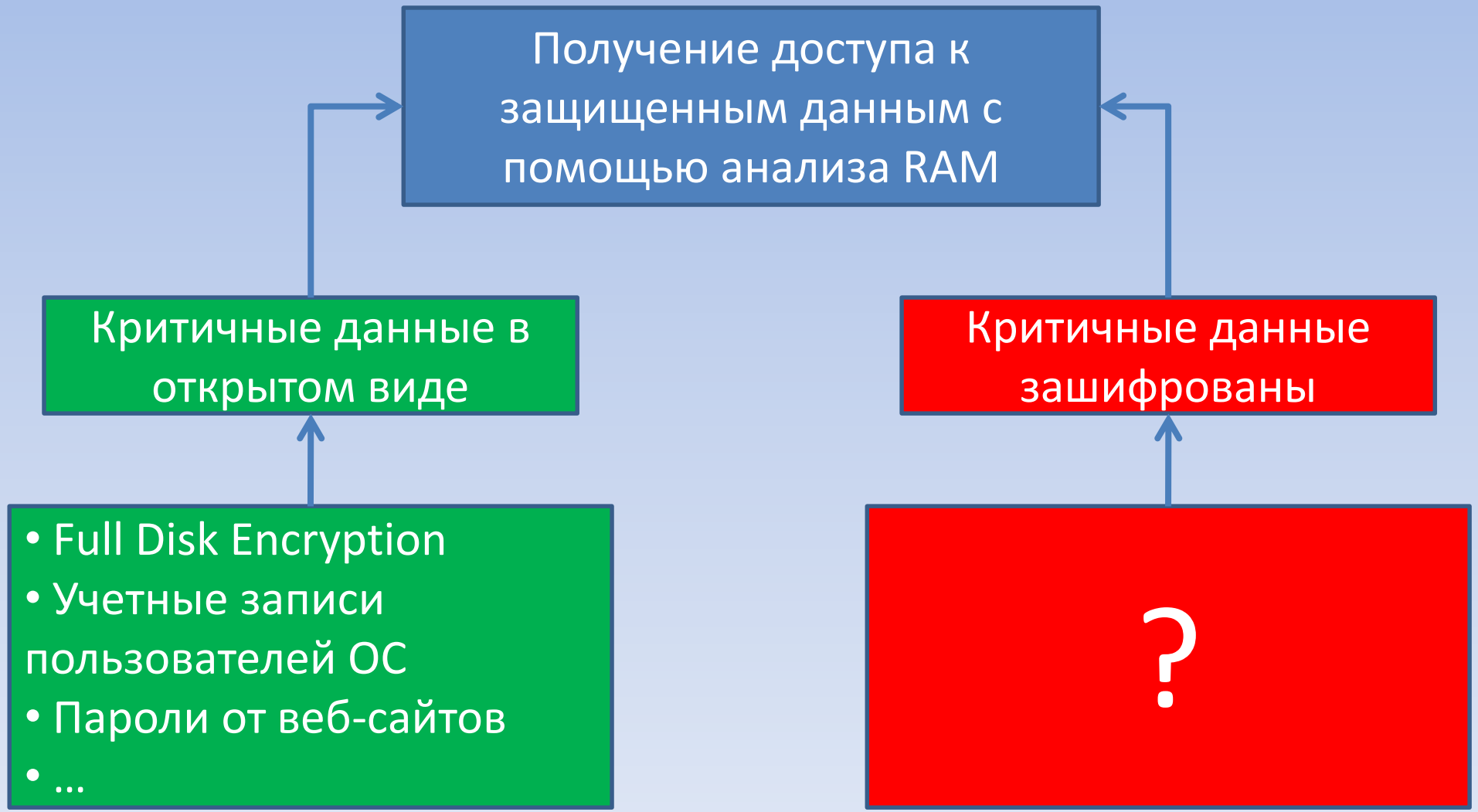
Криминалистический анализ RAM: обнаружение, расшифрование и интерпретация зашифрованных криптографических объектов и данных

Чиликов А.А., к.ф.-м.н., доцент кафедры «Информационная безопасность», МГТУ им.Баумана, Passware Inc.

Хоруженко Г.И., аспирант кафедры «Криптология и дискретная математика», НИЯУ МИФИ , Passware Inc.

19 марта 2015 г.

Актуальность



ПО, использующее механизмы защиты данных в памяти

Программное средство	Механизм защиты	Зашифрованные данные
WinRAR 5.20 for Windows	CryptProtectMemory (ОС Windows)	Пароль от архива
KeePass 1.28 (Classic)	CryptProtectMemory (ОС Windows)	Ключ шифрования БД
KeePass 2.28 (Professional)*	CryptProtectMemory (ОС Windows)	Мастер-пароль
1Password 5.1 for Mac**	Собственная реализация	Мастер-пароль

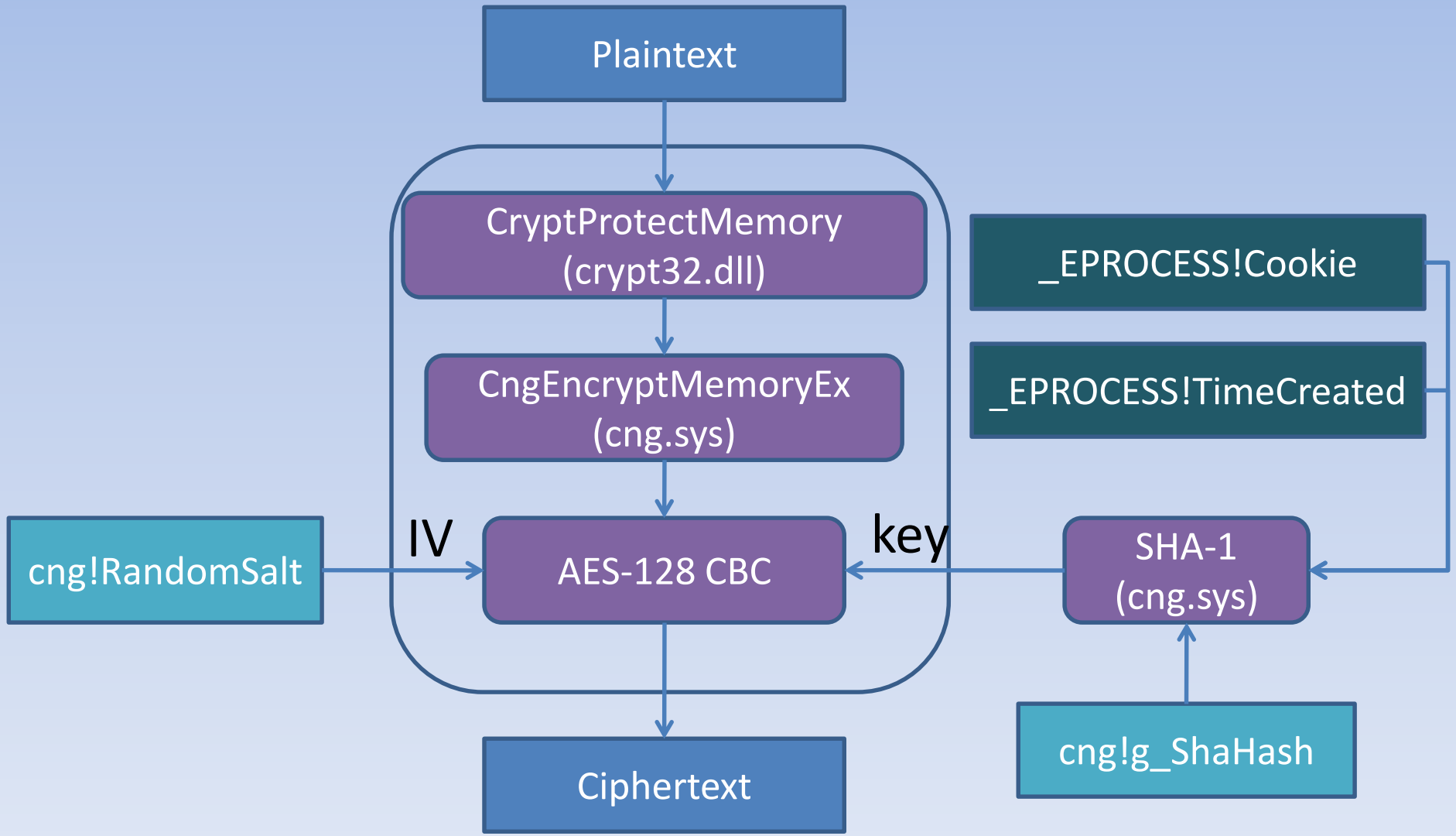
* KeePass Professional хранит ключ шифрования БД в открытом виде

** 1Password for Windows хранит ключ шифрования в открытом виде

Защита участка памяти на уровне прикладных программ в ОС Windows

- Рассматривается случай **Windows 8.1 x64**
- Пара функций **CryptProtectMemory**/ **CryptUnprotectMemory** (crypt32.dll), управление передается **CngEncryptMemoryEx** (cng.sys).
- Различные варианты защиты:
  - Отсутствие привязки (CROSS_PROCESS)
  - Привязка к пользователю (SAME_LOGON)
  - **Привязка к процессу (SAME_PROCESS)**

Схема работы функции CryptProtectMemory



Структуры cng!g_ShaHash и cng!RandomSalt

```
3: kd> db cng!g_ShaHash
fffff801`fb97dd60  19 03 40 fa 14 6b 91 e2-83 76 77 e3 a6 6e 36 3b
fffff801`fb97dd70  44 de d6 8f 29 2a c4 3c-00 00 00 00 00 00 00 00
fffff801`fb97dd80  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
fffff801`fb97dd90  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
fffff801`fb97dda0  01 23 45 67 89 ab cd ef-fe dc ba 98 76 54 32 10
fffff801`fb97ddb0  f0 e1 d2 c3 00 00 00 00-00 00 00 00 00 00 00 00
fffff801`fb97ddc0  18 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
fffff801`fb97ddd0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
```

buffer

SHA-1 registers

buffer size

Поиск в памяти структуры **cng!g_ShaHash** осуществляется по начальному заполнению регистров SHA-1 и длине буфера

cng!RandomSalt – массив размера 0x10 байт

Для каждой версии ОС Windows смещение **cng!RandomSalt** относительно **cng!g_ShaHash** разное, но для всех версий **обе структуры расположены на одной странице образа памяти**

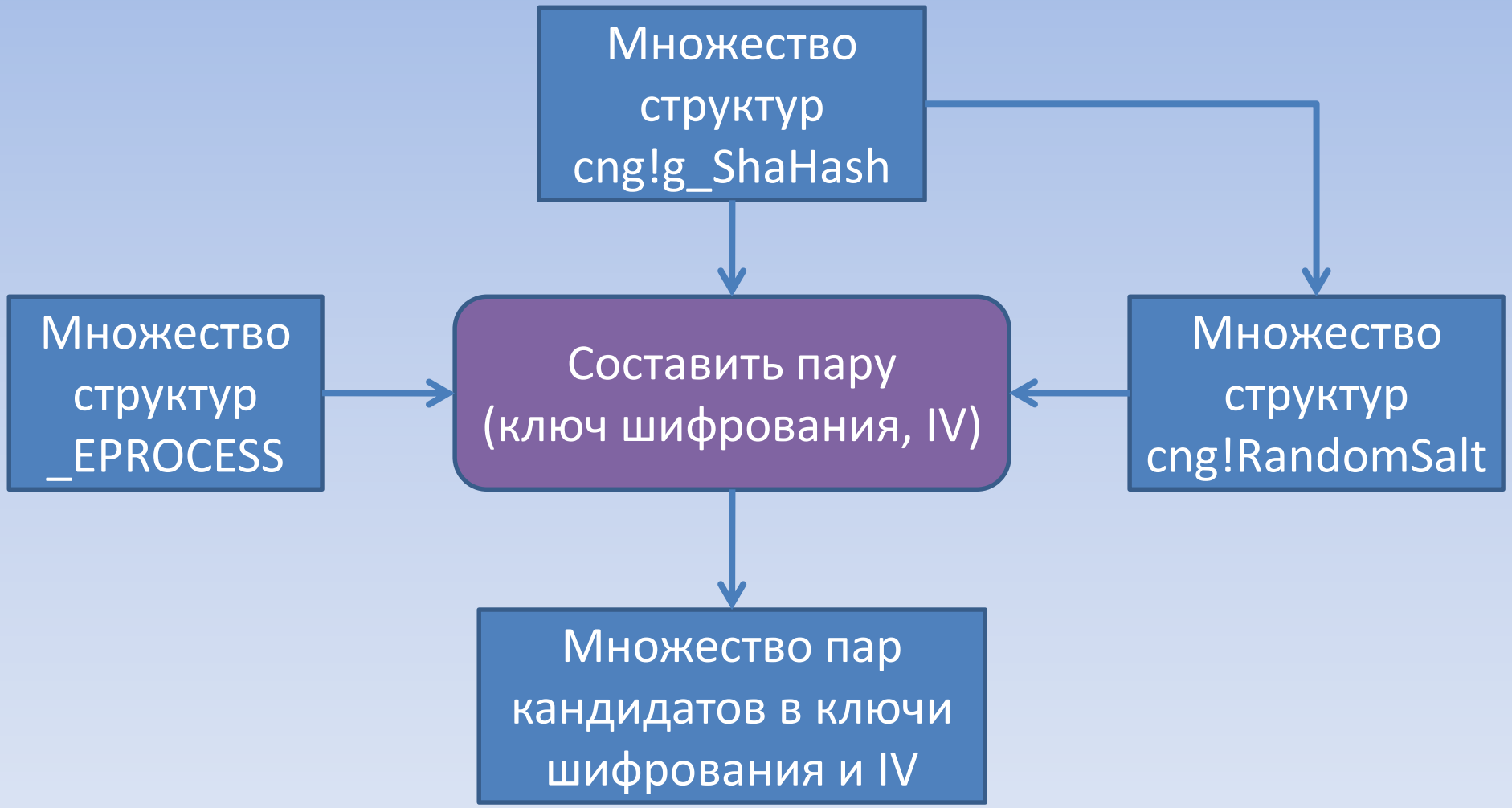
Структура _EPROCESS

- Задача поиска и восстановления структур **_EPROCESS** решена (например, ПО Volatility)
- Подход заключается в поиске сигнатуры структуры **_POOL_HEADER** и в исключении неверных вариантов на основе анализа значений полей предполагаемой структуры **_EPROCESS**

Вычисление ключа шифрования функции CryptProtectMemory

```
# первоначальное заполнение массива структуры cng!g_ShaHash
buffer = g_ShaHash.buffer

# 4-байтовое значение поля Cookie структуры _EPROCESS
buffer += process[ COOKIE ]

# 8-байтовое значение метки времени создания
# процесса из структуры _EPROCESS
buffer += process[ TIME_CREATED ]

# 0x10 - размер ключа шифрования AES-128
key = SHA1( buffer ).digest()[ 0 : 0x10 ]

# массив cng!RandomSalt
iv = RandomSalt
```

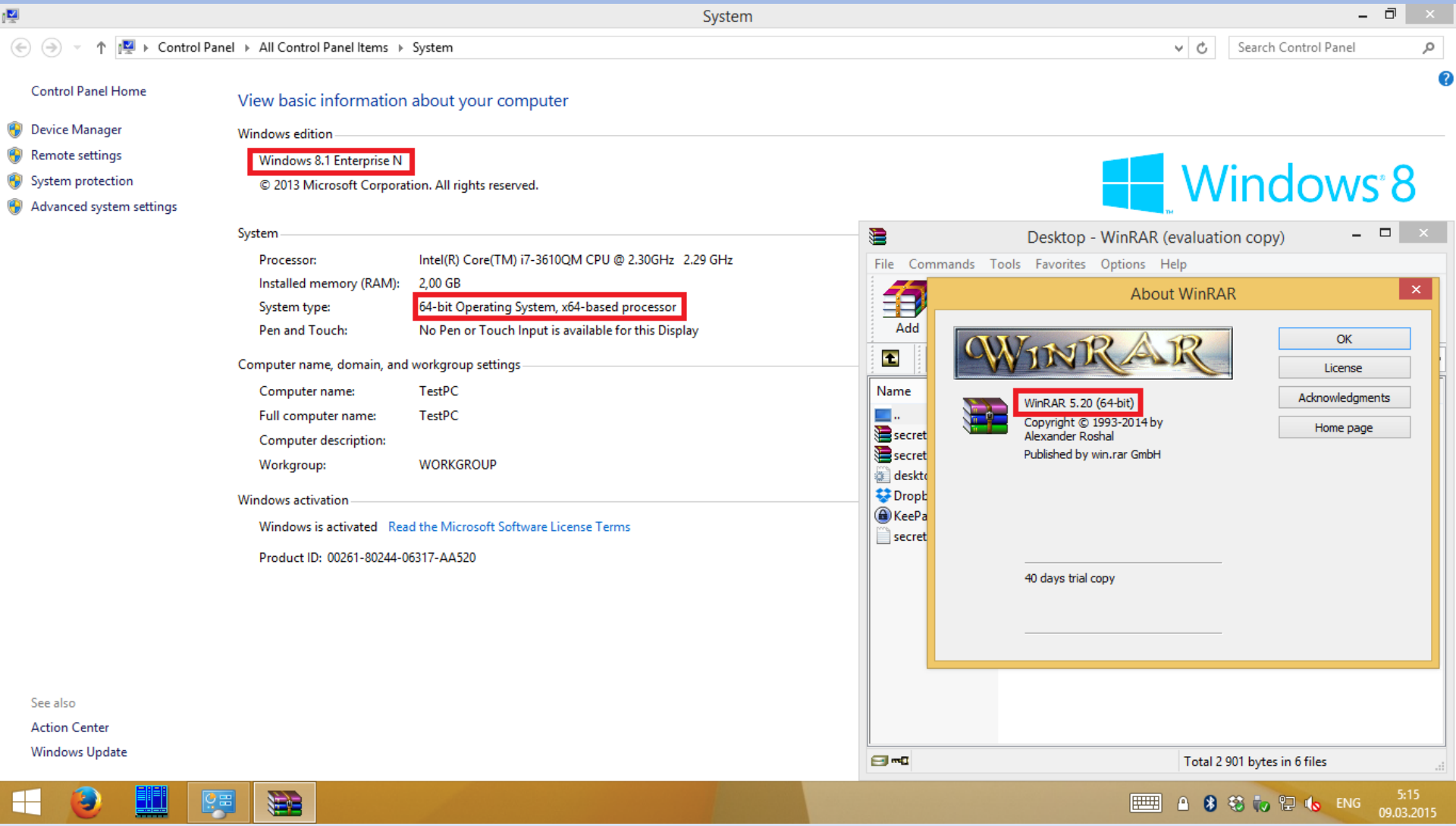
Построение множества кандидатов в ключи шифрования и векторы инициализации



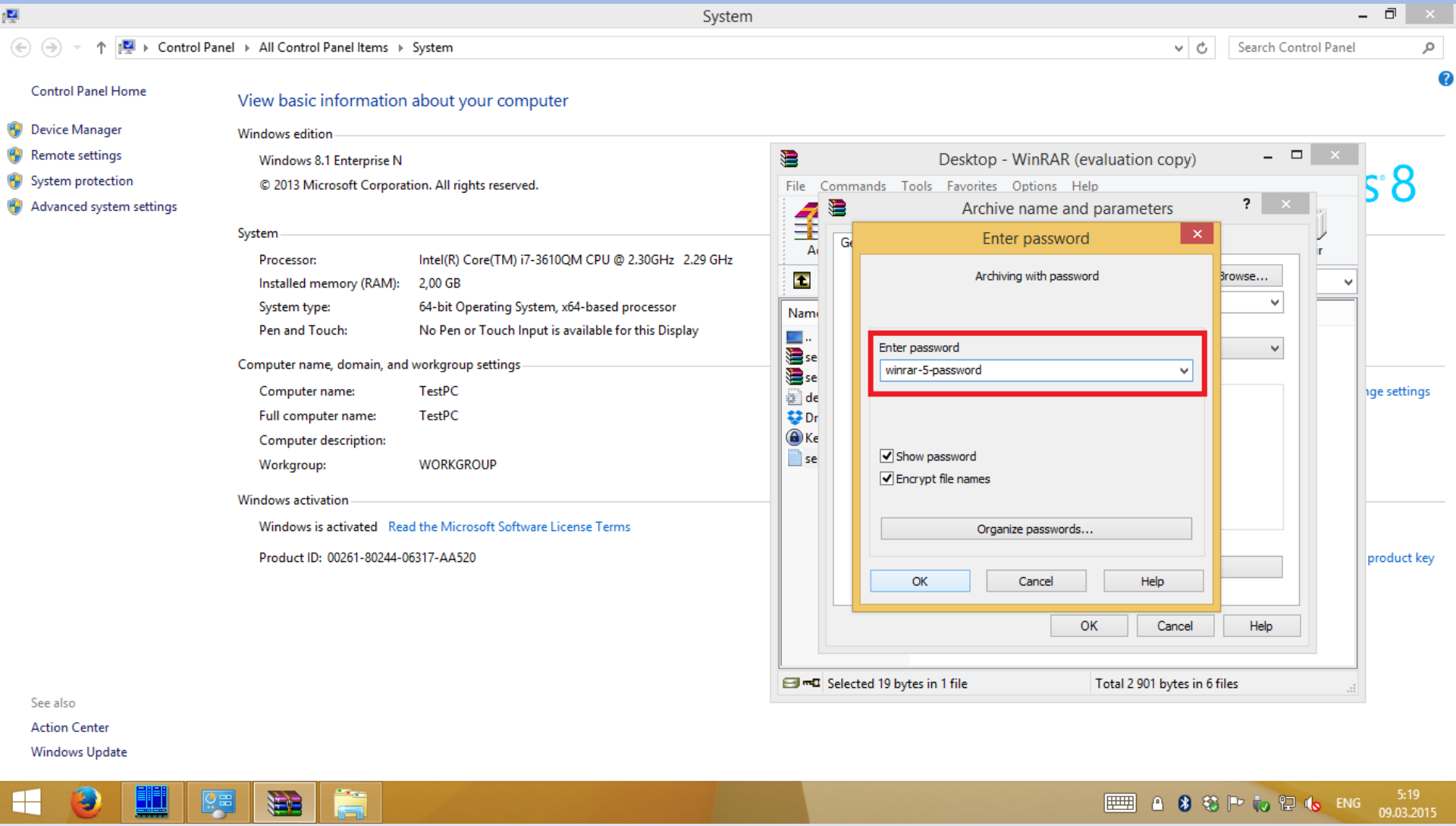
Зашифрованные CryptProtectMemory данные в ПО KeePass, WinRAR

	KeePass 1.28 (Classic)	KeePass 2.28 (Professional)	WinRAR 5.20
Идентификация зашифрованных данных	Сигнатура (на основе данных из файла БД – тип шифрования, число раундов)	Перед буфером с зашифрованными данными расположен QWORD или DWORD с размером буфера	Сигнатура, фиксированная длина (**0x100**)
Зашифрованные данные	Ключ шифрования БД	Пароль, хеш (SHA-256) от пароля	Пароль
Проверка расшифрованных данных	Проверка по файлу БД	Валидные данные в терминах UTF-8, соответствие одному из хешей SHA-256	Проверка по файлу архива

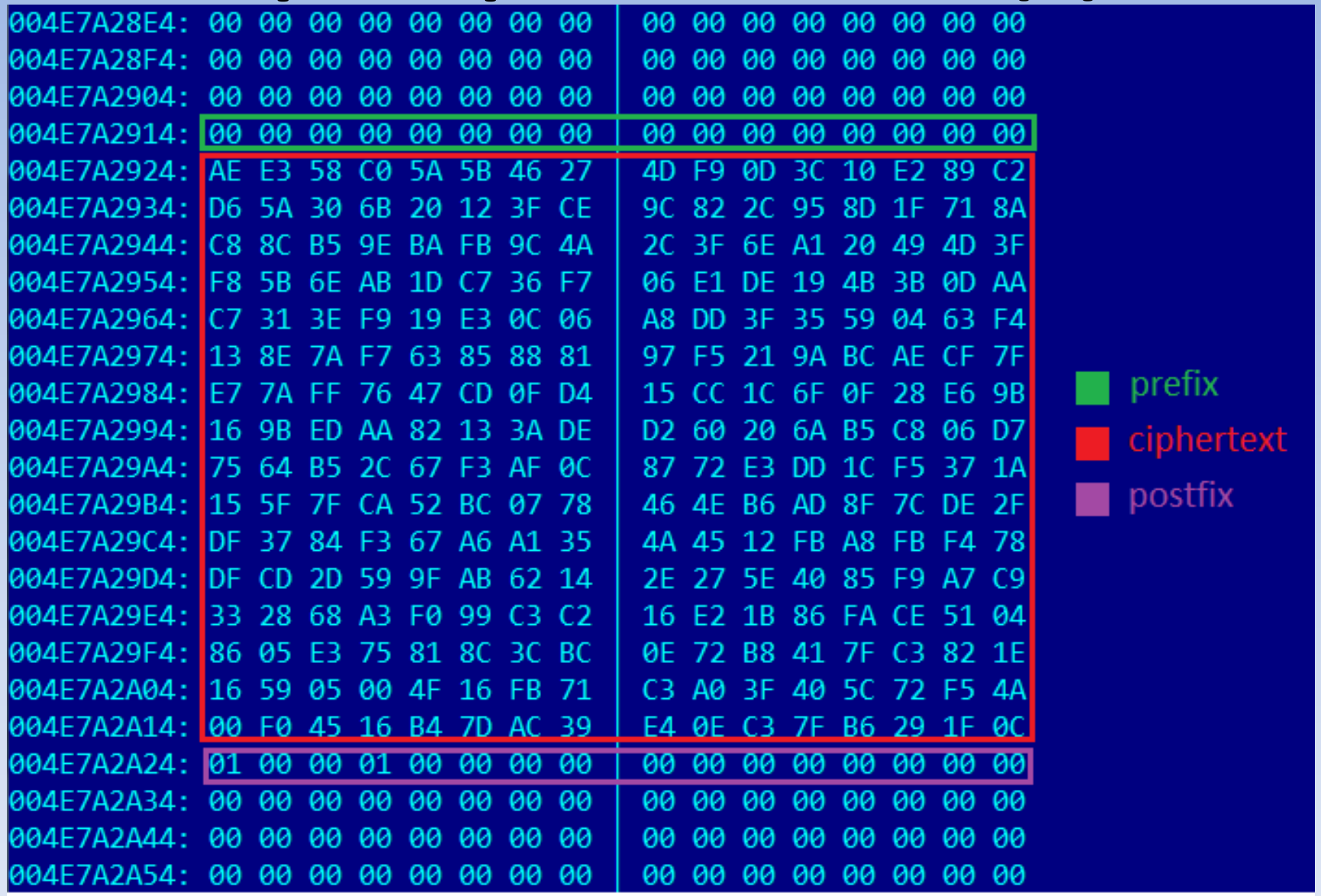
Пример – WinRAR 5 (1)



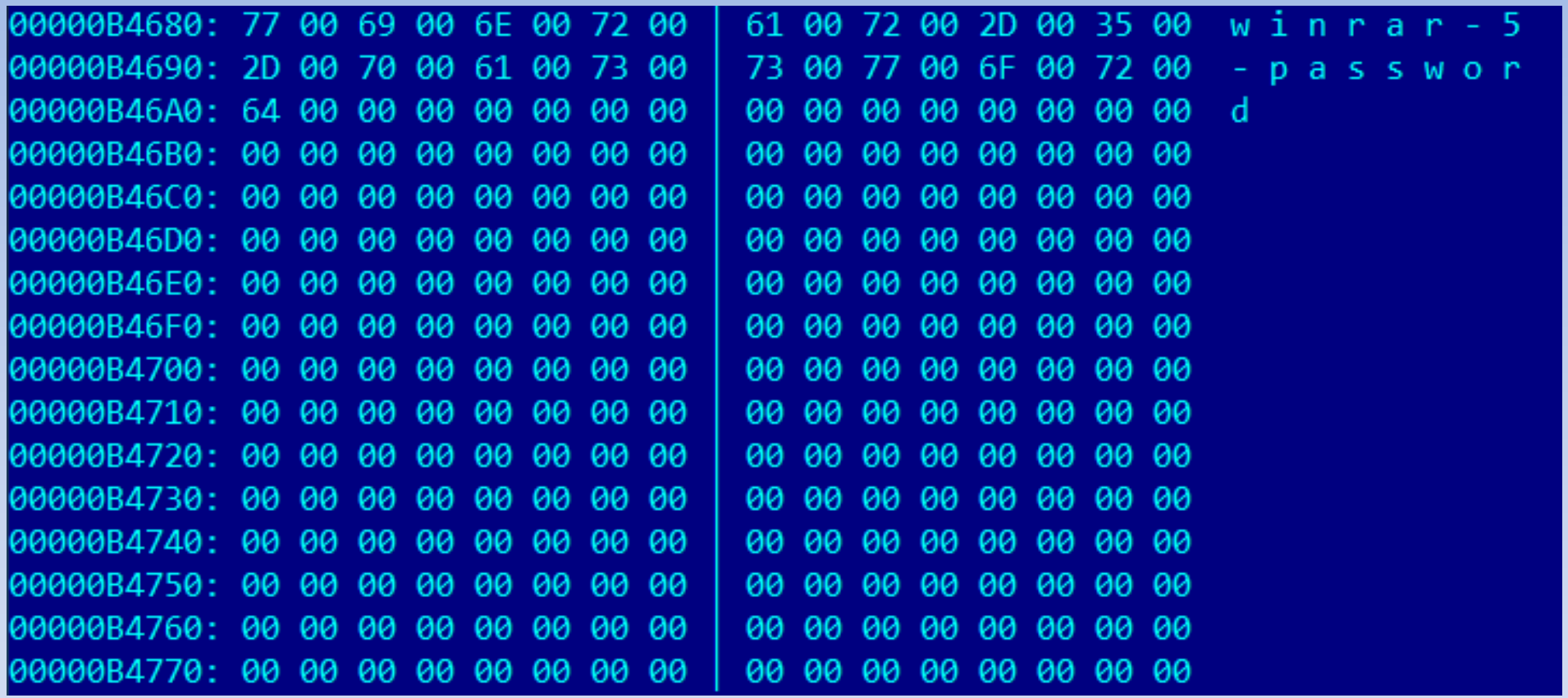
Пример – WinRAR 5 (2)



Пример – WinRAR 5 (3)



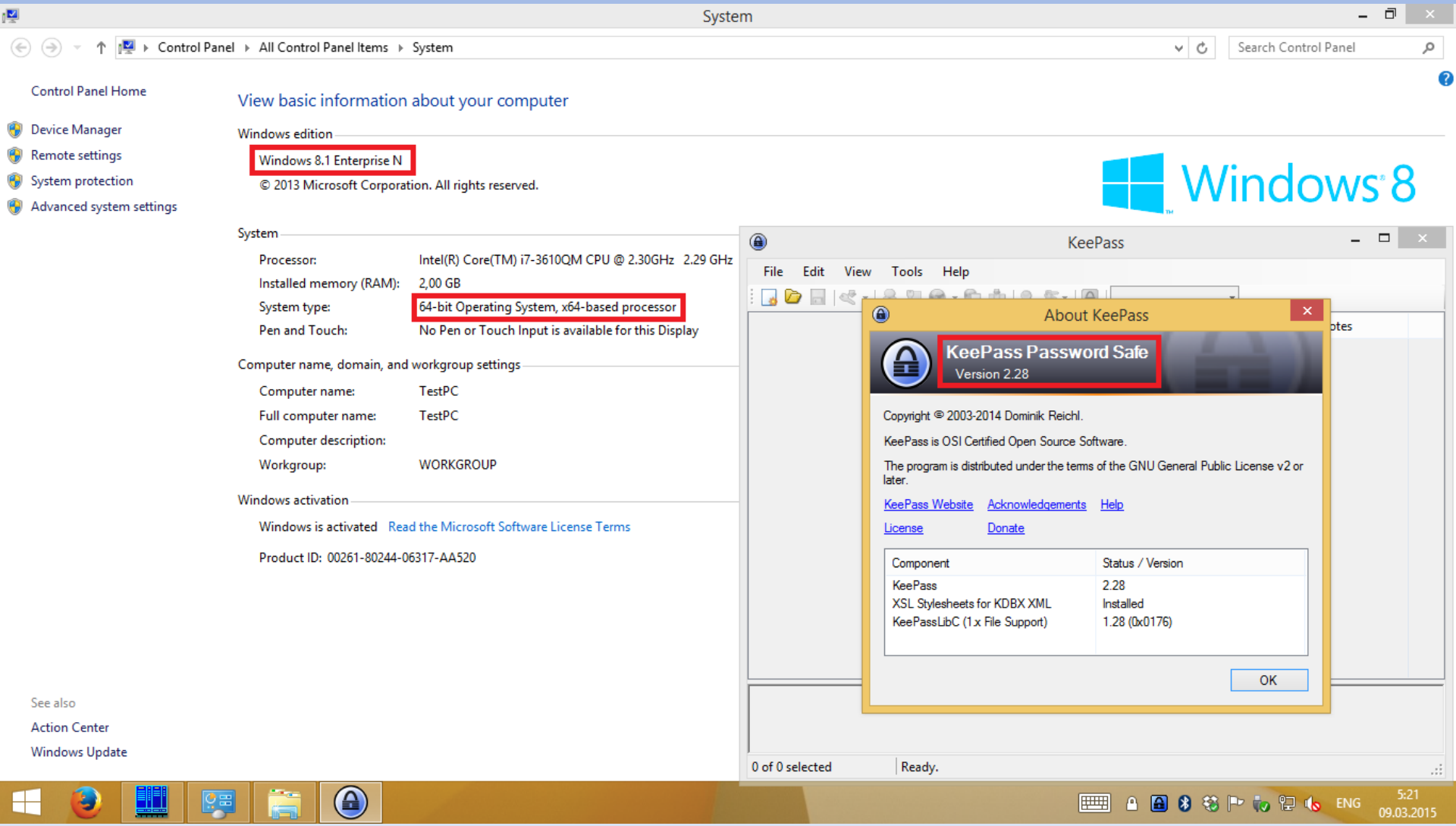
Пример – WinRAR 5 (4)



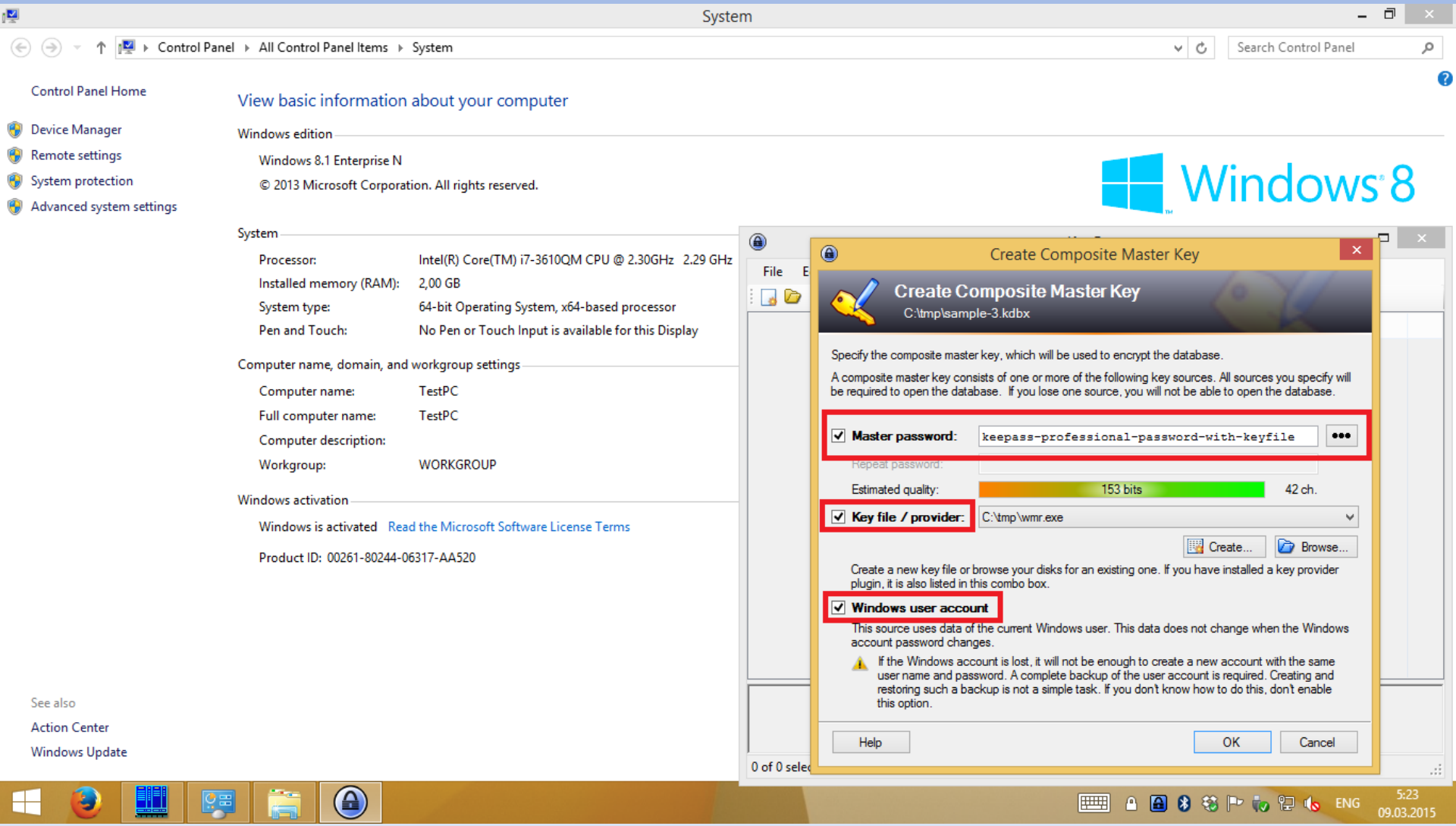
```
00000B4680: 77 00 69 00 6E 00 72 00 | 61 00 72 00 2D 00 35 00  w i n r a r - 5
00000B4690: 2D 00 70 00 61 00 73 00 | 73 00 77 00 6F 00 72 00  - p a s s w o r
00000B46A0: 64 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00  d
00000B46B0: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
00000B46C0: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
00000B46D0: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
00000B46E0: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
00000B46F0: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
00000B4700: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
00000B4710: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
00000B4720: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
00000B4730: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
00000B4740: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
00000B4750: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
00000B4760: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
00000B4770: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
```

В общей сложности проведено 10 успешных экспериментов с различными длинами пароля (до 128 символов включительно)

Пример – KeePass 2 (1)



Пример – KeePass 2 (2)



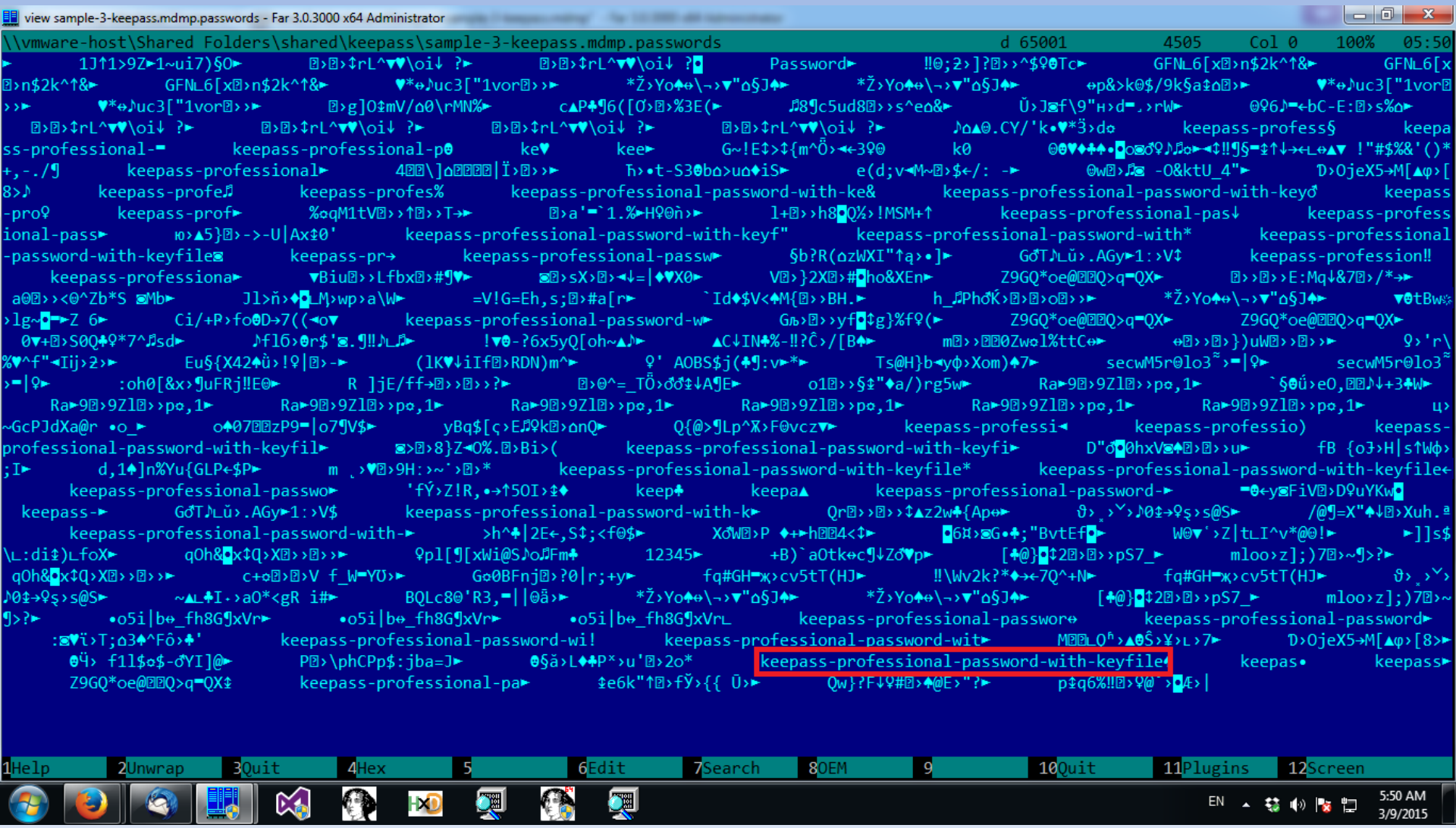
Пример – KeePass 2 (3)

```
002DF37470: 00 00 00 00 00 00 00 00  00 6D 93 32 FE 7F 00 00
002DF37480: 98 01 1A 02 00 00 00 00  00 00 00 00 00 00 00 00
002DF37490: 2A 00 00 00 01 00 00 00  00 00 00 00 00 00 00 00
002DF374A0: 28 7F 93 32 FE 7F 00 00  00 00 00 00 00 00 00 00
002DF374B0: F8 01 1A 02 00 00 00 00  E0 01 1A 02 00 00 00 00
002DF374C0: 2B 00 00 00 00 00 00 00  2A 00 00 00 01 00 00 00
002DF374D0: 01 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
002DF374E0: 00 00 00 00 00 00 00 00  B8 11 EF 90 FE 7F 00 00
002DF374F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
002DF37500: D0 66 EF 90 FE 7F 00 00  30 00 00 00 00 00 00 00
002DF37510: 15 C0 DF 4E 30 1A C6 77  F8 BA DA 68 BE BC FD E4
002DF37520: 8A C0 93 0C 13 0E E4 89  DC 2C 6F D7 09 DA 10 A5
002DF37530: A2 B4 B0 BA B9 76 47 68  95 9C 4B ED F3 1E 54 66
002DF37540: 00 00 00 00 00 00 00 00  D0 BB 6D 94 FE 7F 00 00
002DF37550: 04 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
002DF37560: D0 66 EF 90 FE 7F 00 00  2A 00 00 00 00 00 00 00
002DF37570: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
002DF37580: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
002DF37590: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
002DF375A0: 00 00 00 00 00 00 00 00  D0 BB 6D 94 FE 7F 00 00
002DF375B0: 08 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
002DF375C0: D0 BB 6D 94 FE 7F 00 00  04 00 00 00 00 00 00 00
002DF375D0: 00 00 00 00 00 00 00 00  D8 0B EF 90 FE 7F 00 00
```

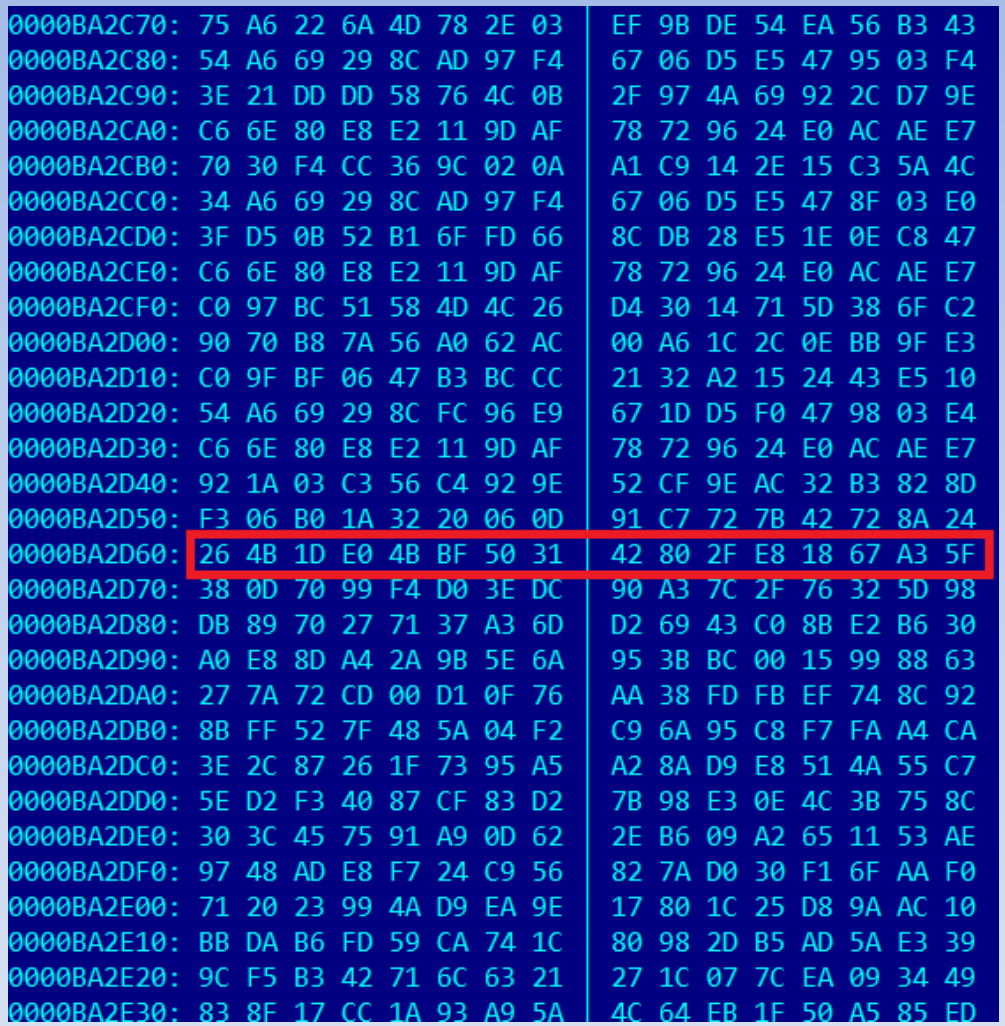
ciphertext size

ciphertext

Пример – KeePass 2 (4)



Пример – KeePass 2 (5)



SHA-256(‘keepass-professional-password-with-keyfile’) =

1063ce908562ae6eece8db593151cbe9**264b1de04bbf503142802fe81867a35f**

Пример – KeePass 2 (6)

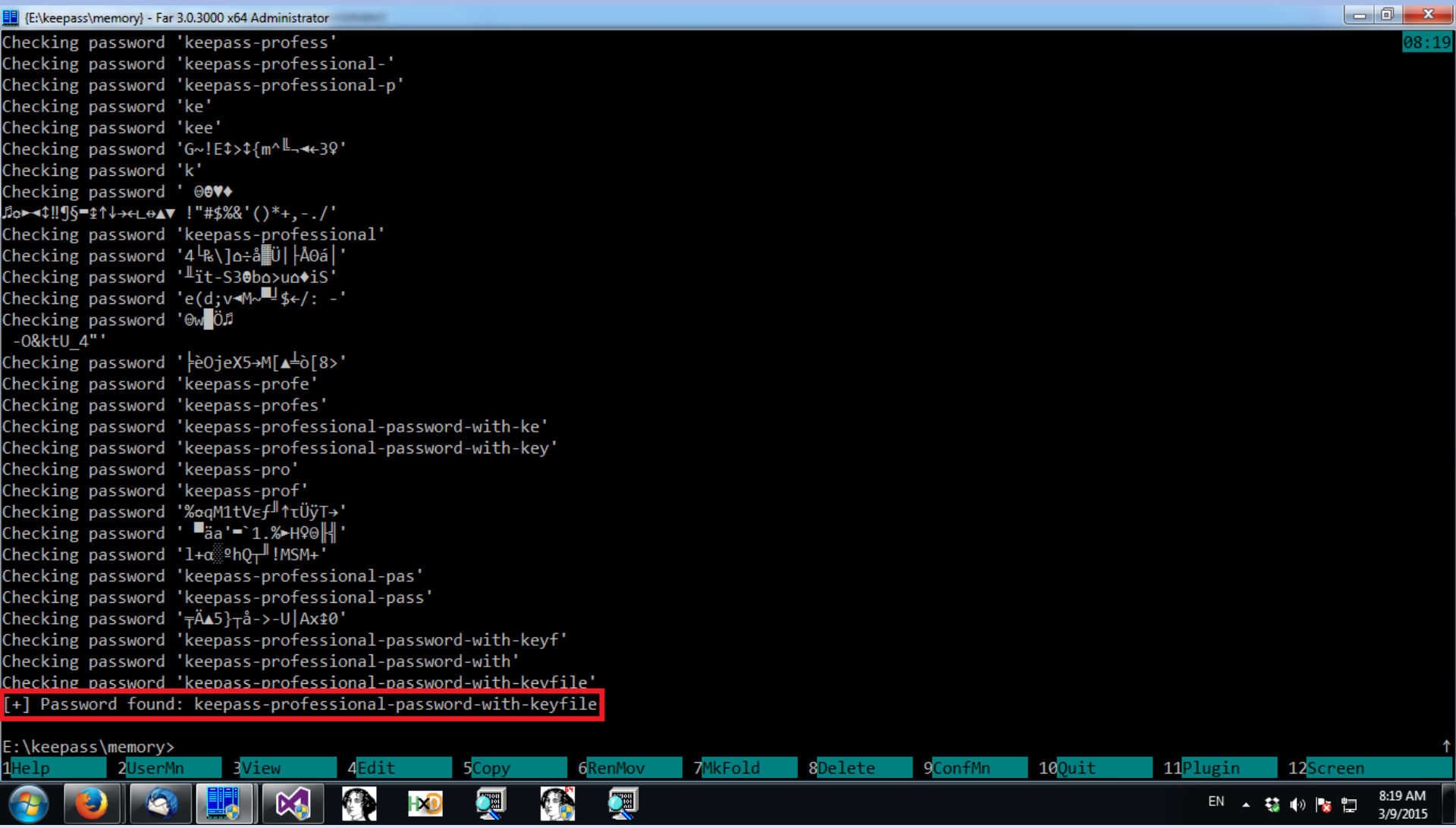
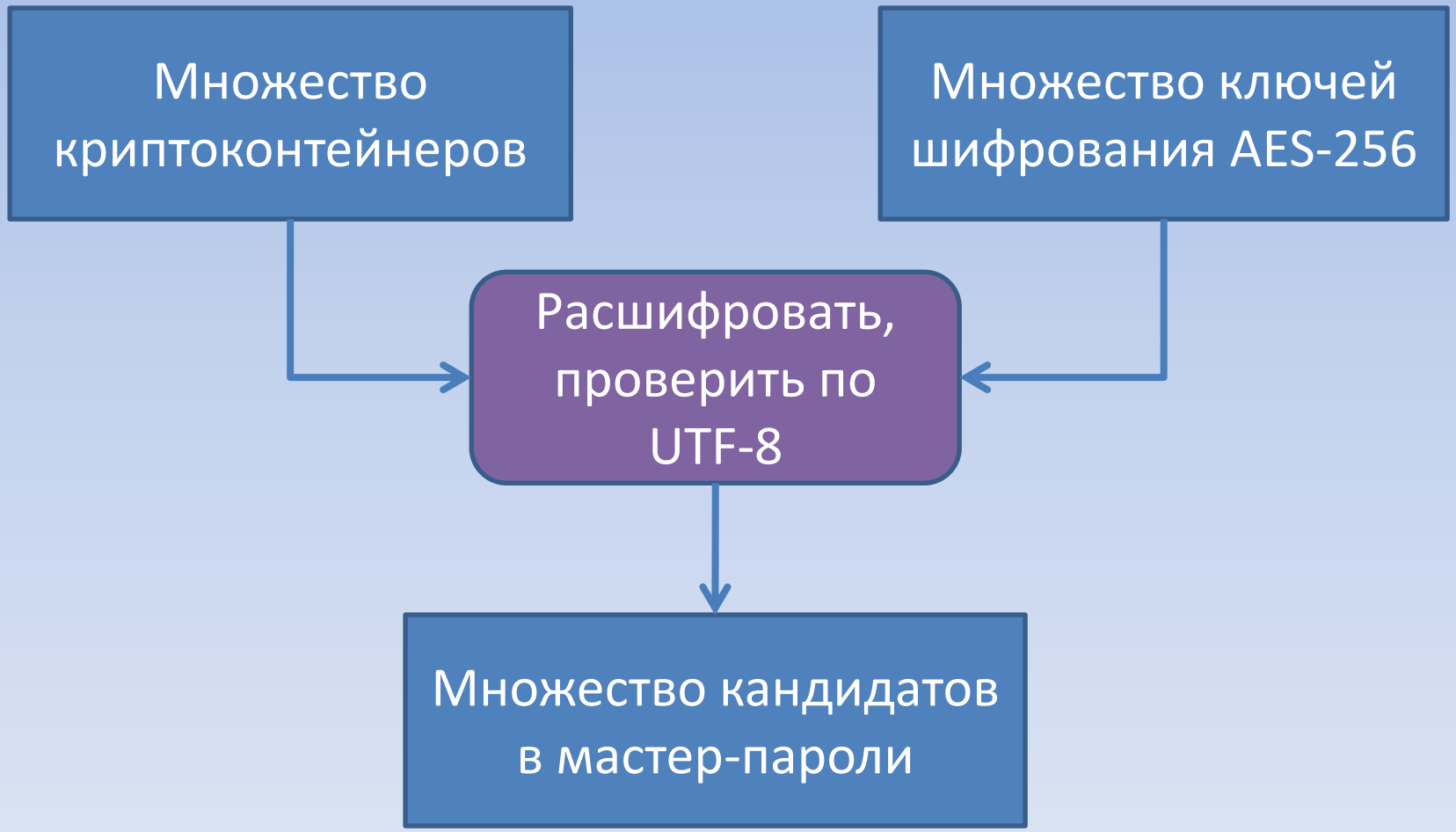


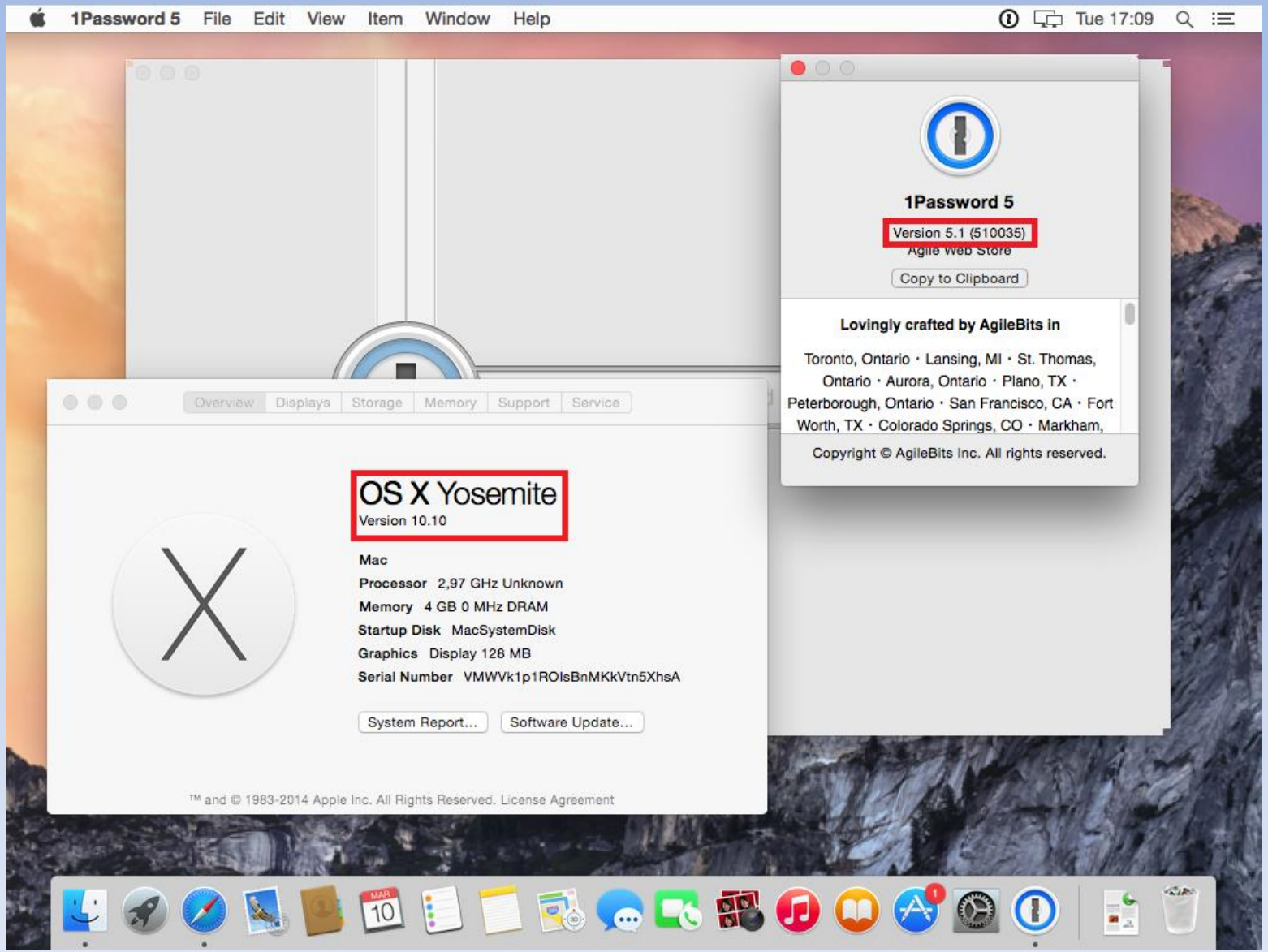
Схема защиты данных в памяти ПО 1Password (Mac OS X)

- Все критичные данные (в том числе, мастер-пароль) хранятся в криптоконтейнерах
- Данные зашифрованы с помощью **AES-256**, ключи шифрования в памяти лежат в открытом виде
- В контейнере данные хранятся в кодировке **UTF-8**

Построение множества кандидатов в мастер-пароли



Пример – 1Password (1)



Пример – 1Password (2)

```
[i] Step 1 - collect possible AES keys
[i] Key count: 7569
[i] Step 1 ran in 311 seconds
[i] Step 2 - collect possible opdata blocks
[i] Full containers count: 35
[i] Step 2 ran in 207 seconds
[i] Step 3 - generate dictionary
[i] AES key = 3a99377ac3548a97c5f046c706ea3c2bd7b1f21ad45f75b3a7dab344f4e47fe7
[i] password = {"title":"Login 2","ainfo":"login2","ps":0}
[i] AES key = 9e8c8fe28f5444f6c1137f4e4e54b53afac078493d549a4c99b3c4409e294f95
[i] password = {"fields":[{"value":"login2","name":"username","type":"T","designation":"username"}]}
[i] AES key = 3a99377ac3548a97c5f046c706ea3c2bd7b1f21ad45f75b3a7dab344f4e47fe7
[i] password = {"title":"Login 1","ainfo":"login","ps":1}
[i] decrypted data = 0000000000007b227469746c65223a224c6f67696e2031222c2261696e666f223a226c6f67696e22
...
[i] AES key = 35422038e4edcb5715ada7f97b1be045f3abc390c34f33f3758d4a806c859f3c
[i] password = 1password-mac-password-2
[i] decrypted data = 0000000000000000317061737376f72642d6d61632d70617373776f72642d32
[i] AES key = 08e137ed72f88bff35360e489774614a8975279c22873ac48c0a2fec81e3ae68
[i] password = {"fields":[{"value":"login","name":"username","type":"T","designation":"username"},{"v
[i] Step 3 ran in 2 seconds
Total passwords: 1924
```

В общей сложности проведено 10 успешных экспериментов с различными длинами мастер-пароля (до 128 символов включительно)

Результаты работы

Название ПО	Результат (на основе анализа образа памяти)
1Password 5.1 (Mac)	Восстановление мастер-пароля (независимо от блокировки БД)
KeePass 1 (Classic)	Восстановление ключа шифрования БД
KeePass 2 (Professional)	Восстановление мастер-пароля (в том числе, в случае использования ключевого файла и/или привязки к локальному пользователю)
WinRAR 5.20 (Windows)	Восстановление пароля от архива

Спасибо за внимание!