Бумажник Oracle Wallet: использование для шифрования данных на внешнем носителе

Владимир Пржиялковский
Преподаватель технологий Oracle
prz@yandex.ru
http://open.oracle.tu2.ru/.index

Ноябрь 2008 г.

Держи карман шире !

Крылатое выражение

Аннотация

Рассматривается использование электронного бумажника Oracle Wallet для защиты от прочтения данных на внешнем носителе путем прозрачного шифрования (TDE) содержимого столбцов таблиц и табличных пространств, а также путем шифрования резервных копий.

Введение

БД Oracle не является замкнутой системой. СУБД вступает в контакт с участниками компьютерной сети, а данные базы, равно как и резервные копии, технически хранятся на внешних носителях. Хотя СУБД Oracle имеет собственную систему защиты данных, внешнее окружение, с которым она взаимодействует, вовсе не подконтрольно ей. Например, канал связи прикладной программы с СУБД может испытать постороннее вмешательство; существует риск постороннего же обращения к содержимому файлов с данными в обход Oracle.

Проблема не связана исключительно с Oracle и носит общий характер для информационных систем как таковых. Стандартное решение состоит в использовании шифрования передаваемых данных и подсчете контрольной свертки (суммы). Более того, передача шифра (необходимого для расшифровки принимающей стороной) часто вдобавок вынуждено сопровождается ссылкой на «свидетельство подлинности», так называемый сертификат. Когда взаимодействие субъектов и объектов доступа в информационной системе идет активно, предъявлять в программе шифр или его свидетельство подлинности многократно и «вручную» и неудобно, и небезопасно. Гораздо легче запомнить эти сведения где-то один раз и «попросить» систему делать проверку по мере надобности самостоятельно.

Популярным приемом является использовать для такого локализованного расположения данных параметров защищенного доступа «электронный бумажник» (ewallet). По сути это файл на компьютере (сервере или клиенте), где информация о параметрах доступа сама в свою очередь защищена. Английским прообразом для «удостоверяющих параметров доступа» является слово credentials, заимствованное из юридического языка и обозначающее там «верительные грамоты». (Слово credentials должно быть знакомо администраторам работе с Oracle Enterprise Manager). В «информационном», электронном бумажнике могут храниться такие персональные «верительные грамоты», как ключи шифрования данных и сертификаты подлинности, а кроме того еще и «прочие секретные сведения» (цитата) об участнике доступа.

Фирма Oracle предполагает использовать в качестве электронного бумажника собственное решение, Oracle Wallet, существующее в рамках расширения Advanced Security для Oracle Enterprise Edition или для клиента. Oracle Wallet позволяет хранить и обеспечивать использование системой следующие основные категории сведений:

- «главный ключ» (masterkey) – шифр для автоматического шифрования данных на диске: в требуемых столбцах таблиц или целиком в табличных пространствах (transparent data encryption, TDE), а также для шифрования результатов сохранного (резервного) копирования;
- сертификаты подлинности ключей шифрования, используемых при передаче данных по защищенным каналам и сами ключи;
- имена пользователя и пароли ради упрощенного («беспарольного») соединения программы с сервером по данным из бумажника клиента.

В этой статье будет рассказано о применении электронного бумажника Oracle в первом качестве, а в последующей – во втором. Использование бумажника для упроцения процедуры соединения клиентом рассматриваться не будет.

В примерах ниже подразумевается файловая система ОС Windows, что никак не сказывается на существе дела.

Электронный бумажник Oracle

Бумажник Oracle Wallet представляет собой двоичный файл, составленный по промышленному стандарту PKCS #12 «синтаксиса обмена *персональной информацией*» (http://www.rsa.com/rsalabs/node.asp?id=2138), что обеспечивает ему совместимость с ПО третьих фирм. Файл имеет имя *ewallet.p12*, и он должен располагаться в любом доступном СУБД каталоге файловой системы. Тем не менее, когда бумажник используется для хранения главного ключа шифрования (о чем речь в этой статье), имеется умолчательное местонахождение (не требующее явного указания): по системе обозначений Windows это *%ORACLE_BASE%\admin\%ORACLE_SID%\wallet*. (Оно разумно привязано к месту хранения рабочих файлов СУБД, так как несколько СУБД на одном компьютере не имеют права пользоваться общим бумажником).

Сам бумажник Oracle появился в версии 8.1, однако приводимое далее его применение оказалось возможным только в версии 10.2.

Для работы со своим электронным бумажником фирма Oracle дает следующие средства:

- команды SQL;
- программу Oracle Wallet Manager (она же *owm*);
- программу *orapki*;
- программу *mkstore*.

Три последние суть обращения к методам `main` классов Java, соответственно:

- `oracle.security.admin.wltmgr.owma.OwmaApp`,
- `oracle.security.pki.textui.OraclePKITextUI`,
- `oracle.security.pki.OracleSecretStoreTextUI`.

(Все они, вместе с адресованными по цепочке, располагаются в каталогах ПО Oracle в разных архивах ORACLE_HOME).

Два последние класса позволяют работать с электронным бумажником Oracle посредством командной строки, а первый – посредством графики; он и назначен быть основным инструментом для пользователя. Описания API, позволяющего пользователю работать с бумажником из своей программы, фирма Oracle не дает.

Создание, открытие и закрытие бумажника с главным ключом

Создать бумажник с главным ключом шифрования можно неявно командой `ALTER SYSTEM SET ENCRYPTION WALLET ...` или явно программами Oracle Wallet Manager и *mkstore*. Обратите внимание, что главный ключ (симметричный) будет применяться СУБД для шифрования данных при размещении на диске и расшифровке при взятии с диска, поэтому дальнейшая работа предполагается на сервере, с серверным ПО.

К сожалению, для электронного бумажника в Oracle ПО написано недостаточно тщательно, так что несоблюдение описанного далее порядка действий способно привести вплоть до ошибки ORA-00600 и необходимости перезагрузки СУБД.

Проще завести бумажник неявно, командой SQL. Для этого *нужно создать* необходимый умолчательный каталог и выдать команду SQL от имени SYS, например в SQL*Plus:

```
SQL> HOST mkdir C:\oracle\admin\orcl\wallet
SQL> ALTER SYSTEM SET ENCRYPTION KEY IDENTIFIED BY "amicus123";
```

Приведенный пароль будет служить для последующего *открытия* бумажника и вообще доступа к его содержимому. Двойные кавычки неслучайны, так как в пароле различаются большие и малые буквы. Кроме того пароль должен содержать более 8 символов и складываться одновременно из букв и цифр. Для смены пароля в будущем надо будет использовать программу Oracle Wallet Manager, обратиться к которой в Windows можно через меню запуска программ или из командной строки ОС, а в Unix – набором *owm* в командной строке ОС.

По указанной команде SQL в (умолчательном) каталоге появится файл бумажника с главным ключом доступа. К фактическому использованию главного ключа из бумажника мы еще не готовы, так как бумажник нужно открыть. Команды открытия и закрытия бумажника:

```
ALTER SYSTEM SET ENCRYPTION WALLET OPEN AUTHENTICATED BY "amicus123";
ALTER SYSTEM SET ENCRYPTION WALLET CLOSE;
```

Выполним первую команду, и созданый предварительно бумажник можно использовать.

Защита данных на внешнем носителе средствами TDE

Термин «прозрачное шифрование данных» (TDE) используется для обозначения процессов автоматической шифровки данных при размещении на носитель и дешифровки при чтении с носителя. «Прозрачное» здесь означает «прозрачное» для прикладной программы, в которой ни коим образом нет нужды учитывать фактическое выполнение указанных действий. В противовес этому программа имеет право самостоятельно шифровать нужные значения прежде чем передавать их БД и самостоятельно расшифровывать по извлечению, пользуясь, например, пакетом DBMS_CRYPTO. Оба подхода имеют свои достоинства; при этом TDE проще для программиста, но и требует бумажника.

Техника TDE не присуща исключительно СУБД Oracle (пример: SQL Server), или даже базам данных вообще (пример: патент IBM *7426745*).

Создание столбцов с зашифрованными данными

Создадим таблицу для работы, для определенности в схеме SCOTT:

```
CREATE TABLE closed (
  a VARCHAR2 ( 10 )
, b VARCHAR2 ( 10 )
, c VARCHAR2 ( 10 )
);
INSERT INTO closed VALUES ( 'normal1', 'secret', 'normal1' );
INSERT INTO closed VALUES ( 'normal2', 'secret', 'normal2' );
COMMIT;
```

Для начала рассмотрим обычное хранение строк. Выдадим в SQL*Plus:

```
SQL> SELECT
  2    DBMS_ROWID.ROWID_RELATIVE_FNO ( ROWID ) file#
  3  , DBMS_ROWID.ROWID_BLOCK_NUMBER ( ROWID ) block#
  4  FROM closed
SQL> ;

     FILE#     BLOCK#
---------- ----------
         4        597
         4        597
```

Обе короткие строки попали в один блок на диске. В моем случае это блок № 597 в файле № 4. Подставим полученные значения в команду, которую выдадим от имени SYS:

```
ALTER SYSTEM DUMP DATAFILE 4 BLOCK 597;
```

В трассировочном файле серверного процесса, обслуживающего сеанс, появилась примерно следующая информация:

```
...
EEF7DB0 32656662 5F353633 72756F53 6F546563  [bfe2365_SourceTo]
EEF7DC0 4C4D5448 766E6F43 0703012C 6D726F6E  [HTMLConv,...norm]
EEF7DD0 06326C61 72636573 6E077465 616D726F  [al2.secret.norma]
EEF7DE0 012C326C 6F6E0703 6C616D72 65730631  [l2,...normal1.se]
EEF7DF0 74657263 726F6E07 316C616D A1810604  [cret.normal1....]
Block header dump:  0x01000255
...
```

Пока еще не засекреченные значения выделены серым фоном. Зашифруем их в блоке:

```
ALTER TABLE closed MODIFY ( b ENCRYPT );
```

Это можно сделать только при открытом бумажнике. Вот что получим в файле трассировки серверного процесса:

```
...
EEF7D30 0B6A7807 12380B13 0703022C 6D726F6E  [.xj...8.,...norm]
EEF7D40 34326C61 0E8464A4 1304CAAE E1116635  [al24.d......5f..]
EEF7D50 5B06C8CD C3AABA4A 8B4EA144 CE2B4987  [...[J...D.N..I+.]
EEF7D60 EB2DEBB1 E97EB0C5 CDCA2CDB 99AC18E9  [..-...~..,......]
EEF7D70 52E1F415 CA85201C 726F6E07 326C616D  [...R. ..normal2]
EEF7D80 0703022C 6D726F6E 34316C61 1F21C866  [,...normal14f.!.]
EEF7D90 D765D462 EC2D8A33 8DF8F328 EC42E142  [b.e.3.-.(...B.B.]
EEF7DA0 CB3EEB72 A36909CE A28C845E 0F04567C  [r.>...i.^...|V..]
EEF7DB0 3C74BEC5 8F3EAEA8 B3612F6E 6009C40C  [..t<..>.n/a....`]
EEF7DC0 726F6E07 316C616D 0703002C 6D726F6E  [.normal1,...norm]
EEF7DD0 06326C61 72636573 6E077465 616D726F  [al2.secret.norma]
EEF7DE0 002C326C 6F6E0703 6C616D72 65730631  [l2,...normal1.se]
EEF7DF0 74657263 726F6E07 316C616D A9BB0601  [cret.normal1....]
Block header dump:  0x01000255
...
```

Значения двух слов 'secret' в блоке действительно оказались зашифрованы (и стали занимать больше места). Обратите внимание, что шифрование ранее занесенных данных Oracle отрабатывает алгоритмом команды UPDATE, который *допускает* временное сохранение в блоке старых данных, на радость злоумышленникам (в SQL Server то же самое, если верит интернету).

Выдача значений не раскроет того обстоятельства, что в блоке они хранятся зашифровано:

```
SQL> SELECT * FROM closed;

A          B          C
---------- ---------- ----------
normal1    secret     normal1
normal2    secret     normal2
```

Упражнение. Закройте бумажник командой, приведенной выше, и убедитесь, что в этом случае Oracle не сможет показать значение зашифрованного поля B таблицы CLOSED. Откройте бумажник снова для продолжения работы.

Бросается в глаза, что одно и то же значение ('secret') дало разный результат шифрования. Так произошло потому, что для дополнительной защиты Oracle перед шифрованием приписывает к значению случайную последовательность байтов, так называемую «привязку» (salt). Это препятствует расшифровке, и даже лишает возможности злоумышленника понять, что в разных полях исходно были одинаковые значения. С

другой стороны (а) «привязка» замедляет обработку данных и (б) препятствует созданию (древовидного) индекса на шифруемый столбец. Последнее вызвано тем, что значения полей индексируемого столбца помещаются в структуру индекса, и поскольку равные исходные значения будут давать разные шифрованные, индекс не сможет отрабатывать поиск по диапазону, то есть обесценится. Вот Oracle это и запрещает:

```
SQL> CREATE INDEX closed_b ON closed ( b );
CREATE INDEX closed_b ON closed ( b )
                                  *
ERROR at line 1:
ORA-28338: cannot encrypt indexed column(s) with salt
```

От применения привязки можно отказаться:

```
ALTER TABLE closed MODIFY ( b ENCRYPT NO SALT );
```

Вот пример последовавшего результата из трассировочного файла:

```
...
EEF7CC0 02C10265 06C102FF 0703012C 6D726F6E  [e.......,...norm]
EEF7CD0 24326C61 C565F542 45F6CA9D C4516D27  [al2$B.e....E'mQ.]
EEF7CE0 902DEF69 C655685B 00844987 08803082  [i.-.[hU..I...0..]
EEF7CF0 A6106E22 45F862E5 726F6E07 326C616D  ["n...b.E.normal2]
EEF7D00 0703012C 6D726F6E 24316C61 C565F542  [,...normal1$B.e.]
EEF7D10 45F6CA9D C4516D27 902DEF69 C655685B  [...E'mQ.i.-.[hU.]
EEF7D20 00844987 08803082 A6106E22 45F862E5  [.I...0.."n...b.E]
EEF7D30 726F6E07 316C616D 0703002C 6D726F6E  [.normal1,...norm]
EEF7D40 34326C61 0E8464A4 1304CAAE E1116635  [al24.d......5f..]
EEF7D50 5B06C8CD C3AABA4A 8B4EA144 CE2B4987  [...[J...D.N..I+.]
EEF7D60 EB2DEBB1 E97EB0C5 CDCA2CDB 99AC18E9  [..-...~..,......]
EEF7D70 52E1F415 CA85201C 726F6E07 326C616D  [...R. ..normal2]
EEF7D80 0703022C 6D726F6E 34316C61 1F21C866  [,...normal14f.!.]
EEF7D90 D765D462 EC2D8A33 8DF8F328 EC42E142  [b.e.3.-.(...B.B.]
EEF7DA0 CB3EEB72 A36909CE A28C845E 0F04567C  [r.>...i.^...|V..]
EEF7DB0 3C74BEC5 8F3EAEA8 B3612F6E 6009C40C  [..t<..>.n/a....`]
EEF7DC0 726F6E07 316C616D 0703002C 6D726F6E  [.normal1,...norm]
EEF7DD0 06326C61 72636573 6E077465 616D726F  [al2.secret.norma]
EEF7DE0 002C326C 6F6E0703 6C616D72 65730631  [l2,...normal1.se]
EEF7DF0 74657263 726F6E07 316C616D A9BB0601  [cret.normal1....]
Block header dump:  0x01000255
...
```

Теперь два верхних отмеченных серым фоном значения в блоке стали одинаковыми.

Упражнение. Проверить открывшийся шанс завести индекс по полю B таблицы CLOSED.

Справочная информация и некоторые подробности

Посмотреть состояние и местонахождение бумажника с шифроключами можно только с версии 11 в таблице V$ENCRYPTION_WALLET.

Перечень шифруемых столбцов в таблицах БД можно получить из отдельных системных таблиц DBA/ALL/USER_ENCRYPTED_COLUMNS. В нашем случае мы получим:

```
SQL> SELECT * FROM user_encrypted_columns;

TABLE_NAME           COLUMN_NAME          ENCRYPTION_ALG                 SAL
-------------------- -------------------- ------------------------------ ---
CLOSED               B                    AES 192 bits key               NO
```

Полное имя последнего столбца SALT. Поле ENCRYPTION_ALG сообщает алгоритм шифрования. Кроме умолчательного AES192 возможны AES128, AES256 и 3DES168. Эти алгоритмы следует указывать явно:

```
ALTER TABLE closed REKEY USING 'aes256';
```

Обратите внимание, что алгоритм указывается не для столбца, а для всей таблицы. Это потому, что Oracle не позволяет разным столбцам таблицы использовать разные алгоритмы шифрования: технически «шифрование выполняется на уровне блока» (цитата). Фактически последняя команда сменяет не только алгоритм, но и ключ шифрования, тоже один на всю таблицу. Этот собственный ключ шифрования таблицы (не «главный») тут же сам шифруется главным ключом из бумажника и запоминается для нее в БД, в системной таблице ENC$. Оттуда он будет браться при всякой шифровке/расшифровке значений полей строк таблицы. Так, при обращении к зашифрованному полю таблицы СУБД возьмет из ENC$ зашифрованый ключ этой таблицы, расшифрует его главным ключом из бумажника, и уже восстановленым ключом таблицы расшифрует значение поля.

Если нужно сменить хранимые шифрозначения, можно выдать просто:

```
ALTER TABLE closed REKEY;
```

На «законную» возможность читать данные это никак не скажется (равно как и предыдущее действие), но хранимые значения будут подменены на вычисленные заново.

Упражнение. Проверить, изменится ли хранение в БД зашифрованных данных при смене шифроключа таблицы.

Посмотреть ключи шифрования в бумажнике можно программой *mkstore*, например:

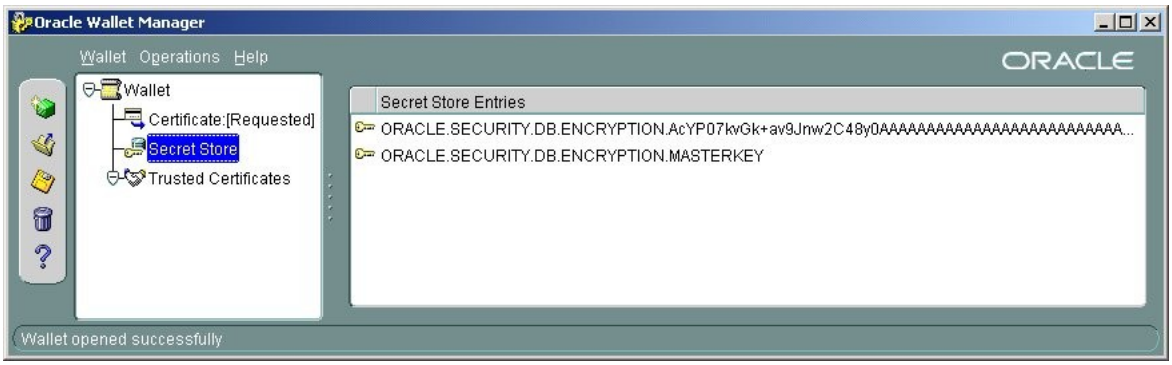
```
>mkstore -wrl C:\oracle\admin\orcl\wallet -list
Enter password: amicus123

Oracle Secret Store entries:
ORACLE.SECURITY.DB.ENCRYPTION.AcYP07kvGk
+av9Jnw2C48y0AAAAAAAAAAAAAAAAAAAAAAAAAAAA
ORACLE.SECURITY.DB.ENCRYPTION.MASTERKEY
```

То же, но чуть иначе, покажет программа *orapki*, если выдать:

```
>orapki wallet display -wallet C:\oracle\admin\orcl\wallet -pwd amicus123
```

Эти же ключи покажет и Oracle Wallet Manager:



(Для этого нужно будет вызвать программу, сослаться на каталог с файлом бумажника и ввести пароль).

А вот значение главного ключа покажет только программа *mkstore* (следует одна строка):

```
>mkstore -wrl C:\oracle\admin\orcl\wallet -viewEntry
ORACLE.SECURITY.DB.ENCRYPTION.MASTERKEY
```

Сменить значение главного ключа можно командой типа:

```
ALTER SYSTEM SET ENCRYPTION KEY IDENTIFIED BY "amicus123";
```

Упражнение. Проверить, изменится ли хранение в БД зашифрованных данных при смене главного ключа в бумажнике.

Создание табличных пространств с зашифрованными данными

Когда возникает надобность шифровать данные многих столбцов в многих таблицах, приходится иметь дело с большим количеством справочных сведений, что технологически хлопотно. Версия 11 дает для таких случаев обобщенное решение: шифрование всех данных, помещаемых в конкретное табличное пространство. Такое свойство табличного пространства неизменно и указывается один раз при его создании, например:

```
CREATE TABLESPACE encrypted_data
DATAFILE 'c:\oracle\oradata\orcl\encrypt_df.dat' SIZE 1M
ENCRYPTION USING '3DES168'
DEFAULT STORAGE ( ENCRYPT )
;
```

Оно отображено в поле ENCRYPTED таблицы DBA_TABLESPACES.

Создание шифрованых запасных копий

Данные БД могут находиться на внешнем носителе и в виде запасных (резервных) копий. Версия 10 позволила их шифровать. Это делается программой *rman*, и распространяется на резервные наборы (но не на копии образов файлов).

Шифрование резервного набора программой *rman* может осуществляться (1) с явным указанием ключа шифрования; (2) «прозрачно», то есть без явного указания, когда ключ берется из бумажника; (3) в «двойном режиме», когда допускается указывать ключ явно или пользоваться бумажником – по выбору. Администратору проще использовать для шифрования резервных наборов бумажник, и этот способ рассмотрен ниже.

Выдадим:

```
>rman TARGET /
... [ответ] ...
RMAN> SET ENCRYPTION OFF;
... [ответ] ...
RMAN> BACKUP AS BACKUPSET TABLESPACE users TAG "unencrypted";
... [ответ] ...
RMAN> SET ENCRYPTION ON;
... [ответ] ...
RMAN> BACKUP AS BACKUPSET TABLESPACE users TAG "the last one";
... [ответ] ...
```

Для проверки можно последовательно выдать команды (далее в этом разделе – все для *rman*):

```
RESTORE TABLESPACE users VALIDATE;
SQL "ALTER SYSTEM SET ENCRYPTION WALLET CLOSE";
RESTORE TABLESPACE users VALIDATE;
RESTORE TABLESPACE users FROM TAG "unencrypted" VALIDATE;
SQL 'ALTER SYSTEM SET ENCRYPTION WALLET open AUTHENTICATED BY "amicus123"';
RESTORE TABLESPACE users VALIDATE;
```

Алгоритмы, которые можно использовать для шифрования, перечисленны в таблице V $RMAN_ENCRYPTION_ALGORITHMS. Задать нужный алгоритм можно явочным путем, например:

```
SET ENCRYPTION ALGORITHM 'AES192';
```

На практике в *rman* удобно установить умолчания, например:

```
CONFIGURE ENCRYPTION ALGORITHM 'AES256';
CONFIGURE ENCRYPTION FOR DATABASE ON;
```

Другие свойства бумажника

Выбор расположения файла бумажника

Файл бумажника *ewallet.p12* имеет право располагаться в любом месте файловой системы сервера, лишь бы только в области доступности СУБД. Нестандартное его местонахождение следует указать в конфигурационном файле серверного сетевого ПО *sqlnet.ora*. Включим в него параметр:

```
...
WALLET_LOCATION =
   ( SOURCE =
     ( METHOD = FILE )
     ( METHOD_DATA =
       ( DIRECTORY = C:\oracle\product\10.2.0\db_1\encryptionwallet )
     )
   )
...
```

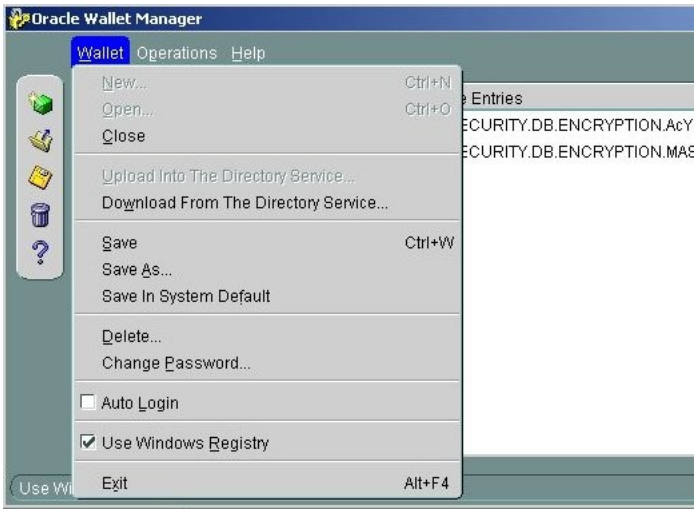
Создадим указанный каталог и перенесем в него ранее созданый файл *ewallet.p12*. Закроем бумажник и откроем снова командами `ALTER SYSTEM`, как выше. Файл бумажника самодостаточен, и поэтому сделанного довольно, чтобы продолжать с ним работу на новом месте.

Вместо названия WALLET_LOCATION файл *sqlnet.ora* понимает название ENCRYPTION_WALLET_LOCATION.

Размещение бумажника в реестре Windows

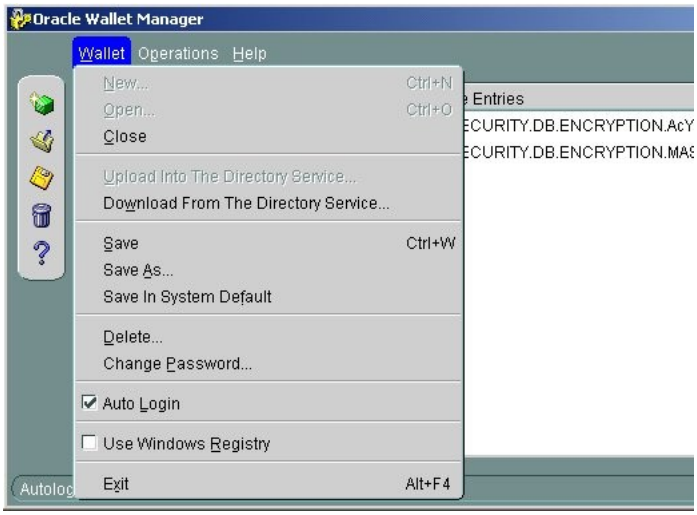
В Windows есть возможность поместить бумажник не в файл, а в реестр. Местонахождением в этом случае будет раздел *\\HKEY_CURRENT_USER\SOFTWARE\ORACLE\WALLETS*, волей-неволей скрывающий бумажник в профиле пользователя; то есть сохранение его в реестре выгоднее и по части управляемости, и по части защищенности.

Перенести бумажник в реестр можно средствами Oracle Wallet Manager (пометка *Use Windows Registry* в меню *Wallet*):



Постоянно открытый бумажник

Бумажник не обязательно открывать и закрывать всякий раз по мере надобности вручную. На компьютере, где он располагается, можно сообщить ему режим «автооткрытия» – при запуске ОС. Сделать это можно опять-таки через Oracle Wallet Manager:



Обратите внимание, что после этого в каталоге бумажника на пару файлу *ewallet.p12* появился файл *cwallet.sso*. Расширение *sso* есть сокращение от single sign-on, обозначающего «технику единого входа». Новый файл есть по сути копия прежнего бумажника (но не копия файла).

Упражнение. Проверить автоматическую готовность бумажника к употреблению после перевода в режим автооткрытия.

Чтобы снять режим автооткрытия, в том же меню *Wallet* следует снять пометку *Auto Logon*. Файл *cwallet.sso* после этого автоматически пропадет. Это же (но не обратное !) действие можно выполнить командой, например:

```
>mkstore -wrl C:\oracle\admin\orcl\wallet -deleteSSO
```

Внешнее хранение ключа

Бумажник Oracle позволяет работать с ключами, хранимыми не в файле, а на внешнем устройстве, например подключаемом через порт USB. В этом случае (де-)шифрующие процедуры отрабатываются также на внешнем устройстве. Связь с таким устройством позволяет организовать программа *orapki*, однако она носит вполне конкретный характер и здесь не поясняется.