

ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ ШИФРОВАНИЯ НА БАЗЕ FIREBIRD ENCRYPTION PLUGIN FRAMEWORK С КЛИЕНТОМ НА DELPHI

**Все о шифровании:
встреча с пользователями Firebird**

Сергей Никитин, IBSurgeon/IBase

www.ibase.ru

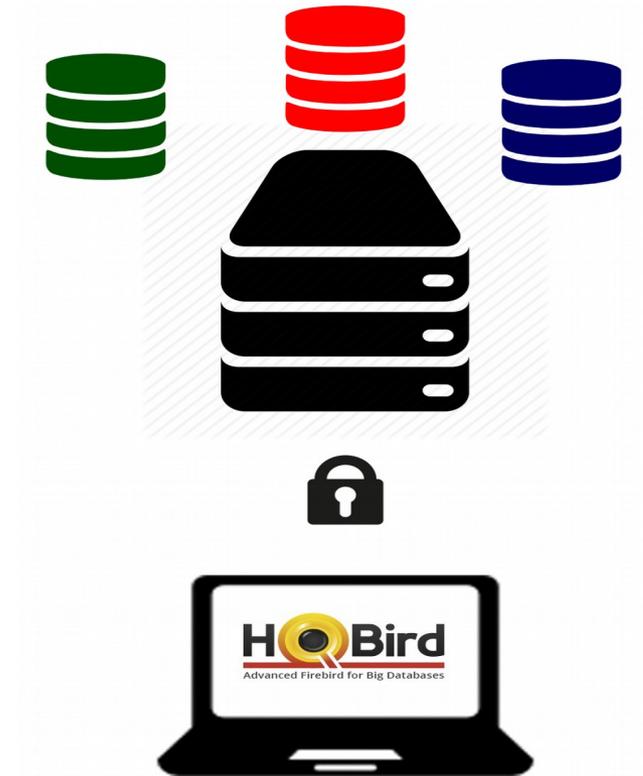
www.ib-aid.com

Регулярные встречи комьюнити Firebird

- 2 марта 2017 года - «Все о шифровании».
[Скачайте материалы](#)
- 9 декабря 2016 года - «Предварительный обзор Firebird 4 и Отказоустойчивые решения на Firebird».
[Скачать материалы](#)
- Не хотите пропустить следующую встречу с комьюнити? Подпишитесь на нашу рассылку (на любой странице www.ibase.ru, в окошке-слайдере справа).

IBSurgeon/iBase

IBSurgeon



План – о чем будем говорить

- 1) Как зашифровать БД
- 2) Обычное соединение к БД (без шифрования)
- 3) Соединение с шифрованием
- 4) Реализация клиента на Дельфи
- 5) Обработка ошибок

Как зашифровать БД

Шаг 1 – подготовка к шифрованию БД

Скачайте файлы плагина (или скомпилируйте)!

Выберите базу данных, которую шифруем (например, `examples/empbuild/employee.fdb`).

Поместите базу в какой-нибудь каталог, например `c:\db\employee.fdb`

Создайте алиас в `databases.conf` с указанием плагина

```
crypt = c:\db\employee.fdb
```

```
{  
  KeyHolderPlugin = KeyHolder  
}
```

Поместите следующие файлы в папку сервера plugins

- DbCrypt.dll
- KeyHolder.dll
- KeyHolder.conf - этот файл содержит ключи, предназначенные только для тестирования.

Не предоставляйте KeyHolder.conf конечным пользователям, и не используйте его для вашего готового решения.

Поместите следующие файлы в корневую папку Firebird:

- libeay32.dll – этот файл имеет разрядность 64бит, не смотря на то что содержит в своем наименовании число 32.
- ucrtbased.dll
- vcruntime140d.dll

Перезапустите Firebird

Шаг 2 – шифрование БД

Подсоединитесь к БД при помощи isql и зашифруйте ее:

```
isql localhost:crypt -user SYSDBA -pass masterkey  
SQL>alter database encrypt with dbcrypt key red;
```

С этого момента БД будет зашифрована (на самом деле шифрование может занять некоторое время, если БД имеет размер гигабайт и более) с использованием ключей, которые находятся в файле KeyHolder.conf.

Проверка статуса шифрования

Проверку статусы можно выполнить в isql командой **show database** или вызовом **gstat -h <имя БД>**

```
Database "G:\DATABASES\ODS12\GREEN.FDB"
Database header page information:
  Flags                0
  Generation           21
  System Change Number 0
  Page size            8192
  ODS version          12.0
  Oldest transaction   13
  Oldest active        14
  Oldest snapshot      14
  Next transaction     14
  Sequence number      0
  Next attachment ID   12
  Implementation       HW=Intel/i386 little-endian OS=Windows CC=MSUC
  Shadow count         0
  Page buffers         0
  Next header page     0
  Database dialect     3
  Creation date        Feb 22, 2017 13:51:56
  Attributes           force write, encrypted, plugin DbCrypt

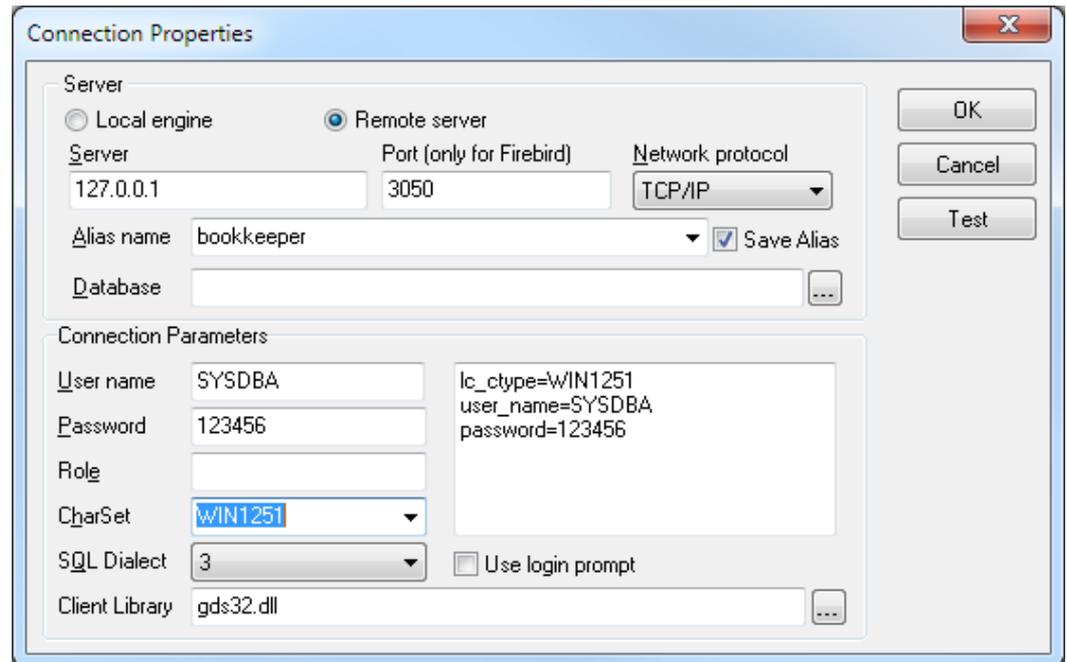
Variable header data:
  Encoded option 5, length 28
  Key hash:           ExY2I3PE7Zu+LqH7266mbCT0Qm0=
  Encryption key name: GREEN
  *END*
```

Шаг 3 – создаем клиентское приложение

- 1) Что требуется при обычном соединении
- 2) Подготовка к созданию клиента – убираем ключи с сервера
- 3) Что надо сделать в клиентском приложении для реализации шифрования

Соединение с БД без шифрования

- Сервер, порт
- Алиас или путь к файлу базы данных
- Логин и пароль
- и т.д.



Задание параметров соединения к БД без шифрования

- Код может выглядеть так:

```
FDConnection1:=TFDConnection.Create(ParentObj);
try
  FDConnection1.LoginPrompt := False;
  FDConnection1.DriverName := 'FB';
  FDConnection1.Params.Clear;
  FDConnection1.Params.Add('DriverID=FB');

  if FDCP.GetFullServerName <>' ' then begin
    //this must be like: 'Server=127.0.0.1/3050:crypt'
    FDConnection1.Params.Add('Server='+FDCP.GetFullServerName);
    FDConnection1.Params.Add('Protocol=TCPIP');
  end;
  FDConnection1.Params.Add('Database='+FDCP.DBFileName);
  FDConnection1.Params.Add('User_Name='+FDCP.UserName);
  FDConnection1.Params.Add('Password='+FDCP.Password);
  FDConnection1.Params.Add('RoleName='+FDCP.RoleName);
  FDConnection1.Params.Add('CharacterSet='+FDCP.Charset);
  FDConnection1.Params.Add('SQLDialect='+FDCP.Charset);
  FDConnection1.Params.Add('ExtendedMetadata=False');
```

Подготовка к созданию клиента

- Переносим нужные ключи из `keyholder.conf` в код приложения
- Удаляем `keyholder.conf` с сервера
 - Подключение плагина шифрования, файла ключей `keyholder.conf`, или удаление `keyholder.conf` требует перезапуска Firebird

После удаления `keyholder.conf` ни одно приложение без передачи ключа шифрования не сможет подключиться к БД

Что надо сделать в клиенте для реализации шифрования

- 1) Инициализация внутренних структур – списка ключей
- 2) Инициализация взаимодействия с сервером - BeforeConnect
- 3) Коннект и выполнение основного функционала приложения

Пример на Delphi

- В примере храним пароли в виде массива байт

```
const keyYourName: array [0..31] of byte  
= ($00, $01, $02, $03, $04 ...
```

- Структура для хранения паролей
(с поддержкой соединения с несколькими зашифрованными БД с разными ключами)

type

```
TCryptKeyValue = array [0..31] of byte;  
PCryptKeyValue = ^TCryptKeyValue;
```

```
TDBCryptKey = record  
  Name      :AnsiString;  
  pValue    :PCryptKeyValue;  
end;
```

```
TCryptKeysArray = array of TDBCryptKey;
```

Необходимо импортировать следующие функции из DLL плагина

```
function
```

```
fbcrypt_init(pszClientPathName:Pointer):integer;  
cdecl; external 'fbcrypt.dll';
```

```
function
```

```
fbcrypt_key(pszKeyName:Pointer;pKeyValue:Pointer  
; iKeyLength:Cardinal):integer; cdecl; external  
'fbcrypt.dll';
```

```
function
```

```
fbcrypt_callback(provider:Pointer):integer;  
cdecl; external 'fbcrypt.dll';
```

Класс-помощник

```
TCryptHelper = class  
public  
    constructor Create(const AClientLibrary :AnsiString);  
    destructor Destroy();override;  
    procedure GrantAccess(DBKeysArray:TCryptKeysArray;  
const AClientLibrary :AnsiString='');  
end;
```

- Все изменения объединены в модуль:
uCryptHelper.pas

Использование в коде

В удобном вам месте создаём экземпляр хелпера и набор ключей

```
type
  TfmMain = class(TForm)
    btExecuteQuery: TButton;
    edSelectToTest: TEdit;
    btConnect: TButton;
    edServerAndDB: TEdit;
    lvResult: TListView;
    procedure btExecuteQueryClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure FDConnection1BeforeConnect(Sender: TObject);
    procedure btConnectClick(Sender: TObject);
  private
    CH          :TCryptHelper;
    KeysArray   :TCryptKeysArray;
    FDCP        :TDBConParamRec;
    MRUDBList: TStrings;
  public
  end;
```

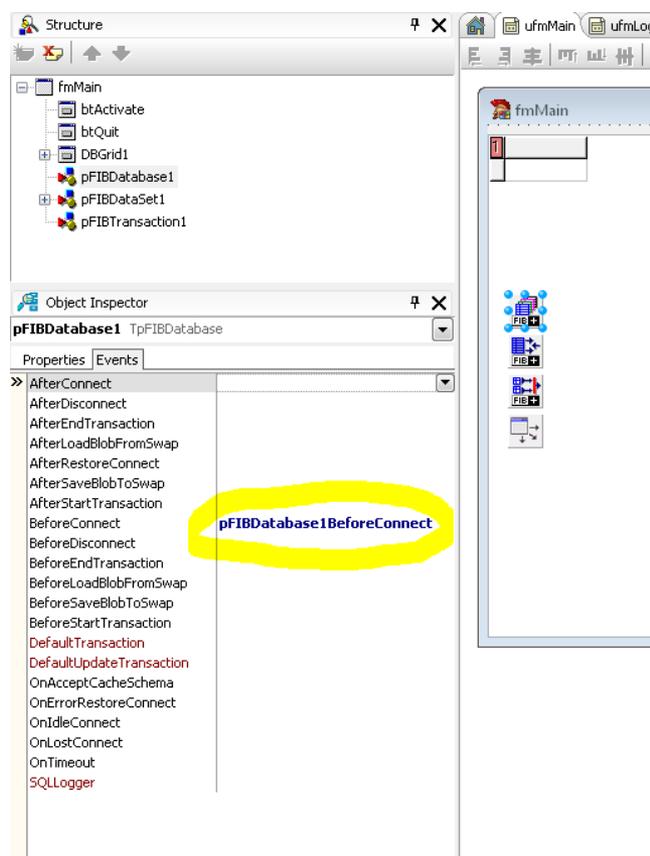
Использование в коде - 2

- Иницилируем объект указывая путь к используемой клиентской библиотеке (если необходимо)
`CH:=TCryptHelper.Create(FDCP.LibraryName);`
- В коде, до установки коннекта к серверу, инициализируем механизм шифрования, при желании уточняя путь к клиентской библиотеке:

```
procedure TfmMain.FDConnection1BeforeConnect(Sender: TObject);  
begin  
    SetLength(KeysArray, 3);  
    KeysArray[0].Name := 'Red';  
    KeysArray[0].pValue := @keyRed;  
  
    KeysArray[1].Name := 'Green';  
    KeysArray[1].pValue := @keyGreen;  
  
    KeysArray[2].Name := 'Blue';  
    KeysArray[2].pValue := @keyBlue;  
  
    CH.GrantAccess(KeysArray, FDCP.LibraryName);  
end;
```

Использование в коде-3

- Варианты — на уровне компонента/на уровне кода



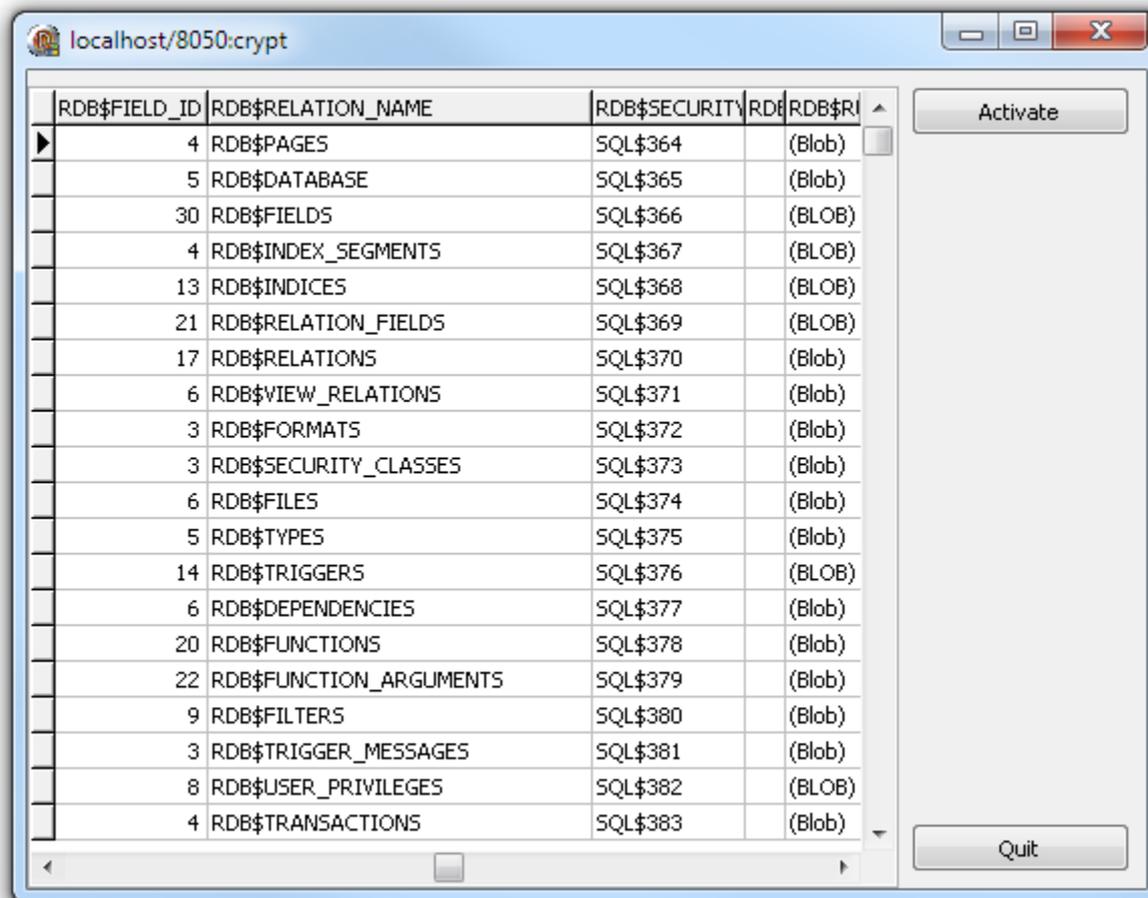
```
FDConnection1.Params.Add('Database='+FDCP.DBFileName);
FDConnection1.Params.Add('User_Name='+FDCP.UserName);
FDConnection1.Params.Add('Password='+FDCP.Password);
FDConnection1.Params.Add('RoleName='+FDCP.RoleName);
FDConnection1.Params.Add('CharacterSet='+FDCP.CharacterSet);
FDConnection1.Params.Add('SQLDialect='+FDCP.CharacterSet);
FDConnection1.Params.Add('ExtendedMetadata=False');
FDConnection1.UpdateOptions.LockWait := False;
FDConnection1.ResourceOptions.AutoConnect:=true;
FDConnection1.ResourceOptions.AutoReconnect:=true;
```

```
FDConnection1.BeforeConnect:=FDConnection1BeforeConnect;
```

```
FDTransaction1 := TFDTransaction.Create(ParentObj);
try
  FDTransaction1.Connection := FDConnection1;
  FDTransaction1.Options.Isolation := xiReadCommitted;
  FDTransaction1.Options.ReadOnly := true;
  FDTransaction1.Options.Params.Clear;
  FDTransaction1.Options.Params.Add('read');
  FDTransaction1.Options.Params.Add('read_committed');
  FDTransaction1.Options.Params.Add('rec_version');
  FDTransaction1.Options.Params.Add('nowait');

  FDConnection1.Open();
```

Проверяем работоспособность



The screenshot shows a window titled 'localhost/8050:crypt' with a table of system tables. The table has four columns: 'RDB\$FIELD_ID', 'RDB\$RELATION_NAME', 'RDB\$SECURITY', and 'RDB\$RI'. The rows list various system tables such as RDB\$PAGES, RDB\$DATABASE, RDB\$FIELDS, etc. To the right of the table are 'Activate' and 'Quit' buttons.

RDB\$FIELD_ID	RDB\$RELATION_NAME	RDB\$SECURITY	RDB\$RI
4	RDB\$PAGES	SQL\$364	(Blob)
5	RDB\$DATABASE	SQL\$365	(Blob)
30	RDB\$FIELDS	SQL\$366	(BLOB)
4	RDB\$INDEX_SEGMENTS	SQL\$367	(BLOB)
13	RDB\$INDICES	SQL\$368	(BLOB)
21	RDB\$RELATION_FIELDS	SQL\$369	(BLOB)
17	RDB\$RELATIONS	SQL\$370	(Blob)
6	RDB\$VIEW_RELATIONS	SQL\$371	(Blob)
3	RDB\$FORMATS	SQL\$372	(Blob)
3	RDB\$SECURITY_CLASSES	SQL\$373	(Blob)
6	RDB\$FILES	SQL\$374	(Blob)
5	RDB\$TYPES	SQL\$375	(Blob)
14	RDB\$TRIGGERS	SQL\$376	(BLOB)
6	RDB\$DEPENDENCIES	SQL\$377	(Blob)
20	RDB\$FUNCTIONS	SQL\$378	(Blob)
22	RDB\$FUNCTION_ARGUMENTS	SQL\$379	(Blob)
9	RDB\$FILTERS	SQL\$380	(Blob)
3	RDB\$TRIGGER_MESSAGES	SQL\$381	(Blob)
8	RDB\$USER_PRIVILEGES	SQL\$382	(BLOB)
4	RDB\$TRANSACTIONS	SQL\$383	(Blob)

Обработка ошибок

- Сервер выдает следующие ошибки
 - неверный ключ
Invalid crypt key RED
 - ключ не найден
Key not set

Пример обработчика ошибок на клиенте

- Уровень инициализации клиента

```
begin
  if AClientLibrary<>' ' then begin //Update ClientLibrary path if it needed
    ClientLibrary:=AClientLibrary;
  end;

  if Length(DBKeysArray)=0 then begin
    raise Exception.Create('At least one key must be present in DBKeysArray!');
  end;

  if (fbcrypt_init(PAnsiChar(ClientLibrary)) < 0) then begin
    raise Exception.Create('fbcrypt_init failed');
  end;

  for f:=0 to Length(DBKeysArray)-1 do begin
    if (fbcrypt_key(PAnsiChar(DBKeysArray[f].Name), DBKeysArray[f].pValue, sizeof(DBKeysArray[f].pValue^)) < 0) then begin
      raise Exception.Create('fbcrypt_key failed');
    end;
  end;

  ($IFDEF FBPROV)
  if (fbcrypt_callback(prov) < 0) then begin
  ($ELSE)
  if (fbcrypt_callback(nil) < 0) then begin
  ($ENDIF)
    raise Exception.Create('fbcrypt_callback');
  end;
end;
```

На чем тестировали

- Non UNICODE Delphi 2007
- UNICODE Delphi — XE8, XE10
- Компоненты
 - FireDAC
 - FIBPlus

Firebird Encryption Plugin Framework

Откуда берутся файлы плагина из рассмотренного примера?

- Firebird Encryption Plugin Framework – набор исходных кодов для создания собственной версии плагина

DBCrypt – серверная часть плагина

FBCrypt – клиентская часть (обертка для функций обмена ключами)

Framework – немного деталей

- Среда разработки - Visual Studio / gcc / etc +
- Нужны исходники Firebird
- Добавляем исходники плагина шифрования
- Получаем файлы lib/obj/dll (dll-для windows Delphi)
- Необходимый набор функций (для приложения):

```
extern "C" int fbcrypt_init(const char* clientPathName);
```

```
extern "C" int fbcrypt_key(const char* name,  
    const unsigned char* data, unsigned dl);
```

```
extern "C" int fbcrypt_callback(void* provider);
```

Зависимости

- OpenSSL (*для Windows libeay32.dll dynamic или static*)
- MS Runtime [debug]: msvc*.dll, vcruntime*.dll
- Файлы
 - fbclient.dll – обычный (из дистрибутива)
 - fbcrypt.dll – обертка функций обмена ключами с сервером
 - libeay32.dll - openssl
 - vcruntime140.dll
 - firebird.conf (при необходимости)

Краткая шпаргалка по шифрованию

- Определить/создать свои пароли для кодирования и декодирования БД
- Зашифровать базу данных.
Для этого надо разместить созданные пароли в файле keyholder.conf на сервере и используя isql, подключиться к нешифрованной базе, выполнить

```
alter database encrypt with "DbCrypt" key MyStrongest;
```
- Внести в код клиентского приложения заданные ключи, удалить keyholder.conf с сервера, для блокировки неавторизованного доступа (без явной передачи паролей, т.е. включая gbak, isql и т.д.)
- Модифицировать код, добавив инициализацию помощника (TCryptHelper), перед установкой коннекта
- Пересобрать приложение с новым кодом

ВОПРОСЫ?

support@ibase.ru

Инструкция и пример (с исходниками клиентского приложения):

<http://www.ibase.ru/download-demo-firebird-encryption-plugin>