Subject: sqlite3 encryption
Posted by jimlef on Sat, 22 Oct 2022 04:12:41 GMT
View Forum Message <> Reply to Message

Just ran across this... Anyone else seen it? Any thoughts?

Encryption for SQLite3

Jim

Subject: Re: sqlite3 encryption
Posted by coolman on Sat, 22 Oct 2022 06:24:52 GMT
View Forum Message <> Reply to Message

U++ sqlite plugin already uses the above implementation.

Subject: Re: sqlite3 encryption
Posted by jimlef on Sat, 22 Oct 2022 11:42:30 GMT
View Forum Message <> Reply to Message

Well well Now I have to try this out - thank you!

Quote:coolman
U++ sqlite plugin already uses the above implementation.

Subject: Re: sqlite3 encryption
Posted by jimlef on Sat, 22 Oct 2022 17:54:46 GMT
View Forum Message <> Reply to Message

So, hit and run q here (got to leave for work shortly) but...
PasswordDlg dlg;
dlg.Run() // ... <- to get password of course
sqlite3.Open(dbname, password, CODEC_TYPE_SQLCIPHER);

Is that change enough to use the encryption? Another related link here, describing the c interface... have to read this all later.
https:// utelle.github.io/SQLite3MultipleCiphers/docs/configuration/c onfig_capi/

Subject: Re: sqlite3 encryption
Posted by coolman on Sat, 22 Oct 2022 18:18:58 GMT
View Forum Message <> Reply to Message

jimlef wrote on Sat, 22 October 2022 19:54
PasswordDlg dlg;
dlg.Run() // ... <- to get password of course
sqlite3.Open(dbname, password, CODEC_TYPE_SQLCIPHER);

Yes, this is enough. Take a look to the source code of the plugin Sqlite3upp.cpp

---

Subject: Re: sqlite3 encryption
Posted by jimlef on Sun, 23 Oct 2022 10:51:45 GMT
View Forum Message <> Reply to Message

Ok, I think I have the basic idea, Thanks!

Source

Subject: Re: sqlite3 encryption
Posted by coolman on Sun, 23 Oct 2022 13:04:14 GMT
View Forum Message <> Reply to Message

You can check encrypted Sqlite before Open() the DB using the function

```
bool Sqlite::IsFileEncrypted(const char *DBfilename) {
    FileIn in(DBfilename);

    if (!in) {
        return false;
    }

    String SqlVersion = "SQLite format 3";
    int SqlVersionLength = SqlVersion.GetCount();
    in.Seek(0);
    String version = in.Get(SqlVersionLength);
    in.Close();

    return (!version.IsEqual(SqlVersion));
}
```

With combination of returned code from Open() function

```
...
        int errCode = sqlite3db.GetErrorCode();
```

```
        String errMsg = Format("[ %s&&Error: %d (%s)]", t_("Loading the database has failed!"),
errCode, sqlite3db.GetErrorCodeString());
        if (SQLITE_NOTADB == errCode) {
            errMsg = t_("The database is encrypted but decryption failed!&[= Did you use the correct
password?");
        }
        ErrorOK(errMsg);
...
```

BR, Radek

Subject: Re: sqlite3 encryption
Posted by jimlef on Mon, 24 Oct 2022 08:49:45 GMT
View Forum Message <> Reply to Message

Thank you again - I've done a bit more, but I wanted SQLCIPHER, so I 'updated' the sqlite3upp plugin on my side. I put the newest amalgamation files from here in lib (sqlite3mc_amalgamation.c & .h). I removed the aeshardware.c section (threw compile errors) from sqlite3mc_amalgamation.c & updated the Sqlite3.h & sqlite3upp.cpp files:

```
 int  ChangePassword(const String& password, int cipher = CIPHER_SQLCIPHER);
 int  CheckDBAccess();
 bool Open(const char *filename, const String& password = Null, int cipher =
CIPHER_SQLCIPHER);
...
 enum Ciphers {
  CIPHER_CHAHA2020_SQLEET,
  CIPHER_CHAHA2020_DEFAULT,
+  CIPHER_AES256,
+  CIPHER_SQLCIPHER
 };
```

For Sqlite3upp.cpp:

```
int Sqlite3Session::SetDBEncryption(int cipher) {
 // "default:cipher" => use SQLCipher during the entire lifetime of database instance
 // CIPHER_CHAHA2020_SQLEET settings are backward compatible with the previous sqleet
implementation in the U++
 // Note: It is not recommended to use legacy mode for encrypting new databases. It is supported
for compatibility
 // reasons only, so that databases that were encrypted in legacy mode can be accessed.

 int retcode = SQLITE_ERROR;
 switch (cipher) {
  case CIPHER_CHAHA2020_DEFAULT: {
```

```
   int value = sqlite3mc_config(db, "default:cipher", CODEC_TYPE_CHACHA20);
   if (value != -1)
    value = sqlite3mc_config_cipher(db, "chacha20", "kdf_iter", 64007);
   if (value != -1)
    value = sqlite3mc_config_cipher(db, "chacha20", "legacy", 0);
   if (value != -1)
    value = sqlite3mc_config_cipher(db, "chacha20", "legacy_page_size", 4096);
   if (value != -1)
    retcode = SQLITE_OK;
   } break;
  case CIPHER_CHAHA2020_SQLEET: {
   int value = sqlite3mc_config(db, "default:cipher", CODEC_TYPE_CHACHA20);
   if (value != -1)
    value = sqlite3mc_config_cipher(db, "chacha20", "kdf_iter", 12345);
   if (value != -1)
    value = sqlite3mc_config_cipher(db, "chacha20", "legacy", 1);
   if (value != -1)
    value = sqlite3mc_config_cipher(db, "chacha20", "legacy_page_size", 4096);
   if (value != -1)
    retcode = SQLITE_OK;
  } break;
+  case CIPHER_AES256: {
+   int value = sqlite3mc_config(db, "default:cipher", CODEC_TYPE_AES256);
+   if (value != -1)
+    value = sqlite3mc_config_cipher(db, "aes256", "kdf_iter", 4001);
+   if (value != -1)
+    value = sqlite3mc_config_cipher(db, "aes256", "legacy", 0);
+   if (value != -1)
+    value = sqlite3mc_config_cipher(db, "aes256", "legacy_page_size", 0);
+   if (value != -1)
+    retcode = SQLITE_OK;
+  } break;
+  case CIPHER_SQLCIPHER:
  default: {
+   int value = sqlite3mc_config(db, "default:cipher", CODEC_TYPE_SQLCIPHER);
+   if (value != -1)
+    value = sqlite3mc_config_cipher(db, "sqlcipher", "kdf_iter", 256000);
+   if (value != -1)
+    value = sqlite3mc_config_cipher(db, "sqlcipher", "legacy", 0);
+   if (value != -1)
+    value = sqlite3mc_config_cipher(db, "sqlcipher", "legacy_page_size", 4096);
+   if (value != -1)
+    value = sqlite3mc_config_cipher(db, "sqlcipher", "hmac_use", 1);
+   if (value != -1)
+    value = sqlite3mc_config_cipher(db, "sqlcipher", "hmac_pgno", 0);
+   if (value != -1)
+    value = sqlite3mc_config_cipher(db, "sqlcipher", "hmac_salt_mask", 58);
+   if (value != -1)
```

```
+    value = sqlite3mc_config_cipher(db, "sqlcipher", "kdf_algorithm", 2);
+   if (value != -1)
+    value = sqlite3mc_config_cipher(db, "sqlcipher", "hmac_algorithm", 2);
+   if (value != -1)
+    value = sqlite3mc_config_cipher(db, "sqlcipher", "plaintext_header_size", 16);
+   if (value != -1)
+    retcode = SQLITE_OK;
+  } break;
 }
 return retcode;
}
```

I haven't modded any other code sections.

I'm sure that the aeshardware section can be fixed to eliminate the aes128 related errors (well, hopeful). This otherwise seems to update things quite a bit. I'm sure I'll run into plenty of introduced issues as I code more into this project of mine, but it's a start

---