Secure Your Future, Encrypt Your Data

By Chris Hardekopf

In BBj® 4.0 and the current PRO/5® family (Visual PRO/5® 5.0 and PRO/5 5.0), BASIS added data encryption to the arsenal of tools available to all BBX® programmers. Data encryption gives programmers the tools to prevent individuals without the correct password from reading the contents of files or fields, either through the interpreter, or directly from the media. As such, it is a valuable security tool.

Keep in mind, however, data encryption by itself does not solve programmers' security problems. The best use of encryption is in conjunction with other security measures to form a complete security system. This article discusses the new data encryption capabilities and provides the developer with some general guidelines for using these tools in their BASIS applications.

The first thing to realize is that by its nature, data encryption imposes a performance penalty on reading and writing the encrypted data. Encrypting and decrypting data as it is read and written obviously imposes a higher load than reading and writing alone. This overhead is generally CPU intensive and the level of intensity depends on the encryption algorithm used. This means that in general, programmers should only encrypt what is necessary to protect. This recommendation also implies that the developer should not use encryption indiscriminately, rather use it selectively, as determined by the security requirements of the project.

The implementation of data encryption in BBj and the PRO/5 family uses the AES-128 (Advanced Encryption Standard) algorithm and CBC (Cipher Block Chaining) to encrypt data. It is not important that programmers understand these algorithms except to know that it is a public standardized method and one that the industry generally accepts as secure.

We provide two different ways of encrypting data. The first is file encryption. This method encrypts the entire file during file creation and only requires that the developer provide a password when opening the file. For example:

```
REM Demo for encrypted file creation and use.
LET FILENAME$="mkeyed_enc.dat"
LET REC_TMPL$="NUM:N(5),SECRET:N(5)"
LET ENC_MODE$="CRYPTPASS=secret"

REM Create an encrypted mkeyed file.
ERASE FILENAME$,ERR=CREATE
CREATE:
MKEYED FILENAME$,[1:5],0,10,MODE=ENC_MODE$

REM Open the encrypted file.
LET CHAN = UNT
OPEN(CHAN,MODE=ENC_MODE$)FILENAME$

REM Write data to the file.
DIM REC$:REC_TMPL$
FOR I = 0 TO 100
LET REC.NUM = I
 LET REC.SECRET = I+1
 WRITE RECORD(CHAN)REC$
NEXT I

REM Read a record from the file.
REC.NUM = 50
READ RECORD(CHAN,KEY=REC$(1,5))REC$

REM Close the file.
CLOSE(CHAN)
```

continued...

The second method is field encryption. This method provides complementary ENCRYPT and DECRYPT functions which, as implied by the names, encrypt and decrypt data respectively given a password. For example:

```
REM Demo for ENCRYPT/DECRYPT functions.
LET FILENAME$="mkeyed.dat"
LET REC_TMPL$="NUM:N(5),SECRET:N(5)"
LET ENC_MODE$="CRYPTPASS=secret"

REM Determine the encrypted size of a field.  This is a one time activity,
REM since the size is constant for a given size of input string regardless
REM of password or contents.
LET TEST_ENC$=ENCRYPT("XXXXX",mode=ENC_MODE$)
LET ENC_LEN=LEN(TEST_ENC$)

REM Template for the encrypted record.
LET REC_ENC_TMPL$="NUM:N(5),SECRET_ENC:C("+STR(ENC_LEN)+")"

REM Create a regular mkeyed file.
ERASE FILENAME$,ERR=CREATE
CREATE:
MKEYED FILENAME$,[1:5],0,5+ENC_LEN

REM Open the file.
LET CHAN = UNT
OPEN(CHAN)FILENAME$

REM Write data to the file.
DIM REC$:REC_TMPL$
DIM REC_ENC$:REC_ENC_TMPL$
FOR I = 0 TO 100
      LET REC.NUM = I
      LET REC.SECRET = I+1

      LET REC_ENC.NUM = REC.NUM
      LET REC_ENC.SECRET_ENC$ = ENCRYPT(REC$(5,5),MODE=ENC_MODE$)
      WRITE RECORD(CHAN)REC_ENC$
NEXT I

REM Read a record from the file.
REC_ENC.NUM = 50
READ RECORD(CHAN,KEY=REC_ENC$(1,5))REC_ENC$
REC.NUM = REC_ENC.NUM REC$(5,5) = DECRYPT(REC_ENC.SECRET_ENC$,MODE=ENC_MODE$)

REM Close the file.
CLOSE(CHAN)
```

BASIS provides these two different mechanisms for encrypting data to offer the developer a choice based upon their particular situation and requirements. Each method has unique advantages and disadvantages.

It is much easier for developers to encrypt an entire file since they only need to create the encrypted file and change the way the program opens the file. All other file activity is identical to unencrypted files. In particular, this means, that encryption has no effect on how developers create and use keys. This method is easier to implement, however, it suffers more runtime performance degradation than using the explicit ENCRYPT and DECRYPT functions.

The ENCRYPT and DECRYPT functions allow developers to encrypt and decrypt specific information on demand. These functions require explicit code, which some developers may find more difficult to use. In addition, ENCRYPT and DECRYPT

continued...

functions impose constraints on how the developers use the encrypted fields as keys in the file. Encrypted fields are accessible directly as a key, but a developer cannot iterate over them in any meaningful fashion since the file itself only sees the encrypted data.

Using data encryption in an application involves some important decisions. Follow these general guidelines when determining the appropriate use of data encryption in your application:

1. **Determine what data actually needs encryption**. What can snoopers read if/when they breach the file security? Determine if there are any legal or industry requirements to meet for data encryption. Keep in mind that encryption imposes a performance penalty, so keep the amount of encrypted data to a minimum.

2. **Establish how to use the encrypted data.** Do you need to use the data as a key? If so, encrypt the entire file. Will the application address the data directly? In this case, there is no constraint imposed on the type of encryption used.

3. **Identify how much of the data requires encryption**. When all the data in a file requires encryption, encrypt the entire file. When the file contains only a portion that requires encryption, consider splitting the file, and encrypt only the new file now containing the data needing encryption. The data not requiring encryption remains in a normal unencrypted file. Splitting the file is also a good solution when only a few people require the encrypted data. If only a few fields require encryption, it is faster to encrypt just those fields.

4. **Decide how to handle passwords for data access**. This is of paramount importance! The encryption is useless, even dangerous, if the developer does not handle passwords correctly. After choosing passwords, follow these two rules:

   Be sure the passwords are available to every entity that needs to read the data. The correct password is the only way to read the encrypted data. There is no back door. If the password is unavailable, then the data is unreadable.

   Be sure that the password is not available to those that should not, or do not, need to read the data. Encrypting data does no good if people that should not have access know the password. When a user requires access to the data through an application, give the application the password, not the user.

The importance of passwords also leads to some further advice and caution. Developers should make it easy to change passwords by not hard coding them or by doing so in only a few places. If someone compromises a password, it is easy to re-encrypt files and fields with a new password. However, changing the password used to access the data is only as easy or hard as the developer makes it. Data encryption is like an uncrackable safe. The password is the combination to the safe so be cautious using and distributing it. Again, if the password is misplaced, all of the encrypted data is useless with no way to retrieve it. Unencrypted backups are the only recourse.

People lock cars and homes to prevent theft and invasion. With the same concern for security, companies should protect their most valuable asset, their data. Identity theft is real. Reports of stolen credit card information are commonplace. BASIS' new data encryption capabilities in PRO/5 and BBj complete the arsenal of tools, arming the developer with an industry-standard encryption algorithm, configurable passwords, and file and field-level encryption.

Secure your future. Encrypt your data.