“Secure Password Managers” and “Military-Grade Encryption” on Smartphones: Oh, Really?

Andrey Belenko and Dmitry Sklyarov

Elcomsoft Co. Ltd.
{a.belenko, d.sklyarov} @ elcomsoft . com
http://www.elcomsoft.com

Abstract. In this paper we will analyze applications designed to facilitate storing and management of passwords on mobile platforms, such as Apple iOS and BlackBerry. We will specifically focus our attention on the security of data at rest. We will show that many password keeper apps fail to provide claimed level of protection.

1 Introduction

We live in the era of mobility and mobile computing. Mobile devices are continually becoming smaller, more powerful, and, consequentially, smarter.

The share between smartphones and conventional mobile phones is shifting towards smarter devices that can be used to perform very different tasks: from replying to an email and browsing the web to navigating on unfamiliar roads (or even in the airspace). In fact, the variety of tasks that can be performed on smart devices is now limited primarily by the availability of the applications, not by the device constraints themselves. Today's mobile devices are no longer restrictive hardware running rudimentary operating system; they are general-purpose computers running full-featured OS.

As the variety of tasks that can be done on mobile devices grows so does the need to store (and securely access) private and confidential data on those devices. One example of such confidential information is passwords. It is well known that passwords should be complex and that one should not reuse the same password for different services, no matter how complex that password might be [1]. Those requirements create a challenge of remembering dozens of complex passwords, something an average human being is not very good at.

Luckily, there's an app for that. Many apps, in fact. There is no shortage of password managers for mobile platforms. Since the apps will be entrusted with sensitive data, the questions arise about their security. We will explore this question in more detail in this paper. Specifically, we will analyze whether password managing applications are utilizing security mechanisms provided by mobile OS and whether they add any additional security on top of that.

We will focus solely on the "data at rest" aspect of security and thus will not consider network attacks. We will consider Research in Motion BlackBerry and Apple iOS platforms only.

2 Threat Model

One important notice is that most mobile devices today do not have physical keyboard, making it harder for users to utilize motor learning to remember complex passwords. Therefore we believe it is safe to assume that, on average, the complexity of a password that has to be entered routinely on a mobile device will be lower than that of a password that is only used on the devices with physical keyboards.

Intuitively, this obvious limitation of mobile devices should be addressed by vendors of password management apps (or other apps relying on password authentication) by making offline attacks as hard and slow as practical, e.g. via utilizing platform security services or by use of cryptography.

All password managers we analyze in this paper maintain local storage of users' passwords, require master password to access it, and many of them employ different protection methods to restrict access to the data without known master password.

In this paper we assume that the attacker wants to recover master password for password manager(s) installed on the mobile device and/or to extract the passwords stored in those managers. The attacker has the device in her possession or she has a backup copy of the device, or she is somehow else able to access password manager database. We will discuss how she can obtain such database in the next section.

3 Obtaining (protected) Password Keeper Databases

3.1 Apple iOS

On Apple iOS platform, two primary sources of app data files are device backups and the device itself.

Device Backup. All password managers we analyze in this paper allow migrating of the users' passwords between devices by allowing iOS to backup data (files and/or keychain records) comprising app state.

iOS offers user-configurable backup encryption. When backup encryption is enabled, device will encrypt all backup data prior to sending it to iTunes (i.e. the device performs actual encryption, not the iTunes). To disable backup encryption, original backup password must be supplied.

Subset of device keychain is included in the backup. If the backup is not encrypted, then exported keychain items are encrypted using the device key and thus decryption without access to the original device is not possible. If the backup is encrypted, however, exported keychain items will be encrypted using the key derived from backup password, thus making it possible to decrypt them if backup password is known.

Backup encryption key is computed by performing 10'000 iterations of PBKDF2-SHA1 function with password as an input, therefore password recovery is relatively slow (tens of thousands passwords per second with GPUs).

In order for iTunes to create a backup of the device, the device must be unlocked (i.e. passcode entered by the user) or it must have been previously paired with this PC/Mac running iTunes. Pairing is done automatically when unlocked device is first connected.

In case of iOS 5 existing pairing can only be used if the device have been unlocked at least once since it was powered on or rebooted.

Device. Application data files may be read directly from the device using tools that utilize Apple MobileDevice framework (two examples of such tools are iExplorer [2] and i-Fun-Box [3]). The device must not have a passcode set, or it must have a pairing with a computer where tool is being run. Device need not to be jailbroken for this to work since application files reside within application sandboxes on the user partition.

Physical Acquisition. It is also possible to acquire physical image of the filesystem and decrypt required files from it. This method is commonly used in iOS forensics and is currently possible for all devices preceding iPad 2 and iPhone 4S. Device passcode is not a problem for the acquisition itself, but may be required to actually decrypt required file(s), depending on the protection class a particular file belongs to.

If device passcode is needed to decrypt file or keychain item, it is possible to mount a passcode recovery attack. It is extremely slow (2-6 passcodes per second) and can only be done on the device itself (i.e. offline passcode recovery is not possible).

3.2 BlackBerry

On BlackBerry, the primary source of app data is device backup.

Device Backup. BlackBerry Desktop Software offers user-configurable backup encryption. When enabled, Desktop Software will encrypt current backup using user-supplied password. Encryption is done by the Desktop Software, not the device. Encryption of future backups can be disabled without entering the original password.

Backup encryption key is computed by performing 20'000 iterations of PBKDF2-SHA1 function with password as an input, therefore password recovery is relatively slow (tens of thousands passwords per second with GPUs).

In order for Desktop Software to create a backup of the device, the device password must be known.

Physical Acquisition. There are products on the market claiming the ability to perform physical acquisition of BlackBerry smartphones [4]. Unfortunately, they cannot do this on a device with an unknown device password and thus do not provide any benefit in our threat model – if device password is known, an attacker can simply obtain unencrypted device backup that contains password manager database(s).

In certain circumstances it is possible to recover BlackBerry device password using encrypted media card from the device [5]. To the best of our knowledge, currently there are no other ways to recover BlackBerry device password.

4 iOS Password Management Apps

As we have mentioned before, there is no shortage of password managers, password keepers and password vaults in App Store. Analyzing all of them would require enormous amount of time, therefore we have picked some most popular and most "interesting" ones. The list includes both paid and free apps; it is by no means exhaustive.

4.1 Free Apps

Keeper® Password & Data Vault [6]. Like many other password management apps, this one uses SQLite database as a storage backend and the data in the database is encrypted. Advanced Encryption Standard (AES) cipher with 128-bit key in ciphertext block chaining (CBC) mode is used for encryption. Encryption key is computed from master password as first 16 bytes of SHA-1 of master password.

Master password verification is performed by comparing MD5 hash of supplied master password against a MD5 hash of a correct master password which is stored in the same database. Hash is plain MD5, without any salt. Thus it is possible to use readily available high-performance MD5 hash cracking tools and MD5 Rainbow Tables to recover the password.

Password Safe - iPassSafe free version [7]. This app also uses SQLite database to store user passwords. The data is encrypted using AES cipher with 256-bit key in CBC mode. The encryption master key is randomly generated. This master key is further encrypted with master password and then stored in the database.

The app blacklists the following master passwords as being insecure: `0000, 1234, 2580, 1111, 5555, 0852, 2222, 1212, 1998, 5683`.

Master password is not hashed before being used as an encryption key; it is only null-padded to 32 bytes. Furthermore, since (random) master key is always 32 bytes long, and it is PKCS7-padded prior to encryption, the last block of the plaintext that gets encrypted on the master password key will be 16 bytes all equal to 0x10. This allows to efficiently verify master passwords; such verification requires only one trial AES decryption.

My Eyes Only™ - Secure Password Manager [8]. This app stores master password, the answer to the secret question to recover the password, and the RSA public and private keys in the keychain, thus its security is at most as good as keychain's one. The data is stored with a protection class `kSecAttrAccessibleWhenUnlocked` meaning in case of physical acquisition it can only be decrypted if device passcode is known or escrow keys are available. Keychain can also be decrypted if encrypted device backup is available and the backup password is known.

Actual user data is stored in the files in application sandbox and is encrypted with RSA key.

The app also maintains a file which contains public and private RSA keys (same ones as in the keychain), master password, and answer to secret question, both encrypted using aforementioned RSA keys. The fact that private RSA key is stored in the file allows to instantly decrypt master password (as well as any other record in the database).

The app uses RSA with 512 bit modulus which, by today's standard, is not sufficient. Even if app wouldn't store private key along with encrypted data, factoring 512 bit RSA modulus is fairly easy.

Strip Lite - Password Manager [9]. This product uses SQLite database for storage, but it takes a different approach in protecting the data. Instead of per-record or per-column data encryption it encrypts SQLite database file in its entirety. It does so with help of an open-source component SQLCipher [10] developed by the same company. The encryption key is computed as PBKDF2-SHA1 with 4'000 iterations from master password and per-database salt. Password verification requires computing an encryption key (thus, 4'000 iterations of PBKDF-SHA1) and one trial AES-256 decryption.

Safe - Password [11]. This app is also known as Awesome Password Lite [12] and as Password Lock Lite [13]. Those apps store passwords in SQLite database. Master password is limited to 4-digits PIN and is stored in the same database in plaintext. Obviously, master password can be instantly recovered. User data (including passwords) is also stored in plain.

iSecure Lite - Password Manager [14]. This app has the same problem as the previous one – the master password protecting its database is stored in plaintext. It can be recovered instantly. User data is also not encrypted.

Ultimate Password Manager Free [15]. This app comes in two versions: paid and free. According to its author, one of the limitations of the free version is "no data encryption". This is indeed true, but not only this app provides no data encryption, it also stores master password in plaintext. Password recovery is instant.

Secret Folder Lite [16]. This is not a password manager application, but rather a different kind of app that offers password protection for files (in this case photos and videos).

This particular app was created by the same author as Password Lock Lite and employs similar ‘protection’: passwords are stored in SQLite database in plaintext. They can be recovered instantly.

4.2 Paid Apps

SafeWallet - Password Manager ($3.99) [17]. This mobile app uses very same database format as used by desktop versions of the same application. Single password verification requires computing PBKDF2-SHA1 with 10 (ten) iterations and one trial AES-256 decryption. User data is encrypted using the key derived from the master password.

SplashID Safe for iPhone ($9.99) [18]. This app uses SQLite database and encrypts data in it like most other apps we have analyzed, only it uses Blowfish instead of AES. Master password is used as a Blowfish key to encrypt user data.

What sets this app apart, however, is that it stores master password in the database using reversible encryption. That is, it uses hard-coded key “`g.;59?^/0n1X*{OQ1Rwy`” to encrypt master password using Blowfish algorithm and then stores the result in the database. Obviously, the master password can be instantly recovered by sinply decrypting the data.

DataVault Password Manager ($9.99) [19]. This app uses system keychain as a storage backend. It stores user data as well as program options in the keychain. It uses protection class `kSecAttrAccessibleWhenUnlocked` for its records, meaning they can only be decrypted if device passcode is known or if encrypted device backup is available and backup password is known.

App also additionally encrypts data before writing it to the keychain. The encryption is AES-128 in electronic code book (ECB) mode; encryption key is master password padded to 16 bytes and used directly, without hashing. Only the first 16 characters are used to compute the key, others are simply ignored.

For master password verification the app stores SHA-256 hash of the correct master password in the keychain. However, for some reason it stores this hash in the Comments attribute of the keychain item, not in the Data attribute as it probably should.

Before iOS 5 only Data attribute of the keychain items was encrypted, and thus hash of the master password is available without the need to decrypt keychain data; keychain database file (either from the device or from the unencrypted device backup) is all what is needed to extract password hash and mount a password recovery attack. Password hashing is done without salt and readily available high-performance tools and Rainbow Tables can be used to recover it.

Starting with iOS 5 Apple encrypts all attributes of a keychain item, not just the Data attribute. But to enable searching for a keychain items by their attributes it also stores SHA-1 hash of the values of the attributes. Therefore, in case of iOS 5 password recovery attack is still possible but it will require additional computation of SHA-1 hash of computed SHA-256 hash of candidate password. Precomputed tables (Rainbow Tables) can also be built for this case.

mSecure - Password Manager ($9.99) [20]. This app uses custom, Apple-specific storage backend based on NSKeyedArchiver Foundation class. Data stored in this database is encrypted with Blowfish encryption algorithm; SHA-256 hash of a master password is used as an encryption key.

Master Password is verified by decrypting a password verifier stored in the database and comparing it to a known hard-coded value. Password verifier is encrypted using same Blowfish key as the user data. Password verification, thus, requires one SHA-256 computation and one trial Blowfish decryption.

LastPass for Premium Customers ($1/month) [21]. LastPass is somewhat different from other apps we have analyzed. It is a subscription-based service, not a standalone app. However, the app can also work in offline mode (provided that user has previously logged in and downloaded data from the server). The app maintains local storage of user data.

To facilitate offline master password verification app stores an encrypted password hash; the encryption key is based on the password. The procedure looks like this:

```
Key  := SHA256 (Username + Password)
Hash := SHA256 (Key)
LoginHash     := ReadProperty (“valid_login_hash”)
DecryptedHash := AES-256-Decrypt (Key, LoginHash)
If Hash = DecryptedHash Then password is correct
```

Password recovery attacks are possible, password verification requires two computations of SHA-256 and a trial AES-256 decryption.

1Password Pro ($14.99) [22]. 1Password uses SQLite database for storage. App allows users to specify two master passwords: a master PIN guarding access to all records, and an optional (more complex) master password guarding selected items in the database. Actual data is encrypted using key derived from either the master PIN, or from the master password, if user has marked this particular item for additional protection.

Encryption key used to protect items is encrypted on the key derived from master PIN or master password and is stored in the database. To verify PIN or password a so-called Validator is also stored in the database. The Validator is database encryption key encrypted on itself. When user supplies PIN or password, app computes a key encryption key (KEK) from it, decrypts database encryption key with it, and then decrypts Validator using database encryption key. If decrypted Validator is equal to database encryption key then the database encryption is correct, and, consequently, supplied PIN or password is also correct. The pseudocode for the process is as following:

```
Read EncryptedDatabaseKey and EncryptedValidator from Database
KEK := MD5 (Password + Salt)
IV := MD5 (KEK + Password + Salt)
```

```
DatabaseKey := AES-128-CBC (KEK, IV, EncryptedDatabaseKey)
Validator := AES-128-CBC (DatabaseKey, NULL, EncryptedValidator)
If Validator = DatabaseKey Then password is correct
```

However, because PKCS7 padding is used when encrypting database encryption key, it is possible to verify password just by computing KEK (using MD5 hash function), decrypting last block of encrypted database key, and checking if it equals to 16 bytes with value 0x10 (this will be the PKCS7-compliant padding when encrypting data whose length is exactly *N* blocks of underlying cipher). Thus, very fast password recovery attack is possible, requiring one MD5 computation and one AES trial decryption per password.

5 BlackBerry Password Management Apps

There is a number of password managers for BlackBerry platform as well. However, in this paper we will study only two of them, both created by Research In Motion itself. The protections employed in both products are quite similar.

5.1 BlackBerry Password Keeper

Password Keeper encrypts stored passwords using key derived from user-supplied master password. The encryption key is computed using 3 (three) iterations of PBKDF2-SHA1 function. To ensure integrity of the data, SHA-1 digest of the data is computed and appended before encryption. After decryption, last 20 bytes are assumed to be SHA-1 hash of the data (first *Len*-20 bytes). If computed digest matches the decrypted one, decrypted data is assumed to be correct.

This integrity verification procedure allows to mount a master password recovery attack. Due to the fact that insufficient number of PBKDF2 iterations is used (only three), the password recovery is fast (millions of passwords per second on modern CPU).

5.2 BlackBerry Wallet

BlackBerry Wallet [23] is a more advanced version of Password Keeper and is available free of charge from the BlackBerry App World. Two versions are currently available: version 1.0 for devices running BlackBerry OS 5.0 and version 1.2 for devices running newer version of BlackBerry OS.

Version 1.0. This version stores SHA256 digest of SHA256 digest of a password in the database. This allows to mount an extremely fast password recovery attack, possibly using pre-computed Rainbow Tables or high-performance GPU password crackers.

Version 1.2. The protection in this version is somewhat similar to the one present in BlackBerry Password Keeper. Encryption key is derived from user password with the help of PBKDF2-SHA1 transformation. However, the number of iterations is now variable (between 50 and 100 iterations) and SHA512 hash of a password is used as an input to PBKDF2 function. Data integrity is ensured in the same way as in Password Keeper: SHA-1 digest of the data is encrypted along with data itself. Password recovery is possible and is quite fast, although significantly slower than for BlackBerry Password Keeper or BlackBerry Wallet version 1.0.

6 Summary

We have analyzed more than a dozen password manager/password keeper applications and very few of them provide reasonable layer of protection beyond one offered by OS itself (i.e. backup encryption or passcode protection).

Many of the products clearly misuse available protection mechanism and/or use insufficient cryptographic protection. Our findings are summarized in the following tables.

Table 1. Summary of our findings.

App name	Encrypts data?	Uses keychain?	Password verification complexity
Keeper® Password & Data Vault	Yes	No	1x MD5
	Remarks: Rainbow Tables and GPU crackers may be used		
Password Safe - iPassSafe free version	Yes	No	1x AES-256
My Eyes Only™ - Secure Password Manager	Yes	Yes	N/A (see Remarks)
	Remarks: Master password and user passwords can be decrypted due to misuse of public-key crypto		
Strip Lite - Password Manager	Yes	No	4000x PBKDF2-SHA1 + 1x AES-256
Safe - Password Awesome Password Lite Password Lock Lite	No	No	N/A (see Remarks)
	Remarks: Master password and user passwords are stored unencrypted		
iSecure Lite - Password Manager	No	No	N/A (see Remarks)
	Remarks: Master password and user passwords are stored unencrypted		
Ultimate Password Manager Free	No	No	N/A (see Remarks)
	Remarks: Master password and user passwords are stored unencrypted		
Secret Folder Lite	No	No	N/A (see Remarks)
	Remarks: Master password and user passwords are stored unencrypted		
SafeWallet - Password Manager	Yes	No	10x PBKDF2-SHA1 + 1x AES-256
SplashID Safe for iPhone	Yes	No	N/A (see Remarks)
	Remarks: Hard-coded key is used to encrypt master password		
DataVault Password Manager	Yes	Yes	1x SHA-256 + 1x SHA-1

App name	Encrypts data?	Uses keychain?	Password verification complexity
mSecure - Password Manager	Yes	No	1x SHA-256 + 1x Blowfish
LastPass for Premium Customers	Yes	No	2x SHA-256 + 1x AES-256
1Password Pro	Yes	No	1x MD5 + 1x AES-128
BlackBerry Password Keeper	Yes	N/A	3x PBKDF2-SHA1 + 1x AES-256
BlackBerry Wallet 1.0	Yes	N/A	2x SHA-256
BlackBerry Wallet 1.2	Yes	N/A	1x SHA-512 + 100x PBKDF2-SHA1 + 1x AES-256
	Remarks: Variable (50-100) number of PBKDF2-SHA1 iterations		

Table 2. Password recovery speeds and recoverable password lengths.

Name	Password verification complexity	Password rate, passwords/ sec (est.)		Password length
		CPU	GPU	
Keeper® Password & Data Vault	1x MD5	60 M	6000 M	14.7
Password Safe - iPassSafe free version	1x AES-256	20 M	N/A	12.2
Strip Lite - Password Manager	4000x PBKDF2-SHA1 + 1x AES-256	5000	160 K	10.1
SafeWallet - Password Manager	10x PBKDF2-SHA1 + 1x AES-256	1500 K	20 M (AES-bound)	12.2
DataVault Password Manager	1x SHA-256 + 1x SHA-1	7 M	500 M	13.6
mSecure - Password Manager	1x SHA-256 + 1x Blowfish	300 K	N/A	10.4
LastPass for Premium Customers	2x SHA-256 + 1x AES-256	5 M	20 M (AES-bound)	12.2
1Password Pro	1x MD5 + 1x AES-128	15 M	20 M	12.2
BlackBerry Password Keeper	3x PBKDF2-SHA1 + 1x AES-256	5 M	20 M (AES-bound)	12.2
BlackBerry Wallet 1.0	2x SHA-256	6 M	300 M	13.4
BlackBerry Wallet 1.2	1x SHA-512 + 100x PBKDF2-SHA1 + 1x AES-256	200K	3200 K	11.4

Password length denotes maximum length of a password composed of random digits that can be recovered in one day. That is, it is equal to:

$$\log_{10} \mathit{MAX}\,(\mathrm{RATE}_{\mathrm{CPU}}, \mathrm{RATE}_{\mathrm{GPU}}) \cdot \mathit{86400} \text{ seconds.}$$

To quickly convert this value to a comparable length of a password composed of random ASCII characters one can simply divide the former number by two (since number of ASCII characters is $95 \approx 10^2$).

To give our reader an idea how this compares to the built-in security mechanisms of iOS and BlackBerry OS suffice it to say that backup encryption in iOS uses 10'000 iterations of PBKDF2-SHA1, and backup encryption in BlackBerry Desktop Software uses twice of that. It is also important to note that user is not challenged (in the case of iOS) with entering backup password every time she backups the device, while frequent need to access the password keeper application might tempt her to use much shorter and simpler password, prone to bruteforce or guessing attacks.

User, however, will be routinely challenged to provide device password (or, in iOS terminology, passcode). This password probably won't be of a great complexity, because it must be entered over and over again. Both iOS and BlackBerry have protections in place to make attacking device password hard. In case of iOS this currently is only possible for devices before iPhone 4S and iPad 2 (unless device is jailbroken) and must be done on the device itself. This means that offline attacks on device password are not possible. The process also is slow, somewhat 5-7 passwords per second. BlackBerry pushes the device password security even further: currently there are no ways to attack or recover device password, unless media card encryption is misconfigured (section 3.2).

On both considered platforms protections offered by OS are far more stronger than provided by analyzed apps. In many cases, if user is not using platform protections (passcode and backup encryption), obtaining passwords stored by password keepers and decrypting them is quite doable.

7 Conclusions

Many password management apps offered on the market do not provide adequate level of security. We strongly encourage users not to rely on their protections but rather use iOS or BlackBerry security features.

For Apple users: set up a passcode, and a (complex!) backup password. Do not plug the unlocked device to computers you do not trust to prevent creation of pairing. If you can't encrypt backup for some reason, restrict access to it as much as possible.

For BlackBerry users: set up a device password. Make sure media card encryption is off or set to "Encrypt using Device Key" or "Encrypt using Device Key and Device Password" to prevent device password recovery based on media card. Do not create unencrypted device backups or restrict access to them as much as possible.

8 References

1. Choosing Good Passwords – A User Guide, http://hitachi-id.com/password-manager/docs/choosing-good-passwords.html
2. iExplorer, http://www.macroplant.com/iexplorer/
3. i-FunBox, http://www.i-funbox.com/
4. BlackBerry support, Cellebrite UFED Physical Analyzer, http://www.cellebrite.com/mobile-forensics-products/forensics-products/ufed-physical-analyzer/blackberry.html
5. "EPPB: Now Recovering BlackBerry Device Passwords", Elcomsoft Blog, http://blog.crackpassword.com/2011/09/recovering-blackberry-device-passwords/
6. Keeper® Password & Data Vault, App Store URL: http://itunes.apple.com/us/app/id287170072
7. Password Safe - iPassSafe free version, App Store URL: http://itunes.apple.com/us/app/id413222686
8. My Eyes Only™ - Secure Password Manager, App Store URL: http://itunes.apple.com/us/app/id285835523
9. Strip Lite - Password Manager, App Store URL: http://itunes.apple.com/us/app/id316822267
10. SQLCipher: Open Source Full Database Encryption for SQLite, http://sqlcipher.net
11. Safe - Password, App Store URL: http://itunes.apple.com/us/app/id482919221
12. Awesome Password Lite, App Store URL: http://itunes.apple.com/us/app/id480924162
13. Password Lock Lite, App Store URL: http://itunes.apple.com/us/app/id478776410
14. iSecure Lite - Password Manager, App Store URL: http://itunes.apple.com/us/app/id492408448
15. Ultimate Password Manager Free, App Store URL: http://itunes.apple.com/app/id426030098
16. Secret Folder Lite, App Store URL: http://itunes.apple.com/app/id471895662
17. SafeWallet - Password Manager, App Store URL: http://itunes.apple.com/app/id311202163
18. SplashID Safe for iPhone, App Store URL: http://itunes.apple.com/app/id284334840
19. DataVault Password Manager, App Store URL: http://itunes.apple.com/app/id323373347
20. mSecure - Password Manager, App Store URL: http://itunes.apple.com/app/id292411902
21. LastPass for Premium Customers, App Store URL: http://itunes.apple.com/app/id324613447
22. 1Password Pro, App Store URL: http://itunes.apple.com/app/id319898689
23. BlackBerry Wallet, http://us.blackberry.com/smartphones/features/browser/wallet.jsp