Critiquing Bitcoin's rendition of usable key management

No Author Given

No Institute Given

Abstract. Bitcoin users are directly or indirectly forced to deal with public key cryptography, which has a number of security and usability challenges that differ from the ubiquitous password-based authentication underlying most online banking services. In short, users must ensure keys are simultaneously: accessible as needed, resistant to digital theft, and resilient to inadvertent loss. We build on the decades of research into the usability of key management in other domains, such as encrypted email, showing that Bitcoin shares many of the same fundamental challenges. At the same time, some challenges are unique to Bitcoin, including the clear financial value at stake and subtle complexities within the Bitcoin protocol like change addresses. We contribute an evaluation framework for comparing a wide-breadth of Bitcoin proposals. To establish usability issues, we conduct a broad expert review of six representative deployed solutions.

K.4.4Electronic CommerceCybercash, digital cash K.6.5Security and ProtectionUnauthorized access Security, Human Factors

Keywords: Key Management, Bitcoin, Usability

1 Introductory Remarks

In all of the excitement surrounding Bitcoin [15], it is easy to forget that the decentralized currency assumes a solution to the longstanding problem of usable public key cryptography for user authentication. Despite decades of research on this issue exact in parallel technologies, such as digitally signed email, we show that Bitcoin technologies for creating, storing, and managing keys stall on many of the same basic issues.

At the same time, the developer-heavy Bitcoin community has been very prolific in producing deployed technologies with a wide variety of approaches to solving the challenges of a key-based system. We therefore argue that studying this suite of technology offers the clearest picture yet of the challenges in usable key management for digital signatures.

Contributions Our contribution is two-fold. First, the breadth of techniques that have been deployed for Bitcoin key management—from air gapped computers

to paper printouts of QR codes to password-derived keys—demands an equally broad benchmark for objectively rating and comparing solutions. To address this, we develop an evaluation framework for Bitcoin key management, based on [2], to enable direct comparison between the various proposed solutions on usability, deployability and security criteria.

The particular focus of our framework regards usability issues. Evaluating any set of competing approaches is difficult to do in a thorough, objective way without employing a formal usability evaluation. While user studies are the gold standard for this purpose, user studies have been traditionally employed to rigorously compare a small number of systems according to a narrow set of measurable properties. By contrast, we are interested in revealing broader mental model issues across a wider spectrum of tools. Expert review enables the breadth of evaluation we require, and we utilize the cognitive walkthrough methodology to maintain rigour and objectively in the evaluation.

Scope The focus of this work is configuring and managing the cryptographic keys required to hold Bitcoin. This paper is *not* a study of the usability of Bitcoin itself—*e.g.,* sending and receiving transactions with various software tools. There is no doubt that such a study would be a valuable contribution to the literature, however we believe key management is the harder, fundamental problem that ultimately cannot be addressed with better user interfaces or dialogues. Without a secure, usable path-forward, we expect Bitcoin adoption among non-experts to stall.

2 Preliminaries

2.1 Bitcoin

Bitcoin is a cryptographic currency deployed in 2009 [15]. It has reached a level of adoption unrealized by decades of previously proposed digital currencies (from 1982 [4] onward). Unlike many previous proposals, Bitcoin does not distribute digital monetary units to users. Instead, a public ledger is maintained that contains every transaction, where a transaction in its most common form[1] moves some balance of the Bitcoin currency (XBT or BTC) from one or more accounts (called input addresses) into one or more accounts (called output addresses). Bitcoin addresses are indexed by the fingerprint of a public key from a digital signature scheme (ECDSA). They are not registered in any way—they become active when the first transaction moving money into them is added to the ledger.

Every transaction must be signed by the private signing key associated with each input address in the transaction. In order to spend XBT, users require access to the signing key of the account holding their Bitcoin. Thus users do not maintain any kind of units of currency; they maintain a set of keys that provide them signing authority over certain accounts recorded in the ledger.

[1] Technically, a transaction specifies a short script that encodes how the balance can be claimed as the input to some future transaction.

The ledger (called the blockchain) is maintained and updated by a decentralized network using a novel method to reach consensus that involves incentivizing nodes in the network with the ability to mine new XBT and collect transaction fees. The details of the Bitcoin consensus model are not relevant to this paper.

One subtly of Bitcoin transactions is that each XBT amount in the set of inputs must reference past transactions where the address received adequate XBT to cover the input amount. The complexity is that the transaction must completely spend the amount received in these past transactions even if it is larger than the amount in the current transaction. To enable this, transactions will output the surplus XBT back to the sender as 'change.' Change can be sent back to the same input address, or to enhance privacy, it can be sent to a brand new address created by the sender's client (called 'change addresses').

2.2 Usability of Key Management

Passwords remain the most common form of user authentication. By contrast, private keys are rarely used by non-expert users and are nearly never the default configuration. SSL client-side certificates never reached wide-spread deployment, and recent efforts to reintegrate them in a different form (*e.g.,* origin-bound certificates [8]) still rely on passwords as the primary authentication mechanism. Similarly, while SSH can be used with certificates, passwords are the default mechanism.

Password managers, when configured to set system-chosen random passwords, share at least one property of cryptographic keys: such passwords become something you 'have' instead of 'know.' However if access to such a password is lost, recovery mechanisms exist unlike cryptographic keys.

The use of public keys by non-experts that is closest to Bitcoin is arguably encrypted/authenticated email, in particular PGP and its open-source alternatives (*i.e.,* GPG and OpenPGP). Beginning with 'Why Johnny Can't Encrypt' [18], the usability of public key technology has been well-studied from a usability perspective[2] [11,10,16,12]. The findings of this literature are diverse but relevant observations include the following: the terminology of public and private keys is confusing, it is difficult to correctly obtain other users' public keys, key mitigation between devices is difficult, *etc.*. That said, this literature tends to focus primarily on encryption and not signatures, limiting its applicability.

2.3 Usability Evaluation Methodologies

A number of usability evaluation methodologies employ exert review. We use a cognitive walkthrough [17], which has been used previously to study closely-related subjects: public key technology [18] and software configuration [6]. A cognitive walkthrough is premised on the idea that users learn through exploration of the software, instead of reading manuals. They attempt to perform the task they want completed and rely on the interface to intuitively guide them

[2] "Why King George III can encrypt," *Freedom to Tinker* (blog), 6/6/2014.

through proper design, interface cues, and feedback. In a cognitive walkthrough, we define the set of core tasks that users will be interested in performing, a set of evaluation guidelines that should hold true for the software, and use expert review to evaluate the application against the guidelines while performing the core tasks.

Cognitive walkthroughs are primarily relied on when the breadth of the evaluation makes a user or field study prohibitive to run due to time and cost. We examine six Bitcoin key managers, from configuration through transaction authorization through key recovery. If the results of the cognitive walkthrough narrows the field significantly, user studies are an appropriate follow-up for detailed examination of the most challenging set of tasks within one or two solutions. Thus while our result can be considered a first-pass at the problem, we felt the richness of the result merits sole presentation.

2.4 Consumer Protection

The consequences of losing exclusive control over an account containing monetary value connects the threat of losing a Bitcoin key to losing an online banking password. However, in reality, consumers are legally protected in many countries from any liability of such a loss, and because most bank transactions are traceable and reversible, such accounts are difficult to exact value from (most techniques involve a mule) [9]. Bitcoin transactions are also traceable, however they are not reversible. Stolen Bitcoins cannot be recovered, consumers have no legal protection, and while stolen Bitcoins could be 'tainted' in the blockchain, several mechanisms exist for laundering Bitcoins and similar digital currencies [14,3].

3 Systems Evaluation of Tools

Before turning to a detailed usability evaluation, we evaluate from a systems perspective each category of tool for managing Bitcoin private keys. We highlight security and deployability issues, and note relevant details of the Bitcoin protocol that create complexities and potential discrepancies with users' mental models.

3.1 Keys in Local Storage

On first launch, the official bitcoin client, `bitcoin-qt`, creates a `wallet.dat` file in the Bitcoin data directory (usually a hidden folder inside the user's application folder). The `wallet.dat` file contains the set of all private keys belonging to the user, allowing the user to sign transactions. Anyone with access to the private keys inside `wallet.dat` can spend the XBT associated with those keys.

The `wallet.dat` file can be read by any application with access to the user's application folder. Malware is a particularly noteworthy example here, since theft of the `wallet.dat` file by a malicious developer results in immediate access to the

victim's funds. In 2011, Symantec discovered the *Infostealer.Coinbit*[3] malware, which targeted Windows systems in an attempt to find `wallet.dat` files and sent them via email to the attacker. Since then, numerous examples of malware and malicious Bitcoin tools have been discovered stealing `wallet.dat` files.

Unintentional sharing of the `wallet.dat` file is also a concern. Users must be cautious to not inadvertently share their bitcoin application folder on the Internet or to a location outside of the user's control. Possible sharing includes peer to peer (P2P) file-sharing networks, off-site backups, or shared network drive. Physical theft of the system hosting the `wallet.dat` file is also a concern, especially in the case of portable computers.

By keeping the `wallet.dat` file locally, users must also be wary of *threats to digital preservation* [1] such as general equipment failure due to natural disasters and electrical failures; acts of war; mistaken erasure (*e.g.,* formatting the wrong drive or deleting the wrong folder); bit rot (*i.e.,* undetected storage failure); and possibly others. This must also preserve a specification of the exact format of `wallet.dat` to ensure it can continue to be read.

The user must also be wary of *key churn* as the `bitcoin-qt` client sends change to new addresses. By default, it creates private keys in batches of 100 (called a keypool). This has the unfortunate side-effect that backups become obsolete after the user churns through their current keypool. The user interface of `bitcoin-qt` does not display change addresses or give any indication that they are being used, and so it is quite natural that a novice users' mental model will not account for this behaviour, and they will not act accordingly to ensure they re-backup `wallet.dat` each time they deplete the keypool (another event that is not communicated to the user in any way). To address key churn, alternative Bitcoin clients return all change to the same address or derive all change addresses, called a *deterministic wallet*, from a single key.

Another disadvantage of using `bitcoin-qt` is that it requires a copy of the entire blockchain to validate the balance associated with each of the keys it will create. At the time of writing, the blockchain is 18 GB.[4] For a new installation, it is not uncommon for it to take days to obtain a local copy of blockchain from the Bitcoin peer-to-peer network.

Mobile Wallets A number of mobile wallets for OSes like Android and BlackBerry have been proposed, including Bitcoin-wallet[5] and Mycelium.[6] The main challenge for mobile platforms is the general unfeasibility of obtaining and storing the entire blockchain. Instead, they implement a 'thin client' approach which shortcuts full validation of the blockchain by placing bounded trust in the Bitcoin peer-to-peer network (specifically, it uses a protocol called simplified payment verification or SPV [15]). SPV is also used on some desktop clients. Regarding

[3] `http://www.symantec.com/security_response/writeup.jsp?docid=2011-061615-3651-99`

[4] `https://blockchain.info/charts/blocks-size`

[5] `https://androidobservatory.org/app/9A33CF64239CC26F6E52F80BE83AF39B93F261C0`

[6] `http://mycelium.com`

key management, mobile clients are not much different from `bitcoin-qt` and still store keys in a local file on the device that is subject to the same threats: unauthorized access and non-preservation.

3.2 Password-Protected Wallets

Some Bitcoin clients allow a locally stored wallet file to be encrypted with a key derived (*e.g.,* with PBKDF2) from a user-chosen password. At the time of our analysis, this was possible but not the default in `bitcoin-qt`, however it is enabled by default in Multibit[7], another popular desktop client. Password-protected wallets address certain types of theft, requiring brute-force of the password if the file or a backup of the file is stolen physically or digitally. However, in the case of malware, the addition of keystroke logger would moot this protection.

The trade-off of a password-protected wallet is that users can lose their XBT by forgetting the password protecting their wallet. No recovery mechanism exists (as this mechanism could itself be exploited in the case of theft) short of exhaustive search, which is an available service.[8]

Password-protect wallets may also mislead the user to believe that the password provides access to their funds, as would be congruent with their mental model of online banking. Users may be surprised that they cannot type in their password on a new device and access their XBT.

3.3 Offline Storage

To further enhance theft-protection from malware-based threats, wallets can be stored offline on some form of portable media, such as a USB thumbdrive. This enables the use of traditional physical security to protect the media, which users may have a better mental model of. However offline storage has the drawback of leaving the wallet not immediately accessible for use. Offline storage can be used for backup as well, however unless if all copies of the wallet are offline, the theft-protection benefits are not realized.

A special case of offline storage are paper wallets,[9] where the private keys are printed onto paper typically in the form of a QR code. Securing a paper wallet becomes similar to securing cash, a mental model novice users should be comfortable with. However there are differences. First, funds can be stolen from a paper wallet by being able to observe the QR code (*e.g.,* on live television[10]), which is not possible with physical money. Thus transporting a paper wallet securely requires the QR code to be hidden. Second, users must preserve software capable of decoding the QR code as the paper wallet generation service (*e.g.,* a website) may be unavailable when reloading the funds into a device. Finally,

[7] `https://www.multibit.org`
[8] `http://www.walletrecoveryservices.com`
[9] `https://bitcoinpaperwallet.com`
[10] "A Bloomberg TV Host Gifted Bitcoin On Air And It Immediately Got Stolen," *Business Insider*, 10/23/2013.

users must recognize a paper wallet does not contain XBT itself, but rather signing authority over an account. For example, if a paper wallet is discarded after the XBT are spent, the paper wallet still provides access to any future funds that may be added to the account.[11] Finally, users still need to be cautious of key churn and that spending XBT from a paper wallet does not result in XBT being sent to a change address not included in the paper wallet.

3.4 Air Gapped Storage

In offline storage, we assume the device or media holding the wallet cannot perform computations, such as signing transactions. We distinguish this from air gap storage where wallets are stored offline on a device that generates, signs, and exports transactions onto some portable media, which can be imported and transmitted to the Bitcoin network by a secondary online device. A notable client supporting this setup is Armory.[12]

An air gap also enhances theft-resistance in different ways than offline storage. It offers stronger protection against digital theft by never directly exposing the key to an internet-connect device. By contrast, air gap devices are capable of actually executing malware if infected which may attempt to jump the air gap through a variety of methods observed in the wild.

While not literally an air gap, hardware security modules (HSMs) emulate the properties of an air gap by isolating the key material from the host device, and only exposing the ability to sign transactions. Bitcoin-specific HSMs are under active development at the time of writing.

Note that the consequences of obtaining unauthorized access to the wallet itself is not much different from accessing a transaction-signing oracle for the wallet—both allow the current balance of XBT to be stolen (however future funds may be protected if access to the signing oracle is non-persistent).

3.5 Password-Derived Keys

All the tools analyzed to this point have required users to maintain cryptographic keys. The remaining two solutions enable users to access their XBT with a password instead. The first approach is to derive all cryptographic keys from a password—*e.g.,* form a seed from PBKDF2 applied to the password and expand the seed into the required number of keys with a PRG. Thus it is a special case of a deterministic wallet, where the initial seed is password-derived instead of randomly chosen. A notable tool here is Brainwallet.[13]

Password-derived wallets are targeted at loss-prevention and simpler cross-device access. The challenges of preserving access to a digital file are no longer necessary as the wallet can be regenerated from a memorized password, and this

[11] "Five Ways to Lose Money with Bitcoin Change Addresses," *Bitzuma* (Blog), 17/03/2014.

[12] `https://bitcoinarmory.com`

[13] `http://brainwallet.org/`

wallet can be regenerated on any device. The primary drawback of a password-derived wallet is that weak passwords can be found through unthrottled exhaustive search as a fingerprint of the associated public key will be in the ledger if the account holds any amount of XBT. Rainbow tables for Brainwallet password-derived keys have been reportedly developed. Finally, it is not clear that memorization poses an advantage over maintaining a digital file at loss prevention—a forgotten password will orphan all funds in the account.

3.6 Hosted Wallets

A final approach to key management is to host your accounts with a third party webservice; known as a *hosted wallet*. In this case, the webservice maintains possession of the XBT. It provides the user with access to transactional functionalities through standard web authentication mechanisms, such as a password or two-factor authentication, as well as password recovery mechanisms. Exchange services that allow XBT to be bought and sold for standard currency effectively host a wallet, in addition to webservices deployed specifically to serve this purpose, e.g `Coinbase.com`.

The popularity of hosted wallets is understandable as they provide the closest experience to traditional online banking, however their use has also been hampered by high profile breeches and fraud. Users' funds have been unrecovered from services such as Mt.Gox and Bitcoinica, while popular exchanges such as BTC-E have suffered losses but fully reimbursed users. Thefts and losses from/by third party services are catalogued online[14] and include over 40 events involving losses greater than 1000 XBT.[15]

As one counter-measure to theft, wallet hosts often keep only a small float of their holdings online (called *hot storage*) and store the majority of their holdings offline in *cold storage*. This has the drawback of causing delays if the float is exhausted, and is still susceptible if hot storage theft goes undetected as the float will be persistently replenished. Services may also allow audits, where they prove possession of sufficient XBT to match their liabilities, to demonstrate that they are not covering up any missing XBT (which would not necessarily be immediately apparent—*cf.* a fractional reserve).

Another approach that falls under Hosted Wallet categories is Hybrid hosted wallets. Other than server side encryption and security measures, It uses client side encryption (javascript) to encrypt all the private keys and sensitive data with user's password and sends the encrypted data as a random base64 string to the server. With this implementation, there is no access to the private keys and the final balance from anyone whom have access to the server's data [16]. Blockchain.info uses this implementation for its hosted wallet.

---

[14] https://bitcointalk.org/index.php?topic=576337.0

[15] At the time of writing, 1000 XBT ¿ 650 000 USD.

[16] `http://bitcoin.stackexchange.com/questions/5249/how-secure-is-blockchain-info`

4 Usability Evaluation of Tools

In this section, we evaluate the usability of the following tools (with default configurations, unless otherwise stated) on OS X:

1 Key in Local Storage: Bitcoin-Qt
2 Password Protected: Multibit
3 Air gap: Bitcoin Armory
4 Offline Storage: Bitaddress
5 Password-derived Key: Brainwallet
6 Hosted Wallet: Blockchain.info wallet

We evaluate the tools with the following four core tasks:

T1 Configure a new Bitcoin address and obtain its balance.
T2 Authorize a transaction.
T3 Authorize a transaction the same address on a secondary device.
T4 Recover from the loss of the main credential.

The literature contains a variety of evaluation guidelines or heuristics to use for expert review of security-related tools. We chose to use the guidelines proposed in a usability evaluation of configuring the anonymity software Tor [6] since we share a focus on configuration. This set is itself culled from numerous sources in the literature [17,18,7,13,5].

The set of guidelines, from [6], are:

G1 Users should be aware of the steps they have to perform to complete a core task.
G2 Users should be able to determine how to perform these steps.
G3 Users should know when they have successfully completed a core task.
G4 Users should be able to recognize, diagnose, and recover from non-critical errors.
G5 Users should not make dangerous errors from which they cannot recover.
G6 Users should be comfortable with the terminology used in any interface dialogues or documentation.
G7 Users should be sufficiently comfortable with the interface to continue using it.
G8 Users should be aware of the application's status at all times.

4.1 Keys in Local Storage

We begin with an evaluation of `bitcoin-qt`, the original Bitcoin wallet client, which uses locally-stored keys. We assume the user has downloaded and installed the `bitcoin-qt` client (it has a straight forward wizard installation procedure).

T1: Configuration On the first run, the user will see an 'Overview' page that indicates 'Out of Sync' in red. The first part of the task, to obtain a Bitcoin address, is actually completed although not indicated (G3). She can find the address by clicking on the 'Receive Coins' tab of the application, however this could be easily confused with the 'Addresses' tab which contains a contact list of other addresses (G2).

In order to obtain the balance of the account, the application must be connected to internet to obtain the blockchain. Except for a small status indicator on the bottom-right side of the window that shows a small red cross in-between two black windows, there are no other alerts (G1). With a mouse over the icon, it says '0 active connections to bitcoin network' which is likely unfamiliar language that does not help resolve the error (G6, G4 and G8). With an active internet connection, a status bar at the bottom will complete synchronization with the blockchain, a process that may take days due to the large size of the blockchain.

T2: Authorization Authorizing a transaction is straightforward as the required keys are already loaded into the application from `wallet.dat`. The user can use the 'Send' tab to form a transaction. Since our focus is on key management, we do not evaluate the actual completion of transactions. We focus on ensuring the key is available to the software tool (which is not so straightforward with *e.g.,* offline storage).

T3: Authorization from Secondary Device By installing `bitcoin-qt` on a secondary device, a new wallet file is generated. In order to complete the task, the user may not understand that something must be copied from the first installation (G1) and if so, what exact file it is (G2). The correct procedure is to back up the `wallet.dat` with the 'backup wallet...' option in the 'File' tab of the first installation and chose a directory to save the `wallet.dat`. Next the user must securely transfer this file to the secondary device, and no guidance is provided on how to do this (G2) or the dangers of transferring it through an insecure mechanism (G5).

Assuming the user has transferred `wallet.dat` to the secondary device, she could try looking for import options in the newly installed wallet client, or drag and drop the `wallet.dat` into the client, but she would fail to do so as no import option exists (G2). The documentation is inadequate here as well—there is actually nothing in the help menu except a debug window that is for advance user to tweak the application (G2 and G6)!

The only mechanism to activate the wallet on a secondary file is to actually overwrite `wallet.dat` on the secondary device with `wallet.dat` from the first. It is unlikely any novice user would be able to complete this step. It is actually even difficult to find the path to copy `wallet.dat` to on the new device—this could be possible by searching the local file system for `wallet.dat`, which might not succeed due to non-searchable system reserved folders or not knowing the exact file name (spotlight does not return any result for `wallet.dat`). More likely, the user

will search online.[17] On OS X, the path is `/Users/User/Library/Application Support/Bitcoin/wallet.dat`.

The next step is to replace the new `wallet.dat` with the one from the primary device. It should be noted that the name of the file should be exactly `wallet.dat` for the `bitcoin-qt` to be able to read the file. Some of errors that the user might encounter during this procedure are:

- The user might copy `wallet.dat` from the primary device wallet client path instead of the one exported through the back up option. This could cause a corrupted `wallet.dat` that is not readable by the secondary device's `bitcoin-qt`. This is due to `bitcoin-qt`'s procedure to lock `wallet.dat` while it is in use. The error is recoverable by repeating the process correctly (G4).
- On the secondary device, the final balance might be wrong and there would be the need to resynchronize and rescan the blockchain to have the correct final balance (G3).

Finally, this process must be repeated if either client exhausts their keypool. If both do, there is no way to merge the new keys in the keypool, and replacing one `wallet.dat` with the other will lead to unrecoverable funds (G5).

T4: Recovery If only one device is used, there is no way to recover from loss of the `wallet.dat`—*e.g.,* due to a memory failure, file corruption, or loss of the device itself (G5). If the user backed-up `wallet.dat` using the mechanism described in core task 3, it would only be a meaningful backup if the file were moved to a different device. The process for recovering from a backup is the same as configuring a new device as described in core task 3 and shown to violate most of the usability guidelines.

4.2 Password Protected Wallets

Although it is possible to encrypt the `wallet.dat` with a password in `bitcoin-qt`, it is not the default option nor is there any cue to do so. Instead we evaluate the `MultiBit` client, where one of the recommended first steps is to password protect the wallet file. `MultiBit` is a popular client in particular for its use of SPV for lightweight blockchain validation that can complete within minutes instead of, relative to `bitcoin-qt`, days.

T1: Configuration On the first run, a welcome page contains an explanation of common tasks that could be done with `MultiBit`—where the send, request and transaction tabs are and, importantly, how to password protect the wallet file (good evaluation on G1 and G2). It provides help options for other functionalities with direct and non-technical guides (good G6).

After opening the application, a new receiving address is immediately created. The interface shows the status of the program (online, offline, or out of

[17] `https://en.bitcoin.it/wiki/Data_directory`

sync) on the bottom left status bar, the balance of the user's wallet on the upper left, and the latest price of bitcoin on the upper right of the window (G8). The interface seems to minimize jargon and technical vocabulary (G6). As it is mentioned on the 'welcome page' of the application, every option in `MultiBit` has the ability to show help tips by hovering the mouse over that option (G6 and G7).

The displayed balance is not necessarily current until synchronization is completed, however there is no direct cue in the balance area indicating this (G8). By navigating to a 'request' tab along the top of the application, the user can view her address, as well as copy or drag a QR code of it. Within a frame titled 'Your receiving addresses' all the addresses stored in the wallet file is shown and the user may click on the 'new' button to generate additional addresses.

T2: Authorization To authorize a transaction the user navigates to the tab labeled 'send.' If the client is not synced, the send button is disabled (G4). If it is synced, the user fills out the transaction details, destination address and amount and clicks send (recall we are not evaluating these details, only the authorization step). After clicking send, the client cannot immediately authorize the transaction since the wallet is password-protected. The client prompts the user for her password (G2). An incorrect password displays the error 'The wallet password is incorrect' but otherwise allows immediate and unlimited additional attempts. Entering the correct password authorizes the transaction (G3). Subsequent transactions require reentering the password.

T3: Authorization from Secondary Device On the primary device, the user has to look under 'options' to find the backup options, where it is under 'Tools - Export Private Keys' which contains jargon (G6). The interface will display details about current wallet file, the path for the export file to be saved and also a password for the exported file that is enabled by default. If the user attempts to save the exported file without password, a warning is displayed in red: 'Anyone who can read your export file can spend your bitcoin' (G5). By having a password-protected file exported, the user can securely copy the file to the secondary device with some protection against interception. After clicking to export, wallet file is saved in the given path and the client checks that the file is readable (G4 and G5).

On the secondary device, the user installs a fresh copy of `MultiBit` and looks for the import option that is under 'Tools - Import Private Keys.' The window looks the same as the export window. The user browses to her copy of the exported wallet file, enters the password protecting the wallet, and clicks on the 'Import Private Keys' button. The client confirms the completion of the import (G4) and changes the balance according to the balance of the new imported addresses (G8). The user is now able to authorize transactions from the secondary device. `MultiBit` sends change to the originating address, so keypool churn is not an issue.

T4: Recovery As with `bitcoin-qt`, recovery is not possible if no backup of the wallet file was made. Creating a backup and importing it follows the same procedure as core task 3. With a password-protected wallet, it is also necessary to remember the password that protects the wallet. Without both a backup of the wallet and the password (short of exhaustive search of the password space), recovery is not possible (G4). The welcome page could emphasize the importance of a backup.

4.3 Air Gapped Storage

Bitcoin Armory is an advanced bitcoin wallet that allows the wallet to be stored and managed on an offline device, while supporting the execution of a transaction through an online device. `Armory` is also used on the online device to obtain the blockchain and broadcast the transaction created on the offline device. It is possible to use some other online application to implement the airgap, however this is the recommend method and the one we will consider.

T1: Configuration The user begins by installing Bitcoin Armory on the offline computer. On the start, the welcome page offers the option to 'Import Existing Wallet' and 'Create Your First Wallet!' (G2). The user creates her wallet, with passphrase-protection being a mandatory option. Armory asks to verify the passphrase and warns the user not to forget her passphrase (G5). After this step, a backup window pops up with the options to print a paper wallet or save a digital backup of the wallet, and also warns the user if he decides not to backup his wallet (G5). After proceeding, the user must click on 'Receive Bitcoins' to prompt the client to generate the bitcoin address in the wallet file (G3,G4). By contrast, most clients do this step automatically on launch.

In order to see the balance of the account, the device must be online and synced (G2). Thus the user must use the online device, not the offline device, to check her balance. Users can click on the 'Offline Transaction' button, which offers a short documentation of the steps to be taken to sign a transaction and in doing so, explains the offline/online distinction relevant to checking a balance. Within the 'Wallet' window, there is an option to 'Create Watching-Only Copy.' The language is difficult (G6): this option allows a copy of only the bitcoin addresses to be exported, not the private keys, for use on the online computer to display an updated balance for each address. The exported file can be copied to the online computer.

We assume the user has installed `Armory` on the online computer. It should be noted that Armory only works side-by-side with `bitcoin-qt` and uses `bitcoin-qt` to synchronize and read the downloaded blockchain (G1). A pop-up window will alert the user when 'the blockchain' (G6) has been downloaded (G3). `Armory` displays a 'Connected' cue in green in the bottom-right when it connects to `bitcoin-qt`. Upon launching `Armory`, the user should click on 'Import Existing Wallet' and she is prompted to import either a digital backup or watch-only wallet. She should chose the watch-only back up file that has been copied from the

offline computer. After the application is done syncing, the balance is displayed on the main window under 'available wallets' (G8).

T2 & T3: Authorization With an air gap, the distinction between primary and secondary devices is less clear given that the basic setup itself includes two devices: one online and one offline. Our motivation with task 3 is to address the use of secondary devices, such as smartphones and tablets. If used with an air-gap, these are likely to be the online device, while authorization itself happens on the offline device. Thus we collapse the two core tasks into one. The same steps from core task 1 are required on any new online device: adding the watch-only wallet and waiting for it to synchronize with the blockchain to obtain the balance.

To authorize a transaction, the user may begin from `Armory` on the online or offline device (G2). On the either device, the user should click on 'Offline Transactions' in the main window which displays a very detailed description of the steps involved (G1 and G2). On the online computer, the user clicks the option: 'Create New Offline Transaction.' Unless the wallet client is online, this option is disabled (G4). By clicking on this option, user will be asked to enter the transaction details and click 'Continue' to generate an unsigned transaction as a file. The user must transfer this file to the offline computer. As mentioned in this step's documentation, the unsigned transaction data has no private data (the exact data will ultimately be added to the public blockchain) and no harm can be done by an attacker who captures this file (G5) other than learning the transaction is being prepared.

On the offline computer to sign the transaction, the user clicks on 'Offline Transactions' and then 'Sign Offline Transaction' which prompts the user for the unsigned transaction data file. Armory asks the user to review all the transaction information, such as the amount and the receiving addresses (G5). By clicking on a 'sign' button, it will shows the transaction details again in a confirmation window which the user may click through. Next it will show the signed transaction data and user can save it in the file. Text at the top of the window describes the current state of the file (signed) and what must be done (move to online device) to complete the transaction (G1 and G2).

The signed file should be transferred to the online computer and be loaded through the same offline transaction window. Now the difference is that the "sign" button is disabled but the "broadcast" button has been enabled. By clicking on "broadcast" it will once again show the detail of the transaction for the user to review (G5). Upon clicking to 'continue,' the client will broadcast the transaction to the bitcoin network and a prominent 'Bitcoins Sent!' message is displayed in red (G8).

T4: Recovery Like `bitcoin-qt` and `MultiBit`, `Armory` requires a backup to have been made of the wallet from the offline computer (since the online computer does not have the wallet information). Without this backup, recovery is impossible.

`Armory` provides many prompts for the user to back up her wallet keys. At the time of creating the wallet, there are multiple windows and alerts conveying the importance of a back up, with options for digital and paper copies. Even if user decides not to back up her wallet at this stage, she is provided a persistent 'Backup This Wallet' option (G4). In the backup window, there are a number of options to back up: a digital copy, paper copy, and others. By clicking on the 'Make Paper Backup' for example, the paper backup is shown to the user containing a root key that consist of 18 sets of 4-characters words and a QR code. To restore the paper wallet backup, on the main page, the user can click on 'Import or Restore Wallet' and select 'Single-sheet Backup' option. She will be prompted to input the Root Key from the paper wallet. The 'Digital Backup' option provides an unencrypted version of the wallet file that can be securely stored on portable media. Recovering from a lost wallet with a digital backup involves selecting 'Import Digital Backup or watch-only wallet' from the 'Import or Restore Wallet' window as explained in core task 1. `Armory` also enables the user to test the backups to ensure there is no error in the backup file (G5) through the same import procedure.

4.4 Offline Storage

There are different methods to use for offline storage of a bitcoin wallet. For our evaluation, we consider the use of a paper wallet. Specifically we use the `Bitaddress` web-service, a popular Bitcoin paper wallet generator. Many paper wallet generators exist, however `Bitaddress`, at the time of writing, is the first search result for 'bitcoin paper wallet' on Google.

T1: Configuration Upon visiting the site, the user is being asked to move the mouse or enter some random characters in a text box to generate a high-entropy random value to be used as the private key associated with the bitcoin address. A randomness meter increments and then, when sufficient entropy has been generated, the site redirects to a page that shows the receiving bitcoin address and it's associated private key. Both have a QR code and a base 56 string encoding. The public key (Bitcoin address) is labeled 'SHARE' in green text and the private key 'SECRET' in red. An option to 'Print' prints and finalizes the wallet. These steps are accompanied by concise documentation of the steps the user has to (G2). To ensure the webservice does not retain a copy of the users' key (the generation is done client-side in Javascript), the user should complete the process offline (G2).

After printing, the user has a bitcoin receiving address and, as mentioned in the documentation, the balance can be checked through a webservice that provides access to the blockchain, such as `blockchain.info`. The user uses this site to search for her bitcoin address and check her balance. Although it is documented that the private key must remain secret, a user may inadvertently expose the private key by placing the paper wallet where it can be observed or by searching the website for the private key instead of the public key (G5). Gener-

ally, the terminology within the Bitaddress interface uses non-expert language and simple instructions (G6).

T2 & T3: Authorization For this technique, as the keys are printed on a paper, there is no difference between authorizing from a primary or secondary device so we collapse the analysis of core tasks 2 and 3.

To send funds from a bitcoin address that has been stored on a paper wallet, as it is mentioned in the documentation, the user has to import her private key in one of the wallet clients available, such as Armory or the Blockchain.info hosted wallet discussed below. If the user inputs the private key address into a client that returns change to newly generated addresses, she must export these new addresses to a new paper wallet or she will lose the surplus when she removes the wallet from the client (G5). If the user fails to remove the wallet from the client, a second copy is maintained increasing her exposure to theft (but reducing her exposure to key loss).

The process to import a key from a paper wallet depends on the client. For Blockchain.info, after making an online account, the user navigates to the 'Import/Export' tab and uses the option 'Import Using Paper Wallet, Use your Webcam to scan a QR code from a paper wallet.' It is also possible to type in the private key in the 'Import Private Key' text field. After this step, the address now is hosted on the online wallet and is the same as core task 2 in the Hosted Wallet section (Section 4.6) below.

T4: Recovery Loss of a paper wallet makes the funds unrecoverable (G5). Bitaddress prompts the user to acknowledge this fact (also mentioned in its short documentation) when creating a paper wallet (G1).

4.5 Password-Derived Keys

The most popular and complete implementation of a deterministic wallet with password-derived keys, at the time of writing, is Brainwallet.

T1: Configuration On navigating to the Brainwallet website, there is a pre-generated address with the passphrase 'correct horse battery staple.' This default value should obviously not be used, yet a number of uses have been observed on the blockchain, a violation of G5. The user should replace the default passphrase with her own, hopefully ensuring her passphrase is not a commonly used phrase or anything that could be brute forced by an offline dictionary attack [18] as this passphrase is sufficient to access the funds stored in the resulting bitcoin address. On entering the desired passphrase, the public and private keys are displayed on the same page. Now the user has her own bitcoin address, and to check her balance, she uses a block explorer service, as per paper wallets, and searches for his bitcoin address. This step is not mentioned at all by Brainwallet, in violation of G1 and G2. In addition, the interface is confusing with a number of superfluous fields not likely meaningful for new users (G6 and G7).

[18] "Bitcoin Brainwallets and why they are a bad idea," http://insecurety.net/?p=866

T2 & T3: Authorization In any device (primary or secondary) with a browser, the user can navigate to the `Brainwallet` site and enter her passphrase to recover her keys. To then authorize a transaction, user should import the private keys into another wallet client to be able to spend the funds using the same procedure as described above for paper wallets, with the same usability concerns.

T4: Recovery If the user forgets her passphrase, her funds cannot be recovered (G5). As long as the user recalls her passphrase and the `Brainwallet` site is available, her keys can be recovered. However if the site is ever offline or stops its service, the algorithm used to derive the keys from the passphrase may no longer be available (G5).

4.6 Hosted Wallets

A variety of online services offer online hosted wallet clients to users. We use the popular `Blockchain.info` webservice for our evaluation.

T1: Configuration Th user navigates to the `Blockchain.info` site and starts a new wallet by providing her email address and a minimum of 10 character long password. There is a warning on the bottom of the form that on password loss the account is unrecoverable (G5), however the first pop-up window titled 'Wallet Recovery Mnemonic' would show a phrase that can be used to recover the account in case the password is forgotten. After this step user can login to the site and view her bitcoin address and the balance it holds.

T2 & T3: Authorization One of the advantages of using hosted wallets is that it is accessible from any browser on any device. The user can visit the site, log in, and have access to her funds. To perform a transaction, the user would click on the 'Send Money' tab of the main page, and use a straight-forward interface to form the transaction. We note that in the case of some errors, `Blockchain.info` uses jargon that is likely to be unfamiliar to novice users. As one example, if the transaction has insufficient funds, a cryptic error is displayed: "No free outputs to spend" (G4 and G6).

T4: Recovery There are two methods of recovering an account if the user's password is forgotten—one is to use the phrase given on account registration, and the other is to use a backup. For the wallet recovery mnemonic, on the login page, there are different options for anything that is forgotten (identifier or alias for username and/or password). By clicking on the 'Recover Wallet' option, the site will ask for the mnemonic phrase and the email address or mobile phone number associated with the account to send the new confirmation details. The second recovery option is to make backups and import them in case recovery is needed. To do so, in the main wallet page, user has to click on the 'Import/Export' option and 'Export' either an encrypted or unencrypted backup. Unencrypted backups should be kept in a secure storage. There are different options for the unencrypted backup procedure that could confuse the

user and might result in an unrecoverable backup (G5 and G6)—the back up is shown on a text field that the user has to copy and paste into a text file to be able to save it on her computer (G2, G3 and G7). To restore the backups, there is an 'Import Wallet' option.

Category	*Example*	Malware Resistant (5.1)	Key Kept Offline (5.1)	No Trusted Third Party (5.1)	Resistant to Physical Theft (5.1)	Resistant to Physical Observation (5.1)	Resilient to Physical Observation (5.1)	Resilient to Password Loss (5.1)	Immediate Access to Key Churn (5.1)	No New User Software (5.1)	Cross-Device Portability (5.1)
Keys in Local Storage	`bitcoin-qt`		●		●	●	●	●			
Password-Protected Wallets	`MultiBit`		○	●	○	●		●	●		
Offline Storage	`Bitaddress`	○	●	●			●			●	
Air Gapped Storage	`Armory`	○	●	●		●	●	●			
Password-Derived Keys	`Brainwallet`		●	●	○			●	●	●	●
Hosted Wallet (hot)	`Coinbase.com`						●	●	●	●	●
Hosted Wallet (cold)		○	●				●	●		●	●
Hosted Wallet (Hybrid)	`Blockchain.info`		○	○			●	●	●	●	●
Cash		●	●	●		●	●	●	●	●	●
Online Banking							●	●	●	●	●

Table 1. A Comparison of Key Management Techniques for Bitcoin (and Contrasted with Traditional Financial Services).

5 Evaluation Framework

In this paper, we evaluated different approaches to key management in Bitcoin from both a systems perspective and a usability persecutive. On the usability side, in particular, some findings are very specific to the exact design of the software evaluated and its interfaces. Other usability issues are more fundamental to the category of key management the particular tool falls under. In this section, we systemize the major category-wide issues we have uncovered in the previous sections through an evaluation framework summarized in Table 1. This framework both summarizes the advantages and disadvantages of the various approaches we have evaluated, while also providing a benchmark for evaluating

future proposals. The framework is based on a similar framework for evaluating password replacement schemes [2].

5.1 Evaluation Criteria

We briefly enumerate the criteria used to evaluate each proposal in the framework below.

Malware Resistant Malware designed to steal XBT wallets and related passwords has been observed in the wild. Wallets that are not stored on an online/computational devices are considered malware resistant (•), unless forming a transaction involves transferring to a computational device (○).

Key Stored Offline For archival storage of infrequently used keys, keys not directly accessible from an internet-connect device—either due to being offline (•) or online but password-protected (○)—are preferable. (For traditional banking online, we interpret this benefit as authority over the account being kept offline, which it is not).

No Trusted Third Party All Bitcoin key management tools are trusted to a certain extent. Even an open source Bitcoin client might contain an undiscovered backdoor that kleptographicly steals high value private keys. Here we consider the absence of a persistent trusted third party (•) that maintains direct signing authority over a users' XBT.

Resistant to Physical Theft If the cryptographic keys are stored on some media or device that can be physically stolen, we do not consider the tool to be resistant to physical theft. Within our framework, the only tools meeting this requirement rely on a human memorized password being necessary for key recovery. These are awarded ○since passwords tend to be weak and may not resist the kind of unthrottled guessing possible against a stolen password-protected wallet.

Resistant to Physical Observation Physical observation, such as observing key strokes or capturing QR codes with a camera, may result in access to a user's Bitcoin account.

Resilient to Password Loss If passwords are used (• if not), the loss of a password could result in XBT becoming unrecoverable if it is a necessary authentication factor in obtaining access to the signing key. For solutions where funds are held by third parties, these entities could provide a password recovery/reset mechanism (•).

Resilient to Key Churn Assuming the client sends change from transactions to a newly created change addresses, a tool is resilient to key churn if it can maintain access to the funds even after exhausting the initial keypool (•). Tools not awarded this benefit are not guaranteed to maintain persistent access to new change addresses, and XBT sent to these addresses may be lost.

Immediate Access Tools that maintain direct access to the wallet enable XBT to be transacted, effectively, immediately (•). We include tools that require a user to enter a password. We omit the benefit for tools that require data to be obtained from external storage medium or secondary device.

No New User Software Some approaches require users to install new software on their system, for which the user may not have suitable permission, or software may not be developed for their specific platform (*e.g.,* some mobile platforms). By contrast, some tools can be run from common software, like any standards-compliant web browser (•).

Cross-Device Portablity A tool is cross-device portable (•) if it allows easy sharing of the a Bitcoin address across multiple devices with minimal configuration or usability issues due to complexities like key churn.

5.2 Discussion

From Table 1, no one tool provides a clearly beneficial set of properties over another. We also note that the approaches are not necessarily mutually exclusive. One tool may be used for a small float of XBT, while longer term savings may utilize a different mechanism. The hosted wallet solution provides essentially the same set of properties as online banking—what it lacks is regulatory maturity, consumer protection, and established businesses with track records that earn the trust (encapsulated by the trusted third party criteria) instilled in them. Of all the solutions, further development in hosted wallets seems the most promising for further Bitcoin adoption, primarily among novice users.

6 Concluding Remarks

Our detailed security and usability evaluation of six competing tools for key management in Bitcoin essentially demonstrates that no silver bullet solution has been deployed. Addressing the documented usability issues, fundamental to Bitcoin's design due to its reliance on cryptographic keys, is an important first step toward wider adoption of Bitcoin or similar cryptocurrencies.

While usability is often pitched with the novice user in mind, the issue of key management can be equally troubling for expert users. As a thought experiment, imagine how you would protect your Bitcoin wallet were it to contain XBT worth the equivalent of millions of dollars. With a complete lack of consumer protection,

the reality that a loss of the associated private key will orphan the funds, and the threat of malware and targeted attacks tailored to Bitcoin theft, it is doubtful that anyone would prefer the dubious task of maintaining a Bitcoin wallet over depositing the funds into an insured bank account.

References

1. Baker, M., Keeton, K., and Martin, S. Why traditional storage systems don't help us save stuff forever. In *HotDep* (2005).
2. Bonneau, J., Herley, C., van Oorschot, P. C., and Stajano, F. The quest to replace passwords: a framework for comparative evaluation of web authentication schemes. In *IEEE Symposium on Security and Privacy* (2012).
3. Bonneau, J., Narayanan, A., Miller, A., Clark, J., Kroll, J. A., and Felten, E. W. Mixcoin: Anonymity for bitcoin with accountable mixes. In *Financial Cryptography* (2014).
4. Chaum, D. Blind signatures for untraceable payments. In *CRYPTO* (1982).
5. Chiasson, S., van Oorschot, P. C., and Biddle, R. A usability study and critique of two password managers. In *USENIX Security* (2006).
6. Clark, J., van Oorschot, P. C., and Adams, C. Usability of anonymous web browsing: An examination of tor interfaces and deployability. In *SOUPS* (2007).
7. Cranor, L. F. P3p: Making privacy policies more useful. *IEEE Security & Privacy 1*, 6 (2003).
8. Dietz, M., Czeskis, A., Balfanz, D., and Wallach, D. S. Origin-bound certificates: A fresh approach to strong client authentication for the web. In *USENIX Security* (2012).
9. Florencio, D., and Herley, C. Is everything we know about password stealing wrong? *IEEE Security & Privacy 10*, 6 (2012).
10. Garfinkel, S. L., Margrave, D., Schiller, J. I., Nordlander, E., and Miller, R. C. How to make secure email easier to use. In *CHI* (2005).
11. Garfinkel, S. L., and Miller, R. C. Johnny 2: A user test of key continuity management with S/MIME and outlook express. In *SOUPS* (2005).
12. Gaw, S., Felten, E. W., and Fernandez-Kelly, P. Secrecy, flagging, and paranoia: Adoption criteria in encrypted email. In *CHI* (2006).
13. Karat, C., Brodie, C., and Karat., J. Usability design and evaluation for privacy and security solutions. In *Security and Usability*. O'Reilly, 2005.
14. Miers, I., Garman, C., Green, M., and Rubin, A. D. Zerocoin: Anonymous Distributed E-Cash from Bitcoin. In *IEEE Symposium on Security and Privacy* (2013).
15. Nakamoto, S. Bitcoin: A peer-to-peer electionic cash system. Unpublished, 2008.
16. Sheng, S., Broderick, L., Koranda, C. A., and Hyland, J. J. Why Johnny still can't encrypt: Evaluating the usability of email encryption software. In *SOUPS (Poster)* (2006).
17. Wharton, C., Rieman, J., Lewis, C., and Polson, P. The cognitive walkthrough method: a practitioner's guide. In *Usability Inspection*. Wiley & Sons, 1994.
18. Whitten, A., and Tygar, J. D. Why Johnny can't encrypt: a usability evaluation of PGP 5.0. In *USENIX Security* (1999).