Key Management in Bitcoin: Meet the new paradigm, same as the old paradigm

Position Paper

Jeremy Clark
Carleton University
clark@scs.carleton.ca

David Barrera
Carleton University
dbarrera@ccsl.carleton.ca

ABSTRACT

We don't know if Bitcoin millionaires sleep at night, only that they shouldn't. The average user is not prepared to securely manage a large amount of Bitcoins. This is due, in large part, to Bitcoin's use of public key cryptography which forces every one of its users into the unfamiliar territory of key management. We expect this to prevent Bitcoin from gaining traction beyond its enthusiasts. In this paper, we review the state of the art in Bitcoin key management tools, with particular focus on attempts to emulate the functionality of a password-based system within Bitcoin. Each approach presents trade-offs in security, usability and availability that lead us to conclude none is a silver bullet and the area deserves further attention. We thus propose a research agenda of possible future directions in improving key management techniques for Bitcoin users in the short- and medium-term.

Categories and Subject Descriptors

K.4.4 [**Electronic Commerce**]: Cybercash, digital cash; K.6.5 [**Security and Protection**]: Unauthorized access

General Terms

Security, Human Factors

Keywords

Key Management, Bitcoin, Usability

1. INTRODUCTORY REMARKS

- Different paradigm: protecting a key vs protecting a password.
- Thesis: users aren't ready to manage keys.
- State of the art in terms of usability is to somehow convert the key into a password
- security and deployability (availability) problems
- keys=something you have.

Due to the meteoritic rise in Bitcoin's exchange rate, users are coming to the realization that it is no longer a crypto plaything; it is actual money.

We are not arguing that it will hinder adoption, as users will not realize what they are getting into. Rather it will harm traction, since users won't stick around to get fooled twice.

"Fool me once, shame on... shame on you. Fool me... you can't get fooled again." –W

Scope: bitcoin can be split into 3 parts: buying bitcoins (involving exchanges, OTC, etc), holding bitcoins (involving wallets, keys, etc), and spending bitcoins (creating transactions, waiting for verifications, etc). this paper focuses on the middle point.

2. PRELIMINARIES

2.1 Bitcoin Background

2.2 Other PKI-based Systems

- Same paradigm
- SSH - nerds
- PGP - turbo nerds
- SSL - client certs (no one, or spooks)
- SSL - server

3. BITCOIN PRIVATE KEY MANAGEMENT

At the core of Bitcoin's functionality are keypairs. The public key allows users to receive coins and check their wallet balance, and the private key allows users to send coins to other addresses. In this Section, we review the main mechanisms used in the current Bitcoin ecosystem to manage these keys.

3.1 Default Client

On first launch, the official `bitcoin-qt` client 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 (*i.e.,* send coins). Anyone with access to the private keys inside `wallet.dat` can

	Malware Resistant	Key Kept Offline	No Trusted Third Party	Resistant to Physical Theft	Resistant to Physical Observation	Resilient to Equipment Failure	Resilient to Password Loss	Compatible with Change Keys	Immediate Access	No New Software	Portable	Blank	Blank	Blank
Traditional cash	•	•	•		•	•	•	•	•	•	•			
Key in file (default)			•		•		•	•	•	•				
Password-protected		○	•		•			•	•	•				
Air gap	○	•	•		•		•	•						
Offline storage		•	•			•	•				•			
Password-derived key		•	•	•		•			•		•			
Hosted (hot)						•	○	•	•	•	•			
Hosted (cold)		•				•	○	•		•	•			

Table 1: A Comparison of Key Management Techniques for Bitcoin.

spend the coins associated with those keys. Thus, access control on the `wallet.dat` file is extremely important.

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*[1] malware, which targeted Windows systems in an attempt to find `wallet.dat` files and sent them via email to the attacker.

Unintentional sharing of the `wallet.dat` file is also a concern in the default client. 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* [**?**] 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.

Advantages of using the `bitcoin-qt` client in it's default configuration include immediate access to funds, no trust needed in a third party, and no need to recall yet another password. Additionally, the user can spend coins and receive change without the need to perform extra steps (see Section **??**).

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

3.2 Password-protected Wallet

Some Bitcoin clients allow wallets (specifically the private keys in the wallet) to be encrypted with a user-chosen password.

- malware can still mount an offline attack
- mental model: you may think that you're making your password into a key, but it's actually 2 factor

3.3 Offline Storage

- print privkey to paper (paper wallet) (bitbills)
- put wallet onto usb
- mini cog walkthrough. how easy is it to import they back
- hardware wallet (tpm and stuff can be future directions)
- change accounts (*cf.* deterministic wallets: Armory and Electrum)

3.4 Mobile Wallets

- apps, filesystem unavailable, must have export function into shared storage

4. MANAGING WITH A PASSWORD

cf. online banking (cormac herley).

4.1 Password-Derived Key

- entropy reduction
- salt, iteration count
- parsing (spec that was followed) requires a tool. if you lose the tool, you may not be able to recover your keys. standard may not be detailed enough, or not properly implemented.

- change accounts gets messy

4.2 Hosted (Hot Storage)

- trusted party. they can steal if malicious,
- they're good, but don't have good security. external hackers stealing (instawallet, Mybitcoin
- http://www.theverge.com/2013/4/3/4180020/bitcoin-service-instawallet-suspended-indefinitely-after-hack
- http://www.theverge.com/2013/3/8/4080160/hackers-steal-over-12000-of-bitcoins-from-bitinstant
- legitimate shutdown. legal implications (affidavit)
- web service, so any attack (phishing, xss) or DDoS (Mt Gox http://www.theverge.com/2013/4/4/4181726/bitcoin-exchange-mt-gox-says-technical-problems-are-ddos)
- phishing: Coinbase `http://www.theverge.com/2013/4/5/4186808/bitcoin-banker-coinbase-phishing-attacks-user-information-leaked`

4.3 Hosted (Cold Storage)

- Offline wallets aka cold storage. not immediately available, transactions queueing up (coinbase).

5. A RESEARCH AGENDA

- better import/export of keys
- better language and guidance on key management
- how do you know the key is in wallet.dat ?
- How do you know if a file has been read? Specifically, how do you know if your private key file has been read by an app other than your wallet? We have unix `atime` (time of last access), but we can't log what process read the file.
- publishing public keys. people need to keep these around, and they can use to check if they have the right privkey.
- if you have your pubkey and you have the salt, you can avoid rainbow tables, but your pubkeys get longer. (add salt via commitcoin)
- boyen iterated hashing of passwords -> compute amount of iterations based on money, script
- digital death
- sinkhole accounts
- incentives are there because it's money. not like ssh or ssl.
- min password entropy as a function of how much money its protecting: examine in terms of falling cost to brute force (e.g. w/ EC2), vs rising price of bit coins. -> future proof against lowering EC2 costs and raising bitcoin costs
- bitbills, oblivious printing and other physically intuitive media that can behave (sort of) like cash
- theft/loss mitigation, making it easier to split wallets to mitigate loss
- account rollover vs. password changes
- graphical passwords
- security of RNGs. i read that vanitygen (for making vanity bitcoin addresses) uses OpenSSL's RNG and it's secure because it's "used on thousands of websites".
- air gap -> how good is it?

6. RELATED WORK

Blah blah blah.

7. CONCLUDING REMARKS

In this paper, we have ...

8. REFERENCES

[1] Back, A. Hashcash: a denial of service counter-measure, 2002.

[2] Baker, M., Keeton, K., and Martin, S. Why traditional storage systems donâĂŹt help us save stuff forever. In *Proc. 1st IEEE Workshop on Hot Topics in System Dependability* (2005).

[3] Nakamoto, S. Bitcoin: A peer-to-peer electionic cash system. Unpublished, 2008.

[4] Rivest, R. L., and Shamir, A. PayWord and MicroMint: two simple micropayment schemes. In *Security Protocols* (1996).