<table>
<thead>
<tr>
<th>How to Store and Use Bitcoins</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Simple Local Storage</td>
</tr>
<tr>
<td>• Hot and Cold Storage</td>
</tr>
<tr>
<td>• Splitting and Sharing Keys</td>
</tr>
<tr>
<td>• Online Wallets and Exchanges</td>
</tr>
<tr>
<td>• Payment Services</td>
</tr>
<tr>
<td>• Transaction Fees</td>
</tr>
<tr>
<td>• Currency Exchange Markets</td>
</tr>
</tbody>
</table>
Spending Bitcoin

**Q:** I want to spend a Bitcoin, what do I need to know?!

1. Some info from the public blockchain
2. The owner’s secret signing key

So, it’s all about key management!

Instead of

*How to Store and Use Bitcoins*

the title should be

*How to Store and Use Secret Keys*

---

Goals

**Availability:** You can spend your coins.

**Security:** Nobody else can spend your coins.

**Convenience**
Simplest Approach

Store key in a file, on your computer or phone.

Convenience: very convenient!

Availability: just as available as your device!
  device lost/wiped => key lost => coins lost!

Security: just as secure as your device!
  device compromised => key leaked
  => coins stolen!

Wallet Software

Keeps track of your coins.

Provides nice user interface.

Nice trick: use a separate address/key for each coin.
  1. benefits privacy (looks like separate owners)
  2. wallet can do the bookkeeping, user needn’t know
Encoding Addresses

Encode as text string: **base58 notation**

123456789ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz

Encode as **QR code**

---

How to Store and Use Bitcoins

- Simple Local Storage
- **Hot and Cold Storage**
- Splitting and Sharing Keys
- Online Wallets and Exchanges
- Payment Services
- Transaction Fees
- Currency Exchange Markets
Hot Storage vs. Cold Storage

**Hot storage**
- online
- convenient but risky

**Cold storage**
- offline
- archival but safer

---

Hot Storage vs. Cold Storage

**Hot storage**
- online
- hot secret key(s)
- cold address(es)

**Cold storage**
- offline
- cold secret key(s)
- hot address(es)

---

Separate keys
Hot Storage vs. Cold Storage

Hot storage
- online
- hot secret key(s)
- cold address(es)

Cold storage
- offline
- payments

Dealing with Off-line Cold Wallets

Problem:
- Want to use a new address (and key) for each coin sent to cold
- But how can hot wallet learn new addresses if cold wallet is offline?

Awkward solution:
- Generate a big batch of addresses/keys, transfer to hot beforehand

Better solution:
- Hierarchical deterministic wallet
Recall: Regular Key Generation

- generateKeys
  - address
  - private key

Hierarchical Key Generation

- generateKeysHier
  - address gen info
  - i
  - genAddr
  - jth address
- private key gen info
  - genKey
  - jth key
  - i
doesn't leak keys
Implementation using ECDSA

- Recall: $x$ is private key, $g^x$ is public key
- private key generation info ($k$ and $y$ are new):
  \[ k, x, y \]
- $i^{th}$ private key:
  \[ x_i = y + H(k \| i) \]
- address generation info:
  \[ k, g^y \]
- $i^{th}$ public key:
  \[ g^{x_i} = g^{H(k \| i)} \cdot g^y \]
- $i^{th}$ address:
  \[ H(g^{x_i}) \]

Hierarchical Key Generation

Hot Side

Cold Side
How to store Cold Info

1. Info stored in device, device locked in a safe
2. “Brain wallet”
   - encrypt info under passphrase that user remembers
3. Paper wallet
   - print info on paper,
   - lock up the paper
4. In “tamperproof” device
   - device will sign things for you, but won’t divulge keys

How to Store and Use Bitcoins

• Simple Local Storage
• Hot and Cold Storage
• Splitting and Sharing Keys
• Online Wallets and Exchanges
• Payment Services
• Transaction Fees
• Currency Exchange Markets
Secret Sharing

**Idea:** split secret into \( N \) pieces, such that
- given any \( K \) pieces, can reconstruct the secret
- given fewer than \( K \) pieces, don’t learn anything

**Example:** \( N=2, K=2 \)
- \( P = \) a large prime
- \( S = \) secret in \( [0, P) \)
- \( R = \) random in \( [0, P) \)

**Split:**
- \( X_1 = (S+R) \mod P \)
- \( X_2 = (S+2R) \mod P \)

**Reconstruct:**
- \( (2X_1-X_2) \mod P = S \)

**Secret Sharing**

given any two points, can interpolate and find \( S \)

(random slope \( R \))

(0, \( S \))

(1, \( S+R \))

(2, \( S+2R \))

(3, \( S+3R \))

(4, \( S+4R \))

(do arithmetic modulo large prime \( P \))
### Secret Sharing

<table>
<thead>
<tr>
<th>Equation</th>
<th>Random parameters</th>
<th>Points needed to recover S</th>
</tr>
</thead>
<tbody>
<tr>
<td>((S + RX) \mod P)</td>
<td>(R)</td>
<td>2</td>
</tr>
<tr>
<td>((S + R_1X + R_2X^2) \mod P)</td>
<td>(R_1, R_2)</td>
<td>3</td>
</tr>
<tr>
<td>((S + R_1X + R_2X^2 + R_3X^3) \mod P)</td>
<td>(R_1, R_2, R_3)</td>
<td>4</td>
</tr>
</tbody>
</table>

etc.

- support \(K\)-out-of-\(N\) splitting, for any \(K, N\)

---

### The Good: Store shares separately, adversary must compromise several shares to get the key.

### The Bad: To sign, need to bring shares together, and reconstruct the key. This is a vulnerability.

### Solution! MULTI-SIG – Lets you keep shares apart, approve transaction without reconstructing key at any point.
Secret Sharing using MULTI-SIG: Example

Andrew, Bob, Charles, and Edward are co-workers. Their company has lots of Bitcoins.

Each of the four generates a key-pair, puts secret key in a safe, private, offline place.

The company’s cold-stored coins use MULTI-SIG, so that three of the four keys must sign to release a coin.

How to Store and Use Bitcoins

• Simple Local Storage
• Hot and Cold Storage
• Splitting and Sharing Keys
• Online Wallets and Exchanges
• Payment Services
• Transaction Fees
• Currency Exchange Markets
Online Wallet

like a local wallet but “in the cloud”
runs in your browser
site sends code
site stores keys
you log in to access wallet

Online Wallet Tradeoffs

Pros:
• convenient
• nothing to install
• works on multiple devices

Cons:
• security worries
• what if site malicious?
• what if site compromised?
Bank-like Services

You give the bank money (a “deposit”).

Bank promises to pay you back later, on demand.

Bank doesn’t actually keep your money in the back room.
- typically, bank invests the money
- keeps some around to meet withdrawals (“fractional reserve”)

Bitcoin Exchanges

Accept deposits of Bitcoins and fiat currency ($, €, …)

Promise to pay back on demand.

Lets customers:
- Make and receive Bitcoin payments
- Buy/sell Bitcoins for fiat currency
- Typically, match up BTC buyer with BTC seller
What happens when you buy BTC

Suppose my account at Exchange holds $5000 + 3 BTC

I use Exchange to buy 2 BTC for $580 each

Result: my account holds $3840 + 5 BTC

NOTE: No BTC transaction appears on the blockchain!

Only effect: Exchange is making a different promise now.

Exchanges: Pros and Cons

Pros:
- connect BTC economy to fiat currency economy
- easy to transfer value back and forth

Cons:
- risk!
- same kinds of risks as banks
Exchanges and their Risks

In fact...

**Study: 45 percent of Bitcoin exchanges end up closing**

A study of the Bitcoin exchange industry has found that 45 percent of exchanges fail, taking their users' money with them. Those that survive are the ones that handle the most traffic — but they are also the exchanges that suffer the greatest number of cyber attacks.

Computer scientists Tyler Moore (from the Southern Methodist University, Dallas) and Nicolas Christin (of Carnegie Mellon University) found 40 exchanges on the web which offered a service of changing bitcoins into other flat currencies or back again. Of those 40, 30 have gone out of business — 13 closing without warning, and five closing after suffering security attacks. 
Cryptocurrency Technologies

Bank Regulation

For traditional banks, government typically:
- Imposes minimum reserve requirements
  - Must hold some fraction of deposits in reserve
- Regulates behavior, investments
- Insures depositors against losses
- Acts as lender of last resort

Bitcoin is not regulated like this!

Proof-of-Reserve Problem

Bitcoin exchanges can prove a lower bound on fractional reserve by providing:

1. Lower bound for reserves
2. Upper bound for liabilities
Proof of Reserve

Q: How to prove how much reserve you are holding?

1. Publish a valid payment-to-self of claimed amount.
2. Sign challenge string with same private key.

Now the hard part . . .

Proof of Liabilities

Vanilla approach:
Publish list of amounts and usernames of all accounts!

Users can complain if their accounts are missing or amounts are wrong.

Exchange can create fake users, but this only overstates liabilities.

Problem: What about customer privacy?!!
Approach II: Merkle Tree with Subtree Totals

As customer you can verify that:

1. **Root hash pointer** and **root value** are what exchange published.
2. Hash pointers are **consistent** all the way down.
3. **Leaf** contains correct information (customer no. and amount)
4. Each value is **sum of the values of** subtrees beneath it.
5. Neither of values is **negative number**.
Proof of Reserve

Exchange proves that it has at least $X$ amount of reserve currency.

Exchange proves that customers have at most $Y$ amount deposited.

So, reserve fraction $\geq \frac{X}{Y}$

How to Store and Use Bitcoins

- Simple Local Storage
- Hot and Cold Storage
- Splitting and Sharing Keys
- Online Wallets and Exchanges
- Payment Services
- Transaction Fees
- Currency Exchange Markets
Scenario: Merchant accepts BTC

**Customer objectives:**
- to pay in Bitcoin

**Merchant objectives:**
- to receive dollars
- simple deployment
- low risk (tech, security, exchange rate)

---

Generate pay-with-Bitcoin Button

![HTML for payment button](image_url)
**Payment Process**

1. User clicks "Pay with BTC" button with <transID, amount>.
2. Payment service confirms <transID, amount>.
3. Payment interaction.
4. Redirects user.
5. Confirm <transID, amount>.

**End Result**

- **Customer:** pays Bitcoins
- **Merchant:** gets dollars, minus a small percentage
- **Payment service:**
  - gets Bitcoins
  - pays dollars (keeps small percentage)
  - absorbs risk: security, exchange rate
  - needs to exchange Bitcoins for dollars, in volume
How to Store and Use Bitcoins

- Simple Local Storage
- Hot and Cold Storage
- Splitting and Sharing Keys
- Online Wallets and Exchanges
- Payment Services
- **Transaction Fees**
- Currency Exchange Markets

**Transaction Fees**

It costs resources for
- Peers to *relay* your transaction
- Miners to *record* your transaction

*Transaction fee* compensates for (some of) these costs.

Generally, *higher fee* means transaction will be forwarded and recorded *faster*.
**Consensus Fees**

**Current Consensus Fee**

- **No fee** if
  1. tx less than 1000B in size
  2. all outputs are 0.01 BTC or larger; and
  3. priority is large enough

- **Otherwise fee is 0.0001 BTC per 1000B**

\[
\text{Priority} = \frac{\text{sum of inputAge } \times \text{inputValue}}{\text{tx size}}
\]

\[
\text{Approx tx size} : 148 \, N_{\text{inputs}} + 34 \, N_{\text{outputs}} + 10
\]

**Transaction Fee**

Most miners enforce the consensus fee structure.

If you **don’t pay** the consensus fee, your **transaction will take longer** to be recorded.

Miners **prioritize** transactions based on **fees** and the **priority formula**.
How to Store and Use Bitcoins

- Simple Local Storage
- Hot and Cold Storage
- Splitting and Sharing Keys
- Online Wallets and Exchanges
- Payment Services
- Transaction Fees
- Currency Exchange Markets

Markets: Examples
Cryptocurrency Technologies

How to Store and Use Bitcoins

Buy/Sell Bitcoins

Or . . .
Basic Market Dynamics

- Market matches buyer and seller
- Large, liquid market reaches a consensus price
- Price set by supply (of BTC) and demand (for BTC)

Supply of Bitcoins

\[ \text{supply} = \text{coins in circulation} (+ \text{demand deposits}?) \]

- coins in circulation: fixed number, currently about 16.2M

When to include demand deposits? When they can actually be sold in the market.
Demand for Bitcoins

BTC demanded to mediate fiat-currency transactions

1. Alice buys BTC for $
2. Alice sends BTC to Bob
3. Bob sells BTC for $

BTC "out of circulation" during this time

BTC demanded as an investment

if the market thinks demand will go up in future

Simple Model of Transaction-Demand

\[ T = \text{total transaction value mediated via BTC ($ / \text{sec})} \]

\[ D = \text{duration that BTC is needed by a transaction (sec)} \]

\[ S = \text{supply of BTC (not including BTC held as long-term investments)} \]

\[ \frac{S}{D} \quad \text{Bitcoins become available per second} \]

\[ \frac{T}{P} \quad \text{Bitcoins needed per second} \]

Equilibrium:

\[ P = \frac{TD}{S} \]