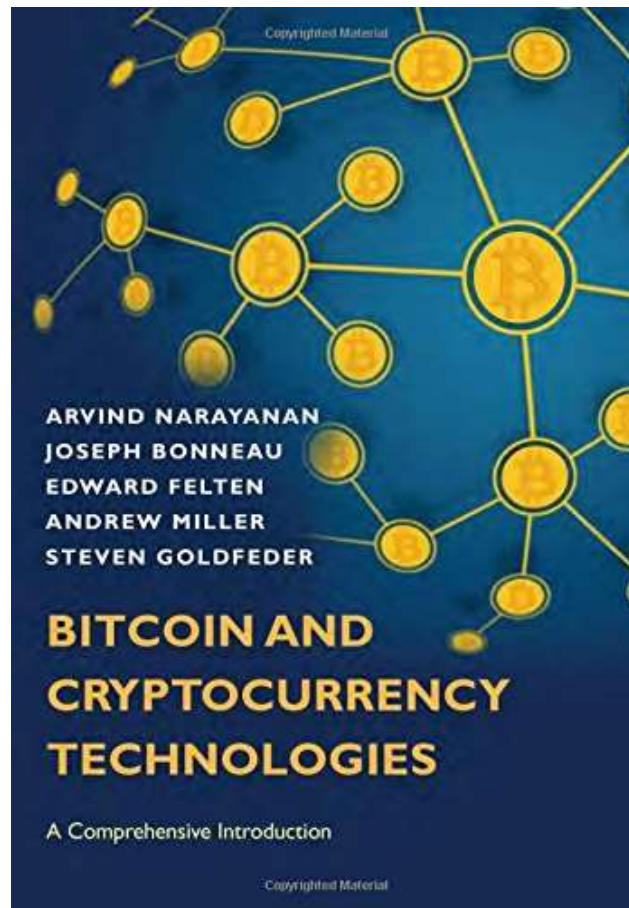


Blockchain & Business Application

Lecture:
Managing and Protecting
Bitcoin Assets

Chapter 4: How to Store and Use Bitcoins



Simple Local Storage

- To spend a bitcoin, you need to know
 - some public info
 - Identity of the coin
 - How much it's worth, etc
 - some secret info
 - Secret key (sk) of the owner of bitcoin (you)
 - You can always get the public info
 - It's all about storing and managing your keys

- Three goals:
 - Availability
 - Capability of spending anytime
 - Security
 - Nobody else can spend your coins
 - Convenience
 - Key management should be easy
- Challenging, trade-offs

- Simplest: Carry on your computer/phone
 - Convenient!
 - Not always available
 - Not always secure
- Wallet software
 - Keeps track of your coins
 - Manages keys

- Encoding keys: base58 and QR codes
 - To spend or receive bitcoins, you also need a way to exchange an address with the other party
 - Two main ways addresses are encoded can be communicated from receiver to spender:
 - as a text string or
 - as a QR code.

- base58:

- Upper & lower case letters & numbers
- Excluding some confusing: {O, 0}

1A1zP1eP5QGefi2DMPTfTL5SLmv7DivfNa

- Barcode



- Vanity addresses:
 - Satoshi Bones (gambling website)
 - 1bonesEeTcABPjLzAb1VkFgySY6Zqu3sX
 - Starts with 1: pay-to-pubkey-hash
 - How to “find” this address?

Hot/Cold Storage

- Pocket: **Hot** (convenient but risky)
- Locked somewhere offline: **Cold**
 - Safer
 - More Secure
 - Not convenient
- Separate keys for each, why?

- Cold: Offline
 - Can't connect hot&cold
 - But Cold can receive coins
 - Can transfer Hot → Cold anytime

- Problem managing Cold addresses:
 - Privacy: Receive each coin at a fresh address
 - Cold is offline: How to manage
 - Solution:
 - Create a bunch of addresses for Cold
 - Use one each
 - Drawback: Connect to repeat periodically

- More effective Solution:
 - Hierarchical wallets
 - Cold: Use unbounded addresses
 - Hot learns these via short&one-time comm.
 - Requiring more cryptography
 - generateKeys that generates a public key, different
 - Some DS don't support hierarchical key generation
 - ECDSA (Bitcoin) supports.

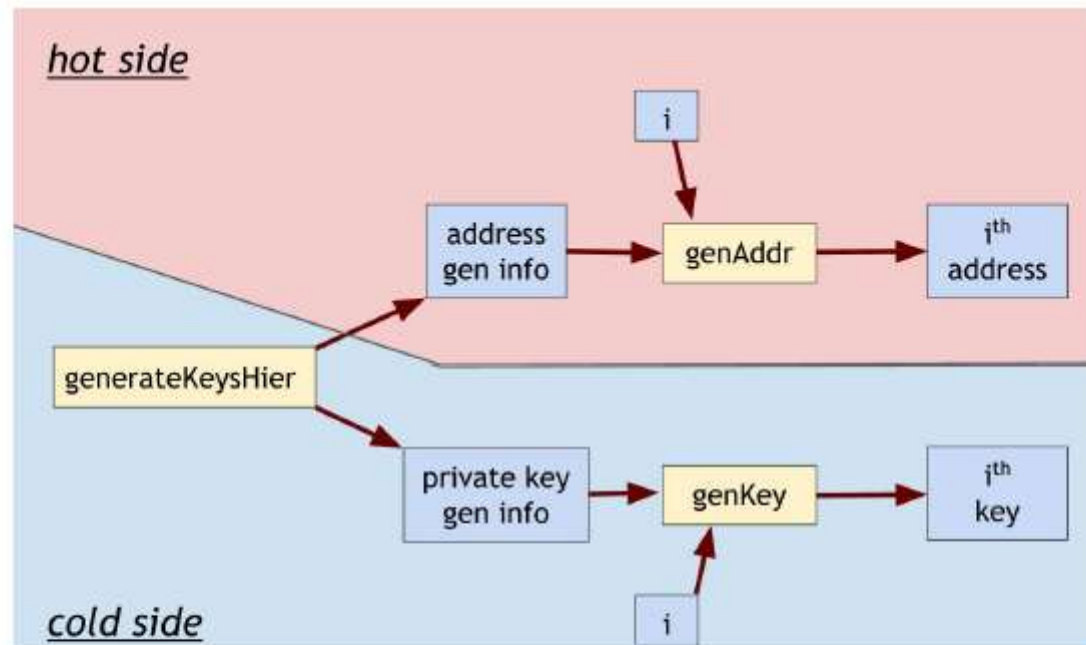


Figure 4.2: Schema of a hierarchical wallet. The cold side creates and saves private key generation info and address generation info. It does a one-time transfer of the latter to the hot side. The hot side generates a new address sequentially every time it wants to send coins to the cold side. When the cold side reconnects, it generates addresses sequentially and checks the block chain for transfers to those addresses until it reaches an address that hasn't received any coins. It can also generate private keys sequentially if it wants to send some coins back to the hot side or spend them some other way.

- Different approaches for Cold
 - Store in a device, keep it safe (lock it up)
 - Brain Wallet
 - Control access of coins via passphrase (no device)
 - Poor physical security (traveling, etc)
 - Have an algorithm to turn passphrase to keys
 - Hash functions ☺ passphrase → private key
 - Security: Can an adversary guess passphrase?

- Generating memorable passphrases.
 - One procedure that gives 80 bits of entropy:
 - Pick a random sequence of 6 words from among the 10,000 most common English words
 - $6 \times \log_2(10000)$ is roughly 80.

***worn till alloy focusing okay reducing
earth dutch fake tired dot occasions***

- Easier than whole random letters, but security?

- Key- stretching :
 - Deliberately slow the function
 - e.g. use SHA-256 but compute 2^{20} iterations
 - pros, cons?
- If brain-wallet gone, coins gone forever!

- Paper Wallet



Figure 4.3: A Bitcoin paper wallet with the public key encoded both as a 2D barcode and in base 58 notation. Observe that the private key is behind a tamper-evident seal.

Splitting and Sharing Keys

- Too technical, only for who is interested
- Research opportunity
 - Threshold cryptography
 - Multi-signatures

PHYSICAL REVIEW A 71, 012314 (2005)

Threshold quantum cryptography

Yuuki Tokunaga,^{1,2,*} Tatsuaki Okamoto,¹ and Nobuyuki Imoto²

¹*NTT Information Sharing Platform Laboratories, NTT Corporation, 1-1 Hikari-no-oka, Yokosuka, Kanagawa 239-0847, Japan*

²*Division of Materials Physics, Graduate School of Engineering Science, Osaka University, Toyonaka, Osaka 560-8531, Japan*

(Received 9 August 2004; published 10 January 2005)

We present the concept of *threshold collaborative unitary transformation* or *threshold quantum cryptography*, which is a kind of quantum version of threshold cryptography. Threshold quantum cryptography states that *classical* shared secrets are distributed to several parties and a subset of them, whose number is greater than a threshold, collaborates to compute a quantum cryptographic function, while keeping each share secretly inside each party. The shared secrets are reusable if no cheating is detected. As a concrete example of this concept, we show a distributed protocol (with threshold) of conjugate coding.

DOI: 10.1103/PhysRevA.71.012314

PACS number(s): 03.67.Dd

Online Wallets & Exchanges

- Online wallets:
 - like local, you might manage yourself,
 - except the information is stored in the cloud,
 - access it using a web interface / an app.
 - Popular: Coinbase and blockchain.info.
- Security:
 - Password protected
 - You need to trust

- Bitcoin Exchanges
 - Similar to traditional banking
 - Accepts fiat currency & bitcoins
 - Promise to give back on demand
 - Exchange between fiat&crypto currencies
 - Match customers

- Exchange example:
 - I have 5000\$ and 3 bitcoins
 - I put in an order to buy 2 bitcoins for 580\$ each
 - If the exchange finds someone, transaction OK
 - Now I have 3840\$ and 5 bitcoins.
- No real transaction occurred, no new block
- Only promises changed

- Pros: Connect the Bitcoin economy
- Cons: Risks (bank risks)
 - Bank run: Too many people show up
 - Ponzi scheme
 - Hack

- Statistics:
 - (by 2013) 18 of 40 exchanges closed!
 - “Mt. Gox”: Japanese, largest exchange bankrupted
 - Still in US&Jp courts
- Regulations for traditional banks
 - Minimum reserve requirement; %3-10 @USA
- For Bitcoin exchanges
 - None!

- Proof of Reserve
 - To give some comfort to customers
 - Makes a self-transaction
 - Not a proof: Just that there's someone willing to coop
 - Under-claim possible (transaction: 10K but actual 15K)

Payment Services

- Store OK
- Manage OK
- Spend?
- Merchants
 - Better accept 😊
 - Probably immediately exchange to \$
 - Should not worry much about the technology
 - Low risk


- Risks
 - Website may go down → \$
 - Security of handling coins
 - Exchange rate fluctuate
- Payment services aim to address these risks
 - Provide interface for generating a pay-with-Bitcoin button → generate a HTML snippet to embed


Choose A Way To Accept Bitcoin or [see examples](#) of each payment method.


Type ☒ Button ☐ Hosted Page ☐ iFrame ☐ Email Invoice


Payment ☒ Buy now ☐ Donation ☐ Subscription

Button Style

☒ 

☐ 

☐ 

☐ 

Item Name

Amount

Item Description

Send Funds To

[Show Advanced Options](#)

- Process of receiving Bitcoin payments through a payment service might look like this to the merchant:
 - 1. The merchant goes to payment service website and fills out a form describing the item, price, and presentation of the payment widget, etc. (example: Coinbase.)
 - 2. The payment service generates HTML code that the merchant can drop into their website.
 - 3. The customer clicks the payment button, various things happen in the background and eventually the merchant gets a confirmation saying,
“a payment was made by customer ID [customer-id] for item [item-id] in amount [value].”

- Manual process OK for donations, few items.
- CTRL C+V HTML code for thousands of items?
- Payment services also provide
programmatic interfaces
for dynamically generated webpages.

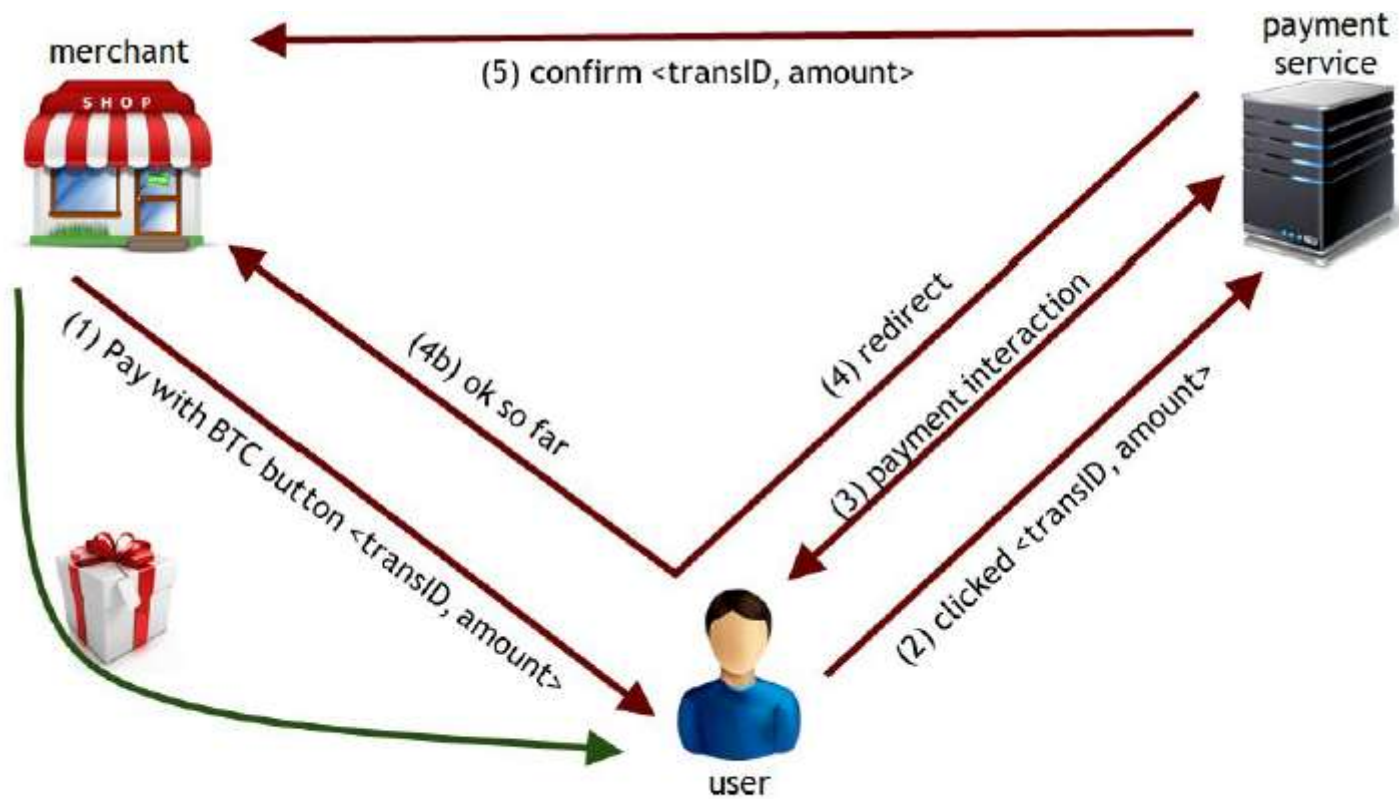
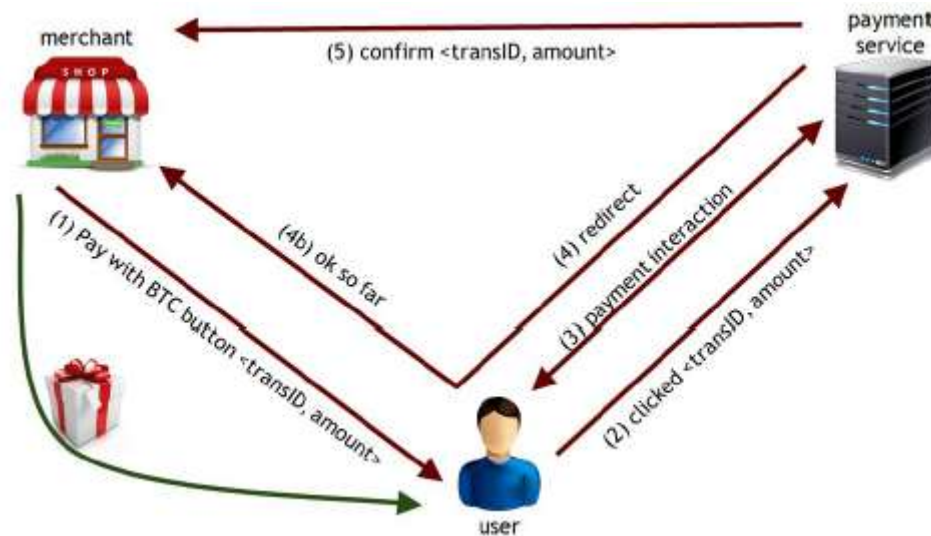
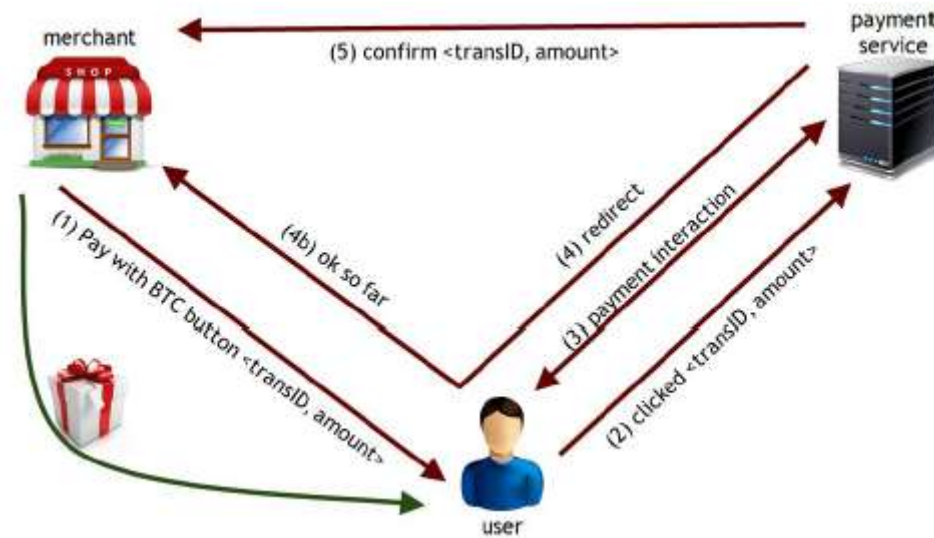


Figure 4.8: Payment process involving a user, merchant, and payment service.



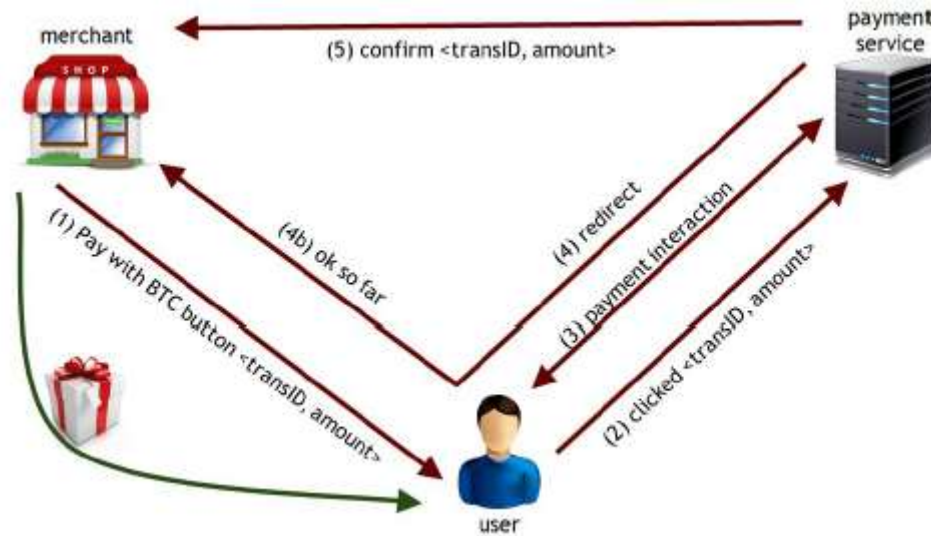
- Step 1

- The user picks out an item to buy on the merchant site,
- The merchant will deliver a webpage which will contain the Pay with Bitcoin button, which is the HTML snippet provided by the payment service.
- The page will also contain a transaction ID — which is an identifier that's meaningful to the merchant and allows them to locate a record in their own accounting system — along with an amount the merchant wants to be paid.



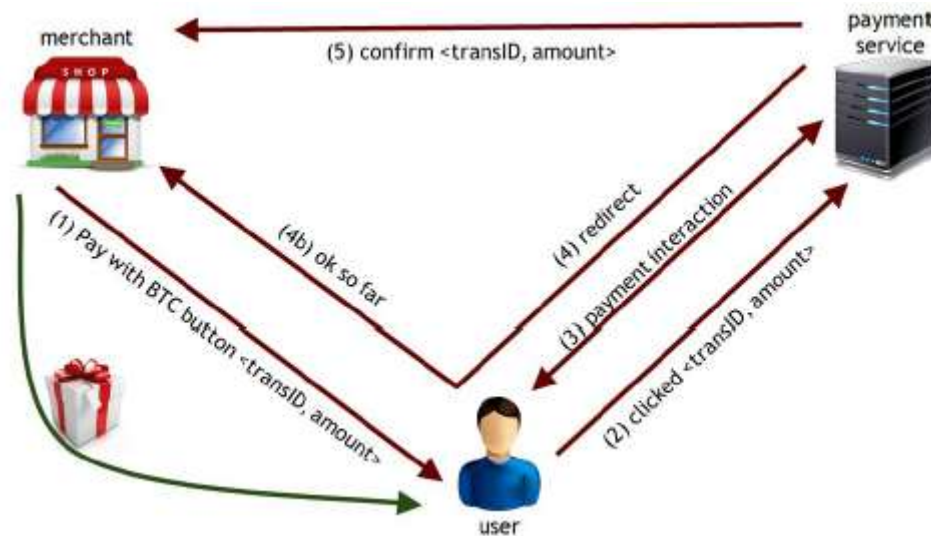
- Step 2

- If user wants to pay with bitcoins, clicks that button.
- That will trigger an HTTPS request to the payment service saying that the button was clicked, passing on the identity of the merchant, the merchant's transaction ID, the amount.



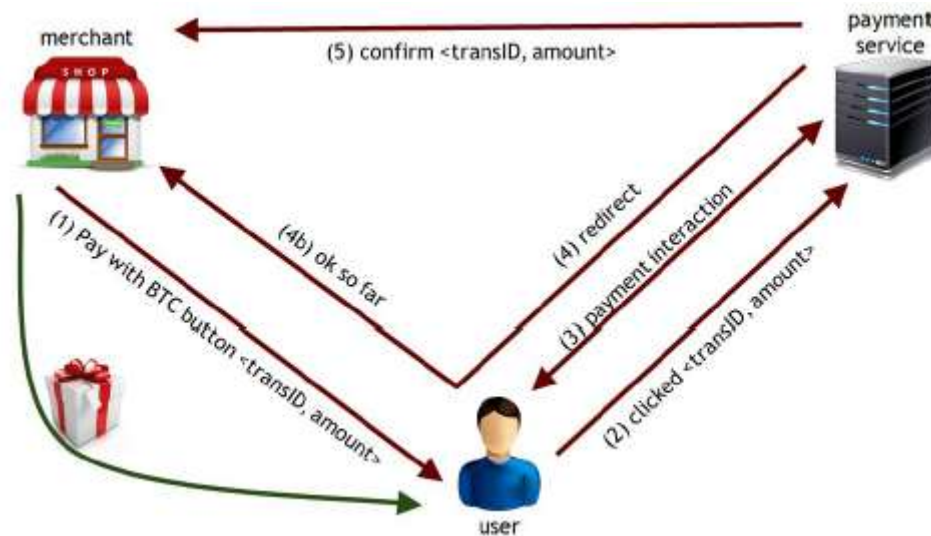
- Step 3

- Now the payment service knows that this customer wants to pay a certain amount of bitcoins,
- and so the payment service will pop up some kind of a box,
- or initiate some kind of an interaction with the user.
- This gives the user information about how to pay,
- the user will then initiate a bitcoin transfer to the payment service through their preferred wallet.



- Step 4

- Payment service will redirect the browser to the merchant, passing on the message from the payment service that it looks okay so far.
- This might mean, e.g. the payment service has observed the transaction broadcast to the peer-to-peer network, but the transaction hasn't received enough confirmations.
- This completes the payment as far as the user is concerned, with the merchant's shipment of goods pending a final confirmation from the payment service.



- Step 5
 - The payment service later directly sends a confirmation to the merchant containing the transaction ID and amount.
 - By doing this the payment service tells the merchant that the service owes the merchant money at the end of the day.
 - The merchant then ships the goods to the user.
- The payment service keeps a small % as a fee

- In summary:
 - the customer pays bitcoins
 - the merchant gets dollars, minus a small %.
 - everyone is happy 😊
- Recall that the merchant wants \$.
- The payment service handles everything else:
 - receiving bitcoins from customers
 - making deposits at the end of the day.

- Crucially, the payment service absorbs all of the risk.
 - Security risk, so it needs to have good security procedures to manage its bitcoins.
 - Exchange rate risk: Loss, gain.. Risk!
- Absorbing it is part of the payment service's business.
- Participate in the exchange market!

Transaction Fees

- Any transaction on the blockchain might include a transaction fee
- Total value of coins IN - Total value of coins OUT
- $IN > OUT$
- As of 2016
- Why fee?
 - Miner: Who builds your transaction to a block
 - That miner's block is longer → Takes longer to propagate
 - Another block was probably found almost simultaneously
 - Cost to Network & Miners

- Nodes are not paid (currently)
- You can set your fee:
 - None
 - High → Relayed & Recorded quickly & reliably

Default Transaction Fees

- No fee if all satisfied:
 - Transaction < 1KB
 - All outputs >= 0.01 BTC
 - Priority is large enough:
*(sum of input age * input value) / (transaction size)*

Typical transaction: 400 Bytes → Free
- Otherwise: 1mBTC per 1KB

Currency Exchange Markets

- Similar in many ways to market for \$ & Euro
- Live value, fluctuate over time
- March 2015
 - Bitfinex per 1 day: 70K BTC = 21 M\$
- Local sites: localbitcoins.com
 - Meet at a café/park
- Directly meet at places, pre-scheduled
 - Anonymous!

Supply & Demand

- Supply
 - October 2015: 13.9 M BTC
 - Eventually: 21 M BTC
- Demand, why?
 - Mediating fiat currency transactions
 - Alice & Bob, use BTC just for transfer
 - Alice buys, sends; Bob receives, sells
 - During transaction, BTCs out of circulation
 - Investment
 - Buy & Hold: Out of circulation

CUMULATIVE BTC IN CIRCULATION

