# Summary

Once the Bitcoin network was in place, many ideas started to appear about how to use it in ways going beyond the original vision of a network for simple payments:

- Bitcoin as an immutable log
- Sub-currencies (coloured coins)
- Escrow transactions
- Technical Fixes: Pay to script Hash
- Off-chain Payment Channels
- Smart Contracts
- Limitations of Bitcoin as a Smart Contract Platform



These lead ultimately to an understanding that a more flexible platform is desirable….

Reading: Bitcoin and CryptoCurrency Technologies,
CH9 - Bitcoin as a Platform

# “Proof of Existence”: Bitcoin as an Immutable Log

The Bitcoin record is **immutable** in the following sense:

Once a block is sufficiently deep in the longest chain,

- with very high probability, it will always be in the longest chain,
- a very large amount of work (solving hash puzzles) is required to erase/alter it



Encoding information into Bitcoin transactions enables Bitcoin to be used as a digital time stamping service

Essentially, this replaces (centralised) Surety Technologies’ use of the NY Times by the decentralised Bitcoin network

Examples:

Factom https://www.factom.com (company now liquidated)
developed an app that does this, e.g., for mortgage documents, counteracting fraud

Image: https://commons.wikimedia.org/wiki/File:Quebrada_de_Cafayate,_Salta_(Argentina).jpg

# Encoding information into a transaction

There are various techniques by which extra information can be encoded into a transaction

One of the common ones uses the opcode:

OP_RETURN = abort this execution (verification always fails)



An output with an locking script of the form

OP_RETURN <data> or OP_RETURN <hash(data)>



can never be spent, but encodes <data> into the transaction



Miners now accept such outputs with zero value as valid

# Overlay Assets / Coloured Coins

Bitcoin provides a way to secure ownership of amounts of a particular asset: Bitcoins

*Coloured Coins* use Bitcoin as public record of ownership other asset types,
e.g., concert tickets, diamonds, artwork, real estate, stocks
with proof of ownership/transfer by digital signature



Method:

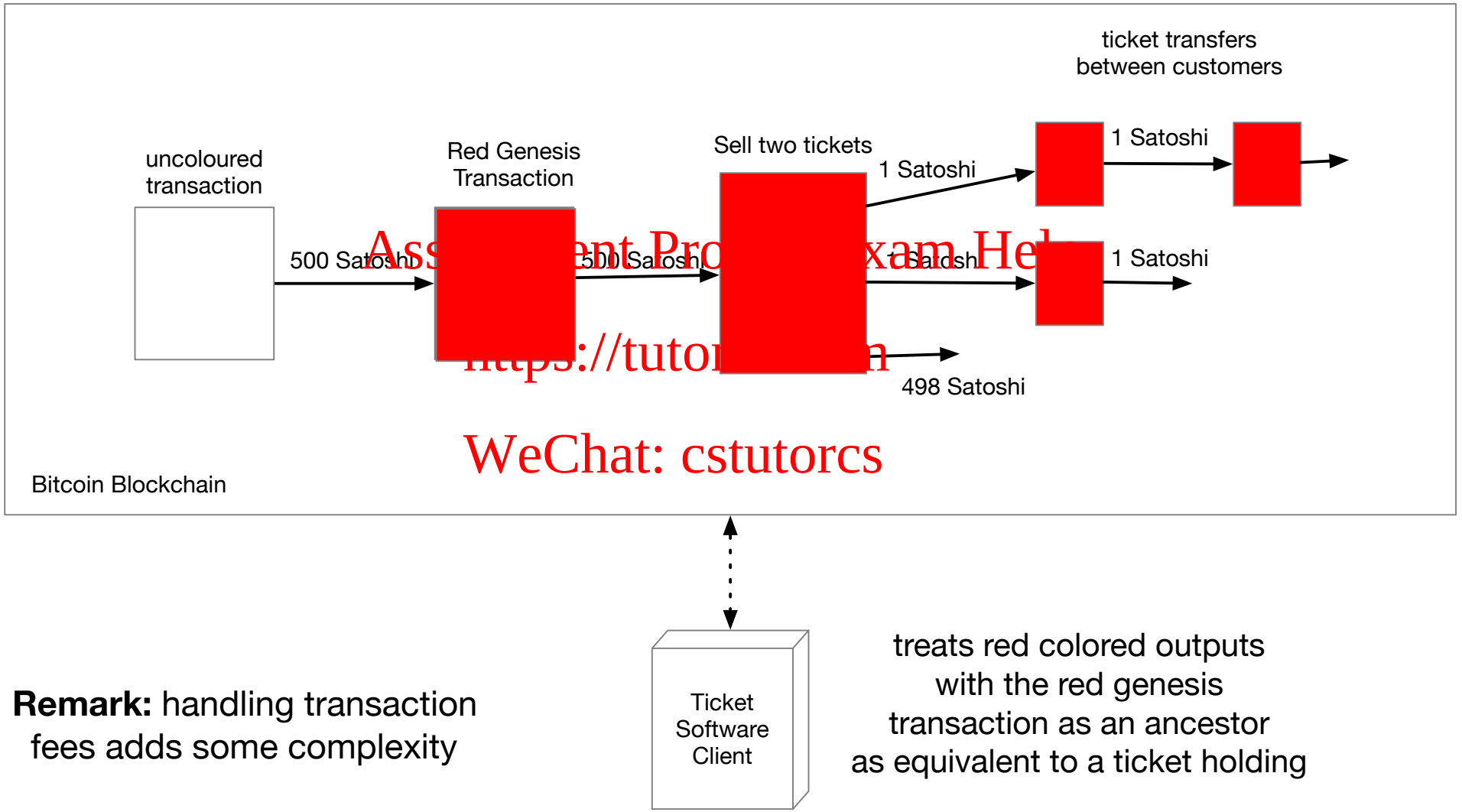
- Encode special markers ("colours") into Bitcoin transactions to encode asset information
- Use client software that understands the marker scheme



- Use low Bitcoin values (e.g. satoshis) to denominate amounts of the asset holding
- To ordinary Bitcoin clients, the coloured transactions look just like ordinary transactions

- Colored Coins Whitepaper, 2012, Yoni Assia, Vitalik Buterin, m liorhakiLior, Meni Rosenfeld, Rotem Lev
  https://docs.google.com/document/d/1AnkP_cVZTCMLIzw4DvsW6M8Q2JC0lIzrTLuoWu2z1BE/edit#heading=h.v3px1rgmf10o

Colour Scheme: red = “outputs represent a number of tickets to see the Rolling Bones on Feb 23”






**Remark:** handling transaction fees adds some complexity

treats red colored outputs with the red genesis transaction as an ancestor as equivalent to a ticket holding

# Proving Ownership of a Ticket

A **Challenge/Response** Protocol:

Alice, with public key KA and private key $KA^{-1}$, owns an unspent ``pay to public key KA" ticket output,

She goes to the concert.



**Alice:** Please let me in. Here is the ID of the transaction on the blockchain that proves I have a ticket. (ID)



**Ticket Taker:** "Yes, I can see on the blockchain that (ID) is a transaction descended from the genesis transaction, with an unspent output paid to KA that represents a valid ticket. But please prove to me that you are the owner of that ticket. Sign this random number (N)



**Alice:** Here you are (N signed using $KA^{-1'}$)

**Ticket Taker:** (verifies the signature using KA, records (ID) as ``has entered") Thank you. Enjoy the concert!

# An Inconvenience of Bitcoin

Suppose Alice wants to pay Bob to purchase Bob's widget

Bob wants the payment secured in his complicated multi-sig wallet

So Alice needs to construct a transaction with an output script that says

"Unlock this output when Bob's multi-sig conditions A, B, C…,Z are satisfied"



Problems:

- Does Alice's wallet understand/handle this unlocking script?
- Why should Alice care how Bob wants to secure his money?
- Shouldn't it be Bob's job to ensure the funds are locked using this rule?

# Pay to Script Hash Transactions

"Pay to Script-Hash" transactions were developed to solve this problem. They essentially say:

Locking script:

OP_HASH160 <hash-value> OP_EQUAL

i.e., "release this output when provided with data whose hash is <hashvalue>"

Unlocking script:

<signatures> <script>



New validation rule applied by miners:

after checking that {unlocking script, locking script} evaluates to TRUE,
(i.e., hash(<script>) = <hash-value>)
verify that <signatures> <script> evaluates to TRUE (*)

This new rule was brought into effect by a soft fork. (Old clients don't check line (*) )

**Benefit:** Alice now does not need to know Bob's complicated <script>, just the <hash-value>

# Escrow Transactions using Multi-Sig

**Problem:**

Alice wants to buy goods costing 1 Bitcoin from Bob,
but she does not trust him to deliver after she pays

Bob wants to sell goods to Alice,
but he does not trust her to pay if he delivers first.

How can they transact, so that each is protected against the other?



**Solution:**

Alice and Bob both trust Charlie to arbitrate any disputes between the two.

Alice pays 1 Bitcoin with multi-sig unlocking condition:
"Pay out on 2/3 signatures from Alice,Bob,Charlie"

If Bob delivers and Alice is happy, Alice & Bob sign a transaction to pay Bob

If Bob delivers and Alice signs the delivery receipt but refuses to pay,
Bob shows Charlie the receipt and Bob & Charlie sign a transaction to pay Bob

If Bob does not deliver and Bob can't show a receipt signed by Alice,
Alice and Charlie sign a transaction to pay the money back to Alice

# Payment Channels

Bitcoin's transaction throughput is limited to 7 transactions/sec, uncompetitive

**Payment Channels** are an approach to obtaining a faster throughput
(example: Lightning Networks http://lightning.network)



**Main idea :**

- parties wanting to interact frequently, lock up collateral value in a Bitcoin transaction
- *handle most transactions off-chain*, with *zero confirmations*,
  by an exchange of signed messages between the parties
- use the Bitcoin blockchain as a fraud-protection mechanism
- after a period of transacting, the parties can claim their appropriate balance by providing a signed message as evidence
- Bitcoin script is used to resolve disputes between the two parties about their balance

# Example: Alice using Bob's internet service

Bob provides an online music distribution service

Alice uses it for one year on a pay 100 Satoshi per download basis, up to 50,000 Satoshi

**Idea:** Alice creates a transaction on chain with
input : 50,000 Satoshi, signed Alice
output (***): 50,000 Satoshi, payable
on multi-signature by Alice and Bob *after Dec 30 (this year)*
OR on signature by Alice *after Jan 30 (next year)*



Alice and Bob each keep a balance, initially Alice : 50,000, Bob: 0

Each time she does a download, worth 100 Satoshi, Alice gives Bob a transaction, signed by Alice, saying



"input: (***) outputs: X+100 Satoshi to Bob, 50,000-(X+100) Satoshi to Alice"

where X was Bob's previous balance.

Bob does not send these transactions to the network, but saves them up to the end of the year.

On Jan 1, he signs and sends to the network the latest transaction that gives him the highest payout.

If Bob does not send a transaction by Jan 30, after Jan 30 Alice can send transaction to reclaim her deposit

# Limitations of Bitcoin Script

- No loops
- Outputs release a fixed amount to one location (no dependent payments)
- Bitcoin outputs are one-shot: no support for multi-stage contracts
- Bitcoin Script branching is blind to most blockchain data (current transaction only)

# Example: Hedging uses Dependent Payments

Alice has 1 BTC but believes the BTC price will fall and
wants to protect the $A value of her holding
(suppose currently 1 BTC= $A 50,000),

Bob is convinced the price of BTC is going up and is prepared to bet up to 0.1 BTC on it.



If the $A value of 1 BTC drops to X < 50,000 then Alice has BTC worth $A X ,
and has lost $A 50,000 - X



At the then prevailing exchange rate, this is (50,000 /X) - 1 BTC



They devise the following contract

"If BTC→AUD = X < 50,000 on next June 30
then Bob pays Alice max(50,000/X - 1 BTC, 0.1 BTC)
else Alice pays Bob 0.1 BTC"

Idea for an implementation:

An exchange trusted by Alice and Bob signs exchange rate messages

Alice and Bob create a transaction with

inputs: 0.1 BTC from Alice, 0.1 BTC from Bob

case 1: if exchange signed "BTC/AUD = X on June 30" and X < 50,000

pay 0.1 + max(50,000/X - 1 BTC, 0.1 BTC) to Alice

pay 0.1 - max(50,000/X - 1 BTC, 0.1 BTC) to Bob

case 2: if exchange signed "BTC/AUD = X on June 30" and X >= 50,000

pay 0.2 BTC to Bob



Bitcoin Script does not support such contingent payouts! We could simulate this using a trusted third party, but can't we avoid that?

# Example: Staged Contracts

TIPS are a type of loan contract issued by the US government that protects the lender against inflation

Lender lends US government $100 for 10 years at 3% annual interest rate

Year 1: US government pays lender $ 100x0.03= $3 interest

Year 2: if inflation in year 1 was x1, principal increases to $100 * (1+x1)

US government pays lender $ 100 * (1+x1) * 0.03 interest



Year 3: if inflation in year 2 was x2, principal increases to $100 * (1+x1) * (1+ x2)



US government pays lender $ 100 * (1+x1) * (1+ x2)* 0.03 interest



…

Year 10 : if inflation in year 9 was x9, principal increases to $100 * (1+x1) * (1+ x2) *… * (1+x9)

US government pays lender $3 interest $ 100 * (1+x1) * (1+ x2)* … (1+x9) * 0.03 interest

Year 10 end: if inflation in year 2 was x10,

US government repays principal $100 * (1+x1) * (1+ x2) *… * (1+x9) * (1+ x10)

# Summary

- Bitcoin as an immutable log
- Coloured coins (e.g., concert tickets)
- Escrow transactions
- Pay to script Hash
- Off-chain Payment Channels
- Limitations of Bitcoin as a Smart Contract Platform

# Smart Contracts

"Smart Contracts"

- = contracts whose terms are enforced by a *computer system*, rather than by the courts
- term coined by Nick Szabo in blogs, e.g.
  "Formalizing and Securing Relationships on Public Networks", 1997
  https://nakamotoinstitute.org/formalizing-securing-relationships/



Examples like the above show the Bitcoin network and Bitcoin Script



- provides functionality for some types of smart contract enforcement
- where the computer system is *decentralised*,
  so "trustless" (don't place trust in a centralised operator of the computer system)



- However, Bitcoin has *limitations* with its ability to express useful smart contracts
- New application ideas (e.g. Lightning) have required
  hacky/tricky implementations and/or difficult to effect forks of Bitcoin

Enter Ethereum …..