Intro to a blockchain - the Teddy Token

A blockchain is a secure record of transactions

Transactions may consist of
- registrations or exchanges of “digital assets” such as coins and tokens*
- registrations of and transactions with “smart contracts”

Transactions cannot be reversed or revised. If an attempt is made, the attempt is obvious and can be stopped

Transactions are grouped into blocks and the blocks are linked into a chain by a secure method

Copies of the blockchain are distributed over a peer network such that there is no single point of failure or control

**Technically, Teddy Token is a digital coin, but “token” rhymes with Teddy!*

Technology involved in blockchains

- cryptographic hashes
- hexadecimal and binary numbers
- consensus, e.g., by proof-of-work
- modular arithmetic
- public key encryption
- peer-to-peer networks over the Internet - P2P

All discussed here except P2P

Did you know that Catoshi Purramoto named the chain in honor of his feline friend Teddy?

Transactions are anonymous and irreversible

```
From: bf61f561918f37cf441fb8a6a1094b7a
To: 9412daed1e0bd204f652677a80192ea9
Amount: 2
Date: Mon Feb 22 2021 13:44:10 GMT-0800 (PST)
Hash: b819e116c0aca3dd431c411b31040160
```

Addresses of sender and recipient are unique and the people's identity are known only to themselves…
or their wallet company, that is

A hash is a cryptographic digest of the transaction that can be used to verify that the transaction hasn't be revised after it was entered into the blockchain

Any tampering? Re-hash and see if the new hash matches that in the chain!

Teddy Token transactions have only one sender and one recipient. Other blockchains allow multiple senders and recipients in one transaction.

Other blockchains may charge the sender a transaction fee

Cryptographic hashes are key to blockchains

A cryptographic hash algorithm processes a message to produce a hash value that is:

- of fixed length, regardless of the length of the message > modular arithmetic
- that cannot be decrypted to determine the original message > "confusion"
- that changes dramatically when only a trivial change is made in the message > "diffusion"

The terms "confusion" and "diffusion" in cryptography are due to Claude Shannon

Teddy Token uses the MD2 hash - out of date but relatively easy to understand

Bitcoin uses SHA-256

```
From: bf61f561918f37cf441fb8a6a1094b7a
To: 9412daed1e0bd204f652677a80192ea9
Amount: 2
Date: Mon Feb 22 2021 13:44:10 GMT-0800 (PST)
Hash: b819e116c0aca3dd431c411b31040160
```

Addresses are hashes of a user's public cryptographic key

This hash is a cryptographic digest of the transaction that can be used to verify that the transaction hasn't be revised after it was entered into the blockchain

What are those strings of numbers and letters?
000869e6e37d5f11f5b71d2641c139d4
Those are **hexidecimal** numbers

Some Number Systems

power, counting from 0 to the left

0 to (base – 1)　　base

Name	Base	Numerals	Example
decimal	base 10	0123456789	

$\# \times 10^2 \quad \# \times 10^1 \quad \# \times 10^0 = \#$

					1	9	3

Why? Easy to compute in your head & by hand

$1 \times 10^2 + 9 \times 10^1 + 3 = 100 + 90 + 3 = 193$ decimal

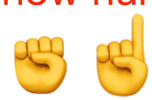
$\# \times 2^7 \quad \# \times 2^6 \quad \# \times 2^1 \quad \# \times 2^0 = \#$

binary　base 2　01

1	1	0	0	0	0	0	1

Why? Digital storage is by bits: 0/1, on/off, high/low…
This is how numbers are stored physically by computers

$2^7 + 2^6 + 1 = 128 + 64 + 1 = 193$ decimal


your digits

$\# \times 16^2 \quad \# \times 16^1 \quad \# \times 16^0 = \#$

hexidecimal　base 16　0123456789abcdef

						c	1

Why? Easy conversion between hex & binary, 1 hex to 4 bits

$12 \times 16^1 + 1 = 192 + 1 = 193$ decimal

The start of the Teddy Token blockchain - a secure record of anonymous transactions

A **CRYPTOGRAPHIC HASH** of a message - in this case a block header - will change dramatically if any single character in the message is changed. This and the presence of the previous block hash in each block header ensure the integrity of the blockchain

Genesis Block - Catoshi's gift to his friends

```
Version: TedTok_01
Block number: 0
Previous hash: 00000000000000000000000000000000
Merkle root: 5f5de6dab6f9f1a636690177cbe629dc
Date: Mon Feb 22 2021 13:43:47 GMT-0800 (PST)
Target: 3
Nonce: 11911
Hash: 0001b04ff27cb9ab55b86477f95815de
----------------------
Number transactions: 5
----------------------
To: 8fca7b9ab6fac411b2443c1303d3bc56
Amount: 10
Hash: 425a3eba0375c91ff94af7ff9fd29209
----------------------
To: f157bab61715d2bca9d81ada45bad57c
Amount: 10
Hash: 402ea2348edb95de31565ed152729aaf
----------------------
To: 263f99c540f4168132cc5fee5726b5ce
Amount: 10
Hash: ebc383bbf91325b0997c9147591c35e5
----------------------
To: bf61f561918f37cf441fb8a6a1094b7a
Amount: 10
Hash: dba01bfd264db14cb52123c6dbeb9f51
----------------------
To: 9412daed1e0bd204f652677a80192ea9
Amount: 10
Hash: d4704122a45d67f174fa36d53dd6b951
============================================
```

Block 1

block header

```
Version: TedTok_01
Block number: 1
Previous hash: 0001b04ff27cb9ab55b86477f95815de
Merkle root: 21e3776b697c3d22922876e7e8ae36c2
Date: Mon Feb 22 2021 13:44:14 GMT-0800 (PST)
Target: 3
Nonce: 4328
Hash: 000f5b9d7ca44d91ba33e9934e82cf27
----------------------
Number transactions: 3
----------------------
Miner reward to: f157bab61715d2bca9d81ada45bad57c
Amount: 0.99
Hash: ad5ce451141e8c30baf4f55e6b4ddb14
----------------------
From: bf61f561918f37cf441fb8a6a1094b7a
To: 9412daed1e0bd204f652677a80192ea9
Amount: 2
Date: Mon Feb 22 2021 13:44:10 GMT-0800 (PST)
Hash: b819e116c0aca3dd431c411b31040160
----------------------
From: f157bab61715d2bca9d81ada45bad57c
To: 8fca7b9ab6fac411b2443c1303d3bc56
Amount: 1
Date: Mon Feb 22 2021 13:44:03 GMT-0800 (PST)
Hash: 5c0a2c83ae1366d3074e923198fec388
============================================
```

Block 2

```
Version: TedTok_01
Block number: 2
Previous hash: 000f5b9d7ca44d91ba33e9934e82cf27
Merkle root: 2569e9f51001fdb295b836228bbfcc84
Date: Mon Feb 22 2021 13:44:40 GMT-0800 (PST)
Target: 3
Nonce: 856
Hash: 000869e6e37d5f11f5b71d2641c139d4
----------------------
Number transactions: 3
----------------------
Miner reward to: 263f99c540f4168132cc5fee5726b5ce
Amount: 0.99
Hash: 3490eb317353dcc32715788714a13769
----------------------
From: 9412daed1e0bd204f652677a80192ea9
To: 8fca7b9ab6fac411b2443c1303d3bc56
Amount: 3
Date: Mon Feb 22 2021 13:44:36 GMT-0800 (PST)
Hash: bb281c938be6c2bee7ffc05bdb4f1e23
----------------------
From: bf61f561918f37cf441fb8a6a1094b7a
To: f157bab61715d2bca9d81ada45bad57c
Amount: 1
Date: Mon Feb 22 2021 13:44:29 GMT-0800 (PST)
Hash: 26a537138d436bbd55a7287edf152951
============================================
```

Any tampering? Re-hash and see if the new hash matches that in the chain!

Web Labs at www.ReactorLab.net

The genesis block of all blockchains - Satoshi’s gift to Hal Finney of 50 bitcoins created out of thin air!

Genesis block - Bitcoin Wiki

Main network genesis block

Here is a representation of the genesis block[1] as it appeared in a comment in an old version of Bitcoin (line 1613). The first section defines exactly all of the variables necessary to recreate the block. The second section is the block in standard printblock format, which contains shortened versions of the data in the first section.

```
GetHash()      = 0x000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f
hashMerkleRoot = 0x4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b
txNew.vin[0].scriptSig     = 486604799 4
0x736B6E616220726F662074756F6C69616220646E6F63657320666F206B6E697262206E6F20726F6C6C65636E61684320393030

txNew.vout[0].nValue       = 5000000000
txNew.vout[0].scriptPubKey =
0x5F1DF16B2B704C8A578D0BBAF74D385CDE12C11EE50455F3C438EF4C3FBCF649B6DE611FEAE06279A60939E028A8D65C10B730
 OP_CHECKSIG
block.nVersion = 1
block.nTime    = 1231006505
block.nBits    = 0x1d00ffff
block.nNonce   = 2083236893

CBlock(hash=000000000019d6, ver=1, hashPrevBlock=00000000000000, hashMerkleRoot=4a5e1e,
nTime=1231006505, nBits=1d00ffff, nNonce=2083236893, vtx=1)
  CTransaction(hash=4a5e1e, ver=1, vin.size=1, vout.size=1, nLockTime=0)
    CTxIn(COutPoint(000000, -1), coinbase
04ffff001d0104455468652054696d65732030332f4a616e2f32303039204368616e63656c6c6f72206f6e206272696e6b206f66

    CTxOut(nValue=50.00000000, scriptPubKey=0x5F1DF16B2B704C8A578D0B)
  vMerkleTree: 4a5e1e
```

Bitcoin uses the SHA-256 hash, which is more secure than Teddy Token’s MD2 hash and twice as long

The very first 50 bitcoin

Hash

The hash of the genesis block, **000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f**,[1] has two more leading hex zeroes than were required for an early block.

Coinbase

The coinbase parameter (seen above in hex) contains, along with the normal data, the following text:[2]

> The Times 03/Jan/2009 Chancellor on brink of second bailout for banks[1]

a comment in the block included a reference to bank bailouts published in The Times, Jan 3, 2009

The **Blockchain Ledger** lists all addresses and can compute their current balances from verified transactions

Simplified Blockchain Ledger

```
Address: 8fca7b9ab6fac411b2443c1303d3bc56
Balance: 14
----------------------
Address: f157bab61715d2bca9d81ada45bad57c
Balance: 10.99
----------------------
Address: 263f99c540f4168132cc5fee5726b5ce
Balance: 10.99
----------------------
Address: bf61f561918f37cf441fb8a6a1094b7a
Balance: 7
----------------------
Address: 9412daed1e0bd204f652677a80192ea9
Balance: 9
```

An anonymous user's **ADDRESS** is the hash of their **PUBLIC CRYPTOGRAPHIC KEY**

The **Merkle root** is a hash produced from the hashes of the transactions in the block.

```
Version: TedTok_01
Block number: 2
Previous hash: 000f5b9d7ca44d91ba33e9934e82cf27
Merkle root: 2569e9f51001fdb295b836228bbfcc84
Date: Mon Feb 22 2021 13:44:40 GMT-0800 (PST)
Target: 3
Nonce: 856
Hash: 000869e6e37d5f11f5b71d2641c139d4
----------------------
Number transactions: 3
----------------------
Miner reward to: 263f99c540f4168132cc5fee5726b5ce
Amount: 0.99
Hash: 3490eb317353dcc32715788714a13769
----------------------
From: 9412daed1e0bd204f652677a80192ea9
To: 8fca7b9ab6fac411b2443c1303d3bc56
Amount: 3
Date: Mon Feb 22 2021 13:44:36 GMT-0800 (PST)
Hash: bb281c938be6c2bee7ffc05bdb4f1e23
----------------------
From: bf61f561918f37cf441fb8a6a1094b7a
To: f157bab61715d2bca9d81ada45bad57c
Amount: 1
Date: Mon Feb 22 2021 13:44:29 GMT-0800 (PST)
Hash: 26a537138d436bbd55a7287edf152951
============================================
```

The Merkle root is placed in the block header.

The block header is hashed and the hash passed to the next block.

Any change in any transaction will be obvious in changes in the hash of the transaction >>

>> the Merkle root >>

>> the hash of the block header >>

>> the hash of all later blocks

The Merkle root is obtained from the transactions by a method that provides fast verification of individual transactions in the blockchain

Proof-of-work consensus method

- Any miner on the peer network can build a block by combining pending transactions
- which block is added to the chain is determined by a consensus method
- Teddy Token and Bitcoin use the "proof-of-work" method
- the protocol specifies a target number of zeros with which the final block header hash must start
- a miner must find the number - the nonce - to add to the header that achieves this
- the first miner who finds the correct nonce can add its block to the chain
- that miner receives a reward, which increases the number of coins in circulation

```
Version: TedTok_01
Block number: 2
Previous hash: 000f5b9d7ca44d91ba33e9934e82cf27
Merkle root: 2569e9f51001fdb295b836228bbfcc84
Date: Mon Feb 22 2021 13:44:40 GMT-0800 (PST)
Target: 3
Nonce: 856
Hash: 000869e6e37d5f11f5b71d2641c139d4
----------------------
Number transactions: 3
----------------------
Miner reward to: 263f99c540f4168132cc5fee5726b5ce
Amount: 0.99
Hash: 3490eb317353dcc32715788714a13769
----------------------
From: 9412daed1e0bd204f652677a80192ea9
To: 8fca7b9ab6fac411b2443c1303d3bc56
```

Lots of terms!
Hash
Address
Merkle root
Miner reward
Nonce
Target
Crypto key

nonce - number used once

One step in the MD2 cryptographic hash
(C array)

- "**Confusion**" by mixing message information with random numbers from the S-box
- "**Diffusion**" by value in current position in hash selecting S-box number for next message position such that confused information cascades and a change anywhere in message affects all locations by repeated passes through the message.

The message

T	e	d	d	⋯	m	o	t	o

text > UTF hexidecimal

54	65	64	64 hex	⋯	6d	6f	74	6f

info for step 4

binary values

0	1	1	0	0	1	0	0

XOR (bitwise exclusive OR)

1	0	1	0	0	1	0	1

=

1	1	0	0	0	0	0	1

Modern algorithms use the same concepts in more secure ways: random "round constants" in SHA-256, mixing by XOR & other operations, diffusion by bit shift.

info from step 3 for step 4

Substitution S-box (hexadecimal values)

29	2e	⋯	13
62	a7	⋯	ca
⋮	⋮	⋱	⋮
31	44	⋯	14

select value from element number hex c1 of S-box

value from element number hex c1 of S-box is hex ec

The hash being computed

		a5	ec	⋯				

info from step 4 for step 5

S-box is array of 256 elements, each containing a **randomly** placed, non-repeating value in decimal range 0-255

MD2 hash length is 32 hex = 16 each, 8-bit bytes = 128 bits, regardless of length of message, and maintained at this length by **MODULAR ARITHMETIC**

Modular arithmetic

Messages are first converted to large integers using text-to-code tables, e.g., A = 65

All variables represent integers. The integers a and b are congruent modulo n.

$$a \equiv b \pmod{n}$$

where "mod" in parenthesis means that the modulus operation applies to all parts of the expression. This can also be expressed as

$$a = b + kn$$

That is, a is divisible k times by n, with remainder b.

Those of us who use a 12 hour clock dial use modular arithmetic everyday. For example, the hour hand points to 6 on the dial at 6 am and 6 pm, where 6 pm is 18:00 in 24-hour time.

$$18 \equiv 6 \pmod{12}$$

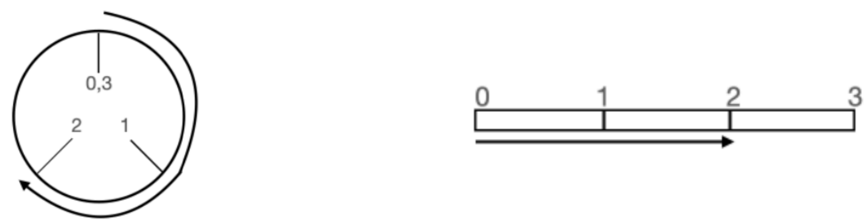
$$18 = 6 + 1(12)$$

Modular arithmetic

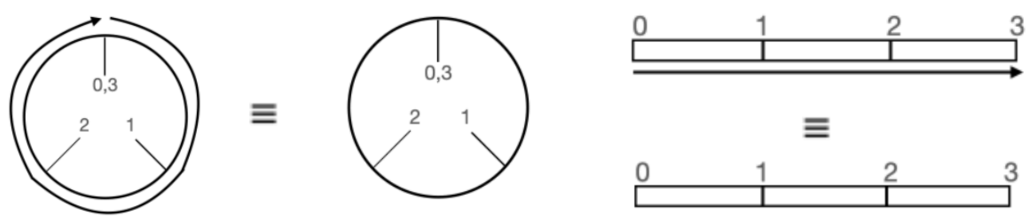
Cryptography and hashing use very large integers. For illustration here, we use small integers. For some operations below, we will use a dial or linear scale with three divisions to simplify things. The relationships shown work for numbers of all magnitudes.

$1 \equiv 1 \pmod 3$

$2 \equiv 2 \pmod 3$

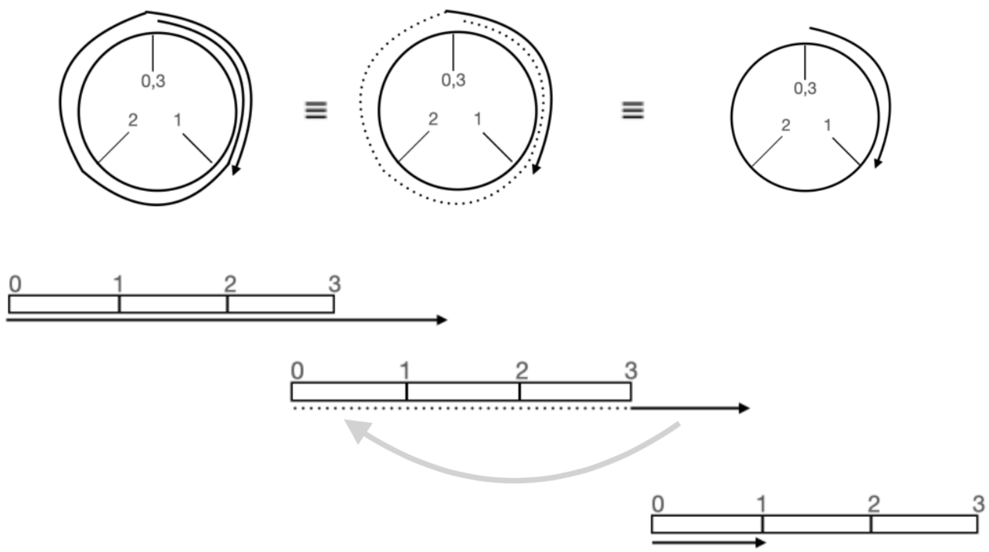


$3 \equiv 0 \pmod 3$



Modular arithmetic

$4 \equiv 1 \pmod 3$



One way to think of this is that numbers greater than some multiple k of the modulus n "wraps around" so that the maximum value in the system is n. In the examples above $k = 1$ but k may have any integer value.

When we discuss hash algorithms, you will see that this "wrap around" feature is what keeps hash values a constant length, regardless of the length of the message being hashed.

The MD2 hash "wraps around" at 16 bytes = 32 hex numerals

Addresses in blockchains are hashes of owners' public encryption keys

- one class of encryption is *symmetric* - use same key for both encryption and decryption
- disadvantage is that anyone who discovers the key can produce fake messages and secretly decrypt private messages

- the class that blockchains use is *asymmetric* encryption
- public key encryption is asymmetric
- a message is encrypted with the public key, which may be known to the public
- an encrypted message cannot be decrypted with the public key
- the encrypted message can only be decrypted with the owner's private key

Bitcoin uses Elliptic Curve Cryptography - Teddy Token uses RSA Cryptography

- both are asymmetric methods
- we discuss RSA cryptography in the following slides

RSA public-key encryption

RSA involves the generation and use of a public key and a private key.

The public key consists of two integers, n and e. The private key consists of two integers, the same value of n, and d. Encryption key integers have very large values.

A plain-text message is first converted to an integer, m, by converting the message to a string of the ASCII or Unicode values of the characters in the message.

Message $m < n$ is encrypted to the encoded message c with the public key by this equation:

$$c = m^e \text{ (mod } n)$$

The holder of the private key can decrypt c in order to recover m by this equation:

$$m = c^d \text{ (mod } n)$$

The primes p and q are the basis numbers of the RSA public-key encryption method. Their choice results in the generation of the public and private keys. They are kept secret.

The public key consists of two integers, n and e. The integer e is chosen to be coprime with $n = pq$. That is, $e < n$ and e will not evenly divide n.

The private key consists of two integers, n and d. The integer d is the "modular inverse" of e, such that

$$ed \equiv 1 \pmod{\phi(n)}$$

This can also be expressed as

$$ed = 1 + k\phi(n)$$

and ϕ is Euler's phi function

$$\phi(n) = \phi(pq) = \phi(p)\phi(q) = (p-1)(q-1)$$

Message $m < n$ is encrypted to the encoded message c by this equation:

$$m^e \equiv c \pmod{n}$$

The holder of the private key can decrypt c in order to recover m by this equation:

$$c^d \equiv m \pmod{n}$$

In actual RSA encryption, the values are large such that the computation of m^e cannot be done directly on a computer before the modulo operation, as can be done with small values such as 2^3. This is because the value of m^e that would be obtained with the large values of m and e that are used would be so large that it could not be contained by a computer's method of storing numbers.

The solution of this problem is to compute the results using the modular exponentiation algorithm. In that algorithm, the largest value that must be stored during computation is m^2.

Why is the private key secure & can't be determined from the public key?

- from the public key, n and e are known
- d has to be determined in order to get the private key
- d is the modular inverse of e: $e*d = 1 \pmod{\phi(n)}$
- to get d knowing e, one needs to get $\phi(n)$
- $\phi(n) = (p-1)*(q-1)$, where $n = p*q$
- this would require finding which two prime numbers - p and q - when multiplied together equals the known value n
- this is called "prime factorization"
- with the large values of p and q used in RSA, the prime factorization of n would take a ridiculously long time

See Web Labs at ReactorLab.net for interactive simulations of

- Teddy Token blockchain
- Cryptographic hash
- Public key encryption