Electronic Cash and Blockchain Security

Yongge Wang

UNC Charlotte, USA

October 15, 2018

Outline

1. Background
   - David Chaum
2. Bitcoin
   - BTC Transaction
   - Merkle Tree
   - BTC Transaction scripts
3. Ethereum and General Block Chain

Outline

Outline

1. Background
   - David Chaum
2. Bitcoin
   - BTC Transaction
   - Merkle Tree
   - BTC Transaction scripts
3. Ethereum and General Block Chain

Motivation

- Real cash could be anonymous though theoretically it is not (sequence numbers, but who record them?)
- Easy to design e-cash using PKI, but traceable
- e-cash or e-wallet is convenient for online small payment

Motivation

- Real cash could be anonymous though theoretically it is not (sequence numbers, but who record them?)
- Easy to design e-cash using PKI, but traceable
- e-cash or e-wallet is convenient for online small payment

Motivation

- Real cash could be anonymous though theoretically it is not (sequence numbers, but who record them?)
- Easy to design e-cash using PKI, but traceable
- e-cash or e-wallet is convenient for online small payment

Requirements for e-cash

- anonymous (non-traceable)
- no double spending
- easy to pay a few cents on line
- many others

Requirements for e-cash

- anonymous (non-traceable)
- no double spending
- easy to pay a few cents on line
- many others

Requirements for e-cash

- anonymous (non-traceable)
- no double spending
- easy to pay a few cents on line
- many others

Requirements for e-cash

- anonymous (non-traceable)
- no double spending
- easy to pay a few cents on line
- many others

Historical Efforts: David Chaum

- The concept of e-cash was originally based on Chaum's blind signature (1984)
- Untraceable Electronic Cash (Chaum, Fiat, Naor 1990)
- many others
- bitcoin 2009
- Ethereum 2015

Historical Efforts: David Chaum

- The concept of e-cash was originally based on Chaum's blind signature (1984)
- Untraceable Electronic Cash (Chaum, Fiat, Naor 1990)
- many others
- bitcoin 2009
- Ethereum 2015

Historical Efforts: David Chaum

- The concept of e-cash was originally based on Chaum's blind signature (1984)
- Untraceable Electronic Cash (Chaum, Fiat, Naor 1990)
- many others
- bitcoin 2009
- Ethereum 2015

Historical Efforts: David Chaum

- The concept of e-cash was originally based on Chaum's blind signature (1984)
- Untraceable Electronic Cash (Chaum, Fiat, Naor 1990)
- many others
- bitcoin 2009
- Ethereum 2015

Historical Efforts: David Chaum

- The concept of e-cash was originally based on Chaum's blind signature (1984)
- Untraceable Electronic Cash (Chaum, Fiat, Naor 1990)
- many others
- bitcoin 2009
- Ethereum 2015

Blind Signature (Chaum)

- the Bank has an RSA public key (e, N) and private key d
- Alice has a coin m (e.g., \$10)
- Alice chooses a random number r, and computes $m' = m \cdot r^e (\text{mod } N)$
- bank signs m' with signature $s' = (m')^d$
- Alice calculates signature s on m as

$$s = s' \cdot r^{-1} = (m \cdot r^e)^d \cdot r^{-1} = m^d$$

- Alice spends (m, s) as \$10 while bank cannot link this coin m to Alice's account

Blind Signature (Chaum)

- the Bank has an RSA public key (e, N) and private key d
- Alice has a coin m (e.g., \$10)
- Alice chooses a random number r, and computes $m' = m \cdot r^e (\text{mod } N)$
- bank signs m' with signature $s' = (m')^d$
- Alice calculates signature s on m as

$$s = s' \cdot r^{-1} = (m \cdot r^e)^d \cdot r^{-1} = m^d$$

- Alice spends (m, s) as \$10 while bank cannot link this coin m to Alice's account

Blind Signature (Chaum)

- the Bank has an RSA public key (e, N) and private key d
- Alice has a coin m (e.g., $10)
- Alice chooses a random number r, and computes $m' = m \cdot r^e (\mathrm{mod}\ N)$
- bank signs m' with signature $s' = (m')^d$
- Alice calculates signature s on m as

$$s = s' \cdot r^{-1} = (m \cdot r^e)^d \cdot r^{-1} = m^d$$

- Alice spends (m, s) as $10 while bank cannot link this coin m to Alice's account

Blind Signature (Chaum)

- the Bank has an RSA public key (e, N) and private key d
- Alice has a coin m (e.g., \$10)
- Alice chooses a random number r, and computes $m' = m \cdot r^e (\text{mod } N)$
- bank signs m' with signature $s' = (m')^d$
- Alice calculates signature s on m as

$$s = s' \cdot r^{-1} = (m \cdot r^e)^d \cdot r^{-1} = m^d$$

- Alice spends (m, s) as \$10 while bank cannot link this coin m to Alice's account

Blind Signature (Chaum)

- the Bank has an RSA public key (e, N) and private key d
- Alice has a coin m (e.g., $10)
- Alice chooses a random number r, and computes $m' = m \cdot r^e (\text{mod } N)$
- bank signs m' with signature $s' = (m')^d$
- Alice calculates signature s on m as

$$s = s' \cdot r^{-1} = (m \cdot r^e)^d \cdot r^{-1} = m^d$$

- Alice spends (m, s) as $10 while bank cannot link this coin m to Alice's account

Blind Signature (Chaum)

- the Bank has an RSA public key (e, N) and private key d
- Alice has a coin m (e.g., $10)
- Alice chooses a random number r, and computes $m' = m \cdot r^e (\text{mod } N)$
- bank signs m' with signature $s' = (m')^d$
- Alice calculates signature s on m as

$$s = s' \cdot r^{-1} = (m \cdot r^e)^d \cdot r^{-1} = m^d$$

- Alice spends (m, s) as $10 while bank cannot link this coin m to Alice's account

Challenges in Blind Signature Scheme

- What happens if $m = 100\$$ instead of 10$ unless all coins have same value?
- Seller must contact bank to make sure m has not been spent yet when accepting the money from Alice
- can we remove the online restrict? In other words, seller does not need to contact bank: Chaum, Fiat, and Naor Scheme (1988)

Challenges in Blind Signature Scheme

- What happens if $m = 100\$$ instead of 10$ unless all coins have same value?
- Seller must contact bank to make sure m has not been spent yet when accepting the money from Alice
- can we remove the online restrict? In other words, seller does not need to contact bank: Chaum, Fiat, and Naor Scheme (1988)

Challenges in Blind Signature Scheme

- What happens if $m = 100\$$ instead of 10$ unless all coins have same value?
- Seller must contact bank to make sure m has not been spent yet when accepting the money from Alice
- can we remove the online restrict? In other words, seller does not need to contact bank: Chaum, Fiat, and Naor Scheme (1988)

Bitcoin: a high level description

- A pseudonym "Satoshi Nakamoto" designed BTC in 2008 and in operation since 2009, http://bitcoin.org/bitcoin.pdf
- w_0 is the start coinbase by Satoshi Nakamoto
- you find a random number r_0 such that $H(w_0, r_0) = w_1$ such that the first two bits of w_1 is 00, you will be rewarded with one BTC
- Another person will mint BTC by finding another r_1 with $H(w_1, r_1) = w_2$ such that the first two bits of w_2 is 00, you will be rewarded with one BTC
- this process continues until computer becomes fast and you have to find a random r_i such that the hash output contains a long prefix of 0
- transactions are included in the hash in order to be verified

Bitcoin: a high level description

- A pseudonym "Satoshi Nakamoto" designed BTC in 2008 and in operation since 2009, http://bitcoin.org/bitcoin.pdf
- w_0 is the start coinbase by Satoshi Nakamoto
- you find a random number r_0 such that $H(w_0, r_0) = w_1$ such that the first two bits of w_1 is 00, you will be rewarded with one BTC
- Another person will mint BTC by finding another r_1 with $H(w_1, r_1) = w_2$ such that the first two bits of w_2 is 00, you will be rewarded with one BTC
- this process continues until computer becomes fast and you have to find a random r_i such that the hash output contains a long prefix of 0
- transactions are included in the hash in order to be verified

Bitcoin: a high level description

- A pseudonym "Satoshi Nakamoto" designed BTC in 2008 and in operation since 2009, http://bitcoin.org/bitcoin.pdf
- w_0 is the start coinbase by Satoshi Nakamoto
- you find a random number r_0 such that $H(w_0, r_0) = w_1$ such that the first two bits of w_1 is 00, you will be rewarded with one BTC
- Another person will mint BTC by finding another r_1 with $H(w_1, r_1) = w_2$ such that the first two bits of w_2 is 00, you will be rewarded with one BTC
- this process continues until computer becomes fast and you have to find a random r_i such that the hash output contains a long prefix of 0
- transactions are included in the hash in order to be verified

Bitcoin: a high level description

- A pseudonym "Satoshi Nakamoto" designed BTC in 2008 and in operation since 2009, http://bitcoin.org/bitcoin.pdf
- w_0 is the start coinbase by Satoshi Nakamoto
- you find a random number r_0 such that $H(w_0, r_0) = w_1$ such that the first two bits of w_1 is 00, you will be rewarded with one BTC
- Another person will mint BTC by finding another r_1 with $H(w_1, r_1) = w_2$ such that the first two bits of w_2 is 00, you will be rewarded with one BTC
- this process continues until computer becomes fast and you have to find a random r_i such that the hash output contains a long prefix of 0
- transactions are included in the hash in order to be verified

Bitcoin: a high level description

- A pseudonym "Satoshi Nakamoto" designed BTC in 2008 and in operation since 2009, http://bitcoin.org/bitcoin.pdf
- w_0 is the start coinbase by Satoshi Nakamoto
- you find a random number r_0 such that $H(w_0, r_0) = w_1$ such that the first two bits of w_1 is 00, you will be rewarded with one BTC
- Another person will mint BTC by finding another r_1 with $H(w_1, r_1) = w_2$ such that the first two bits of w_2 is 00, you will be rewarded with one BTC
- this process continues until computer becomes fast and you have to find a random r_i such that the hash output contains a long prefix of 0
- transactions are included in the hash in order to be verified

Bitcoin: a high level description

- A pseudonym "Satoshi Nakamoto" designed BTC in 2008 and in operation since 2009, http://bitcoin.org/bitcoin.pdf
- w_0 is the start coinbase by Satoshi Nakamoto
- you find a random number r_0 such that $H(w_0, r_0) = w_1$ such that the first two bits of w_1 is 00, you will be rewarded with one BTC
- Another person will mint BTC by finding another r_1 with $H(w_1, r_1) = w_2$ such that the first two bits of w_2 is 00, you will be rewarded with one BTC
- this process continues until computer becomes fast and you have to find a random r_i such that the hash output contains a long prefix of 0
- transactions are included in the hash in order to be verified

Bitcoin with transaction

- the BTC is a chain $w_0, w_1, \cdots, w_n$ where w_n is the current BTC HEAD that everyone works on it
- based on P2P protocol, all person work on the longest chain. If you work on a shorter chain, you waste time and the transaction included in these chains will not be valid
- w_n has prefix of 0...0 where the number of 0 is determined by voting algorithm so one BTC is minted each 10 minutes
- $w_{i+1} = H(w_i, TR, r_i)$ where TR is the Merkle hash output of the transactions that you want to include and r_i is a random number that you find to make w_{i+1} has a certain number 0's in its prefix

Bitcoin with transaction

- the BTC is a chain $w_0, w_1, \cdots, w_n$ where w_n is the current BTC HEAD that everyone works on it
- based on P2P protocol, all person work on the longest chain. If you work on a shorter chain, you waste time and the transaction included in these chains will not be valid
- w_n has prefix of 0...0 where the number of 0 is determined by voting algorithm so one BTC is minted each 10 minutes
- $w_{i+1} = H(w_i, TR, r_i)$ where TR is the Merkle hash output of the transactions that you want to include and r_i is a random number that you find to make w_{i+1} has a certain number 0's in its prefix

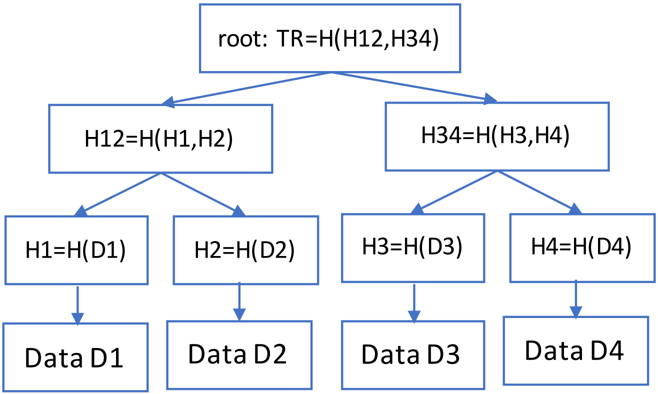
Bitcoin with transaction

- the BTC is a chain $w_0, w_1, \cdots, w_n$ where w_n is the current BTC HEAD that everyone works on it
- based on P2P protocol, all person work on the longest chain. If you work on a shorter chain, you waste time and the transaction included in these chains will not be valid
- w_n has prefix of 0...0 where the number of 0 is determined by voting algorithm so one BTC is minted each 10 minutes
- $w_{i+1} = H(w_i, TR, r_i)$ where TR is the Merkle hash output of the transactions that you want to include and r_i is a random number that you find to make w_{i+1} has a certain number 0's in its prefix

Bitcoin with transaction

- the BTC is a chain $w_0, w_1, \cdots, w_n$ where w_n is the current BTC HEAD that everyone works on it
- based on P2P protocol, all person work on the longest chain. If you work on a shorter chain, you waste time and the transaction included in these chains will not be valid
- w_n has prefix of 0...0 where the number of 0 is determined by voting algorithm so one BTC is minted each 10 minutes
- $w_{i+1} = H(w_i, TR, r_i)$ where TR is the Merkle hash output of the transactions that you want to include and r_i is a random number that you find to make w_{i+1} has a certain number 0's in its prefix

Merkle Hash Tree



Bitcoin Transaction Scripts

- BTC transactions are described using Forth-like Scripts (https://en.bitcoin.it/wiki/Script)
- the scripts enable smart contract (e.g., the transaction will be valid if two persons sign the contract, valid after certain time etc.)
- A transaction means Alice pays x BTC to Bob
- This is achieved by Alice signing the message "reference number, Bob's pub key, BTC amount"
- "reference number" should be contained in some block of the current BTC chain $w_0, w_1, \cdots, w_n$. E.g., w_i
- Alice's public key should be included in the block w_i transaction with the given reference number

Bitcoin Transaction Scripts

- BTC transactions are described using Forth-like Scripts (https://en.bitcoin.it/wiki/Script)
- the scripts enable smart contract (e.g., the transaction will be valid if two persons sign the contract, valid after certain time etc.)
- A transaction means Alice pays x BTC to Bob
- This is achieved by Alice signing the message “reference number, Bob’s pub key, BTC amount”
- “reference number” should be contained in some block of the current BTC chain $w_0, w_1, \cdots, w_n$. E.g., w_i
- Alice’s public key should be included in the block w_i transaction with the given reference number

Bitcoin Transaction Scripts

- BTC transactions are described using Forth-like Scripts (https://en.bitcoin.it/wiki/Script)
- the scripts enable smart contract (e.g., the transaction will be valid if two persons sign the contract, valid after certain time etc.)
- A transaction means Alice pays x BTC to Bob
- This is achieved by Alice signing the message “reference number, Bob’s pub key, BTC amount”
- “reference number” should be contained in some block of the current BTC chain $w_0, w_1, \cdots, w_n$. E.g., w_i
- Alice’s public key should be included in the block w_i transaction with the given reference number

Bitcoin Transaction Scripts

- BTC transactions are described using Forth-like Scripts (https://en.bitcoin.it/wiki/Script)
- the scripts enable smart contract (e.g., the transaction will be valid if two persons sign the contract, valid after certain time etc.)
- A transaction means Alice pays x BTC to Bob
- This is achieved by Alice signing the message "reference number, Bob's pub key, BTC amount"
- "reference number" should be contained in some block of the current BTC chain $w_0, w_1, \cdots, w_n$. E.g., w_i
- Alice's public key should be included in the block w_i transaction with the given reference number

Bitcoin Transaction Scripts

- BTC transactions are described using Forth-like Scripts (https://en.bitcoin.it/wiki/Script)
- the scripts enable smart contract (e.g., the transaction will be valid if two persons sign the contract, valid after certain time etc.)
- A transaction means Alice pays x BTC to Bob
- This is achieved by Alice signing the message "reference number, Bob's pub key, BTC amount"
- "reference number" should be contained in some block of the current BTC chain $w_0, w_1, \cdots, w_n$. E.g., w_i
- Alice's public key should be included in the block w_i transaction with the given reference number

Bitcoin Transaction Scripts

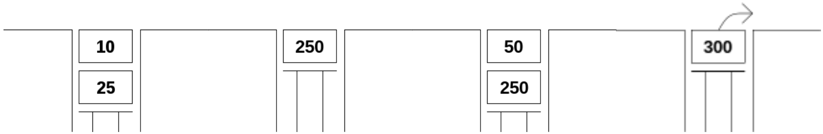
- BTC transactions are described using Forth-like Scripts (https://en.bitcoin.it/wiki/Script)
- the scripts enable smart contract (e.g., the transaction will be valid if two persons sign the contract, valid after certain time etc.)
- A transaction means Alice pays x BTC to Bob
- This is achieved by Alice signing the message "reference number, Bob's pub key, BTC amount"
- "reference number" should be contained in some block of the current BTC chain $w_0, w_1, \cdots, w_n$. E.g., w_i
- Alice's public key should be included in the block w_i transaction with the given reference number

Example Forth Script

- In order to compute $25 \times 10 + 50$, we inputs: 25 10 * 50 + . in the calculator
- It works the the following way by stack

Example Forth Script

- In order to compute $25 \times 10 + 50$, we inputs: 25 10 * 50 + . in the calculator
- It works the the following way by stack



Example Transaction

- scriptPubKey: OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG
- scriptSig: <sig> <pubKey>

Stack	Script	Description
Empty.	<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	scriptSig and scriptPubKey are combined.
<sig> <pubKey>	OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	Constants are added to the stack.
<sig> <pubKey> <pubKey>	OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	Top stack item is duplicated.
<sig> <pubKey> <pubHashA>	<pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	Top stack item is hashed.
<sig> <pubKey> <pubHashA> <pubKeyHash>	OP_EQUALVERIFY OP_CHECKSIG	Constant added.
<sig> <pubKey>	OP_CHECKSIG	Equality is checked between the top two stack items.
true	Empty.	Signature is checked for top two stack items.

Example Transaction

- scriptPubKey: OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG
- scriptSig: <sig> <pubKey>

Stack	Script	Description
Empty.	<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	scriptSig and scriptPubKey are combined.
<sig> <pubKey>	OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	Constants are added to the stack.
<sig> <pubKey> <pubKey>	OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	Top stack item is duplicated.
<sig> <pubKey> <pubHashA>	<pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	Top stack item is hashed.
<sig> <pubKey> <pubHashA> <pubKeyHash>	OP_EQUALVERIFY OP_CHECKSIG	Constant added.
<sig> <pubKey>	OP_CHECKSIG	Equality is checked between the top two stack items.
true	Empty.	Signature is checked for top two stack items.

Example Transaction 2

```
scriptPubKey: <pubKey> OP_CHECKSIG
scriptSig: <sig>
```

Checking process:

Stack	Script	Description
Empty.	<sig> <pubKey> OP_CHECKSIG	scriptSig and scriptPubKey are combined.
<sig> <pubKey>	OP_CHECKSIG	Constants are added to the stack.
true	Empty.	Signature is checked for top two stack items.

Internet Service Platform

- Anybody can upload programs to the Ethereum World Computer and anybody can request that a program that has been uploaded be executed.

What is New in Ethereum

- BTC scripting language has limited capability while Ethereum script is Turing complete
- Ethereum is a blockchain with a built-in Turing-complete programming language, allowing anyone to write smart contracts and decentralized applications where they can create their own arbitrary rules for ownership, transaction formats and state transition functions.
- BTC only supports “Proof of work” while Ethereum also supports “proof of stake”
- Proof of stake: calculating the weight of a node as being proportional to its currency holdings and not its computational resources.

What is New in Ethereum

- BTC scripting language has limited capability while Ethereum script is Turing complete
- Ethereum is a blockchain with a built-in Turing-complete programming language, allowing anyone to write smart contracts and decentralized applications where they can create their own arbitrary rules for ownership, transaction formats and state transition functions.
- BTC only supports “Proof of work” while Ethereum also supports “proof of stake”
- Proof of stake: calculating the weight of a node as being proportional to its currency holdings and not its computational resources.

What is New in Ethereum

- BTC scripting language has limited capability while Ethereum script is Turing complete
- Ethereum is a blockchain with a built-in Turing-complete programming language, allowing anyone to write smart contracts and decentralized applications where they can create their own arbitrary rules for ownership, transaction formats and state transition functions.
- BTC only supports “Proof of work” while Ethereum also supports “proof of stake”
- Proof of stake: calculating the weight of a node as being proportional to its currency holdings and not its computational resources.

What is New in Ethereum

- BTC scripting language has limited capability while Ethereum script is Turing complete
- Ethereum is a blockchain with a built-in Turing-complete programming language, allowing anyone to write smart contracts and decentralized applications where they can create their own arbitrary rules for ownership, transaction formats and state transition functions.
- BTC only supports “Proof of work” while Ethereum also supports “proof of stake”
- Proof of stake: calculating the weight of a node as being proportional to its currency holdings and not its computational resources.

Ethereum

- Ethereum allows users to create their own operations of any complexity they wish (Turing Complete)
- Based on the Ethereum Virtual Machine (EVM): the runtime environment for smart contracts in Ethereum.

Ethereum

- Ethereum allows users to create their own operations of any complexity they wish (Turing Complete)
- Based on the Ethereum Virtual Machine (EVM): the runtime environment for smart contracts in Ethereum.

Ethereum Accounts and Smart Contracts

- Accounts: 20 bytes string.
- An account contains four fields: nonce, ether balance, contract code (optional), and storage (empty by default)
- Externally Owned Accounts (EOAs), which are controlled by private keys
- Contract Accounts, which are controlled by their contract code and can only be "activated" by an EOA. Contract accounts are governed by their internal code which is programmed to be controlled by an EOA with a certain address,
- "smart contracts" refers to code in a Contract Account: programs that execute when a transaction is sent to that account.
- Users can create new contracts by deploying code to the blockchain.

Ethereum Accounts and Smart Contracts

- Accounts: 20 bytes string.
- An account contains four fields: nonce, ether balance, contract code (optional), and storage (empty by default)
- Externally Owned Accounts (EOAs), which are controlled by private keys
- Contract Accounts, which are controlled by their contract code and can only be "activated" by an EOA. Contract accounts are governed by their internal code which is programmed to be controlled by an EOA with a certain address,
- "smart contracts" refers to code in a Contract Account: programs that execute when a transaction is sent to that account.
- Users can create new contracts by deploying code to the blockchain.

Ethereum Accounts and Smart Contracts

- Accounts: 20 bytes string.
- An account contains four fields: nonce, ether balance, contract code (optional), and storage (empty by default)
- Externally Owned Accounts (EOAs), which are controlled by private keys
- Contract Accounts, which are controlled by their contract code and can only be "activated" by an EOA. Contract accounts are governed by their internal code which is programmed to be controlled by an EOA with a certain address,
- "smart contracts" refers to code in a Contract Account: programs that execute when a transaction is sent to that account.
- Users can create new contracts by deploying code to the blockchain.

Ethereum Accounts and Smart Contracts

- Accounts: 20 bytes string.
- An account contains four fields: nonce, ether balance, contract code (optional), and storage (empty by default)
- Externally Owned Accounts (EOAs), which are controlled by private keys
- Contract Accounts, which are controlled by their contract code and can only be "activated" by an EOA. Contract accounts are governed by their internal code which is programmed to be controlled by an EOA with a certain address,
- "smart contracts" refers to code in a Contract Account: programs that execute when a transaction is sent to that account.
- Users can create new contracts by deploying code to the blockchain.

Ethereum Accounts and Smart Contracts

- Accounts: 20 bytes string.
- An account contains four fields: nonce, ether balance, contract code (optional), and storage (empty by default)
- Externally Owned Accounts (EOAs), which are controlled by private keys
- Contract Accounts, which are controlled by their contract code and can only be "activated" by an EOA. Contract accounts are governed by their internal code which is programmed to be controlled by an EOA with a certain address,
- "smart contracts" refers to code in a Contract Account: programs that execute when a transaction is sent to that account.
- Users can create new contracts by deploying code to the blockchain.

Ethereum Accounts and Smart Contracts

- Accounts: 20 bytes string.
- An account contains four fields: nonce, ether balance, contract code (optional), and storage (empty by default)
- Externally Owned Accounts (EOAs), which are controlled by private keys
- Contract Accounts, which are controlled by their contract code and can only be "activated" by an EOA. Contract accounts are governed by their internal code which is programmed to be controlled by an EOA with a certain address,
- "smart contracts" refers to code in a Contract Account: programs that execute when a transaction is sent to that account.
- Users can create new contracts by deploying code to the blockchain.

Ethereum Accounts and Smart Contracts

- Accounts: 20 bytes string.
- An account contains four fields: nonce, ether balance, contract code (optional), and storage (empty by default)
- Externally Owned Accounts (EOAs), which are controlled by private keys
- Contract Accounts, which are controlled by their contract code and can only be "activated" by an EOA. Contract accounts are governed by their internal code which is programmed to be controlled by an EOA with a certain address,
- "smart contracts" refers to code in a Contract Account: programs that execute when a transaction is sent to that account.
- Users can create new contracts by deploying code to the blockchain.

Digital Economy and Smart Contracts

Obama-Trump Contract: *Donald Trump releases his tax return forms as soon as Barack Obama releases his birth certificate*.
How can we design block-chain based Obama-Trump Contract?

- Important issue: privacy does not have a price tag
- How can we deal with contract without deposit?

Digital Economy and Smart Contracts

Obama-Trump Contract: *Donald Trump releases his tax return forms as soon as Barack Obama releases his birth certificate*.
How can we design block-chain based Obama-Trump Contract?

- Important issue: privacy does not have a price tag
- How can we deal with contract without deposit?

Obama-Trump Contract

- Yongge Wang, The Limit of Blockchains: Infeasibility of a Smart Obama-Trump Contract: To appear in *The Communications of the ACM* next month

Legal, Forensic, and Social Impact of Blockchains

Blockchains have become a buzzword and it is believed that smart contract is a panacea to redefine the digital economy. We initiated the study in this direction and investigates the potential legal, forensic, and social impact of blockchains on the society.

- The proof-of-work (or hybrid proof-of-work/proof-of-stake systems) based blockchains may pose serious challenges to both forms of government: dictatorships and constitutional democracies.
- It is predicted that most countries will ban proof-of-work (or hybrid proof-of- work/proof-of-stake systems) based blockchains in future.
- if proof-of-stake based blockchains are appropriately designed, then one could avoid these challenges.

Legal, Forensic, and Social Impact of Blockchains

Blockchains have become a buzzword and it is believed that smart contract is a panacea to redefine the digital economy. We initiated the study in this direction and investigates the potential legal, forensic, and social impact of blockchains on the society.

- The proof-of-work (or hybrid proof-of-work/proof-of-stake systems) based blockchains may pose serious challenges to both forms of government: dictatorships and constitutional democracies.
- It is predicted that most countries will ban proof-of-work (or hybrid proof-of- work/proof-of-stake systems) based blockchains in future.
- if proof-of-stake based blockchains are appropriately designed, then one could avoid these challenges.

Legal, Forensic, and Social Impact of Blockchains

Blockchains have become a buzzword and it is believed that smart contract is a panacea to redefine the digital economy. We initiated the study in this direction and investigates the potential legal, forensic, and social impact of blockchains on the society.

- The proof-of-work (or hybrid proof-of-work/proof-of-stake systems) based blockchains may pose serious challenges to both forms of government: dictatorships and constitutional democracies.
- It is predicted that most countries will ban proof-of-work (or hybrid proof-of- work/proof-of-stake systems) based blockchains in future.
- if proof-of-stake based blockchains are appropriately designed, then one could avoid these challenges.

Poisoning Attack against Mining Pools
M.Ahmed, J.Wei, Y.Wang, and E.Al-Shaer

- Attacks on crypto-currency mining pools
- Deliberately introducing errors under benign miners' names, this attack can fool the mining pool administrator into punishing innocent miner;
- when the top miners are punished, this attack can significantly slow down the overall production of the mining pool.
- An attacker needs only a small fraction (e.g, one millionth) of the resources of a victim mining pool,
- We confirm the effectiveness of this attack schem against well-known mining pools such as Minergate and Slush Pool.

Poisoning Attack against Mining Pools

M.Ahmed, J.Wei, Y.Wang, and E.Al-Shaer

- Attacks on crypto-currency mining pools
- Deliberately introducing errors under benign miners' names, this attack can fool the mining pool administrator into punishing innocent miner;
- when the top miners are punished, this attack can significantly slow down the overall production of the mining pool.
- An attacker needs only a small fraction (e.g, one millionth) of the resources of a victim mining pool,
- We confirm the effectiveness of this attack schem against well-known mining pools such as Minergate and Slush Pool.

Poisoning Attack against Mining Pools

M.Ahmed, J.Wei, Y.Wang, and E.Al-Shaer

- Attacks on crypto-currency mining pools
- Deliberately introducing errors under benign miners' names, this attack can fool the mining pool administrator into punishing innocent miner;
- when the top miners are punished, this attack can significantly slow down the overall production of the mining pool.
- An attacker needs only a small fraction (e.g, one millionth) of the resources of a victim mining pool,
- We confirm the effectiveness of this attack schem against well-known mining pools such as Minergate and Slush Pool.

Poisoning Attack against Mining Pools

M.Ahmed, J.Wei, Y.Wang, and E.Al-Shaer

- Attacks on crypto-currency mining pools
- Deliberately introducing errors under benign miners' names, this attack can fool the mining pool administrator into punishing innocent miner;
- when the top miners are punished, this attack can significantly slow down the overall production of the mining pool.
- An attacker needs only a small fraction (e.g, one millionth) of the resources of a victim mining pool,
- We confirm the effectiveness of this attack schem against well-known mining pools such as Minergate and Slush Pool.

Poisoning Attack against Mining Pools

M.Ahmed, J.Wei, Y.Wang, and E.Al-Shaer

- Attacks on crypto-currency mining pools
- Deliberately introducing errors under benign miners' names, this attack can fool the mining pool administrator into punishing innocent miner;
- when the top miners are punished, this attack can significantly slow down the overall production of the mining pool.
- An attacker needs only a small fraction (e.g, one millionth) of the resources of a victim mining pool,
- We confirm the effectiveness of this attack schem against well-known mining pools such as Minergate and Slush Pool.

Cryptic Labs http://crypticlabs.org

We are building a unique combination of illustrious cryptography and security advisors, researchers and outstanding blockchain practitioners to work on decentralized and distributed trust. By combining cryptography and related security researchers with blockchain practitioners and startups, we have the opportunity to perform a great service to the business community and the world in general.

Q&A

Q&A?