Introduction to public-key cryptography

Part 2: applications of public-key cryptography

Jean-Sébastien Coron

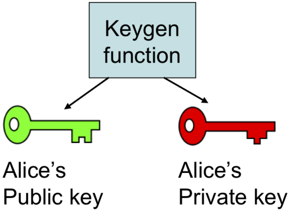
University of Luxembourg

Outline

- Part 1: basic constructions (previous lecture)
  - History
  - Classical cryptography: block-ciphers, hash functions
  - Public-key cryptography: RSA encryption and RSA signatures, DH key exchange
- Part 2: applications of public-key cryptography (this lecture)
  - Security models
  - How to encrypt and sign securely with RSA. OAEP and PSS.
  - Public-key infrastructure. Certificates, HTTPS protocol.
  - Bitcoin and the cryptographic blockchain

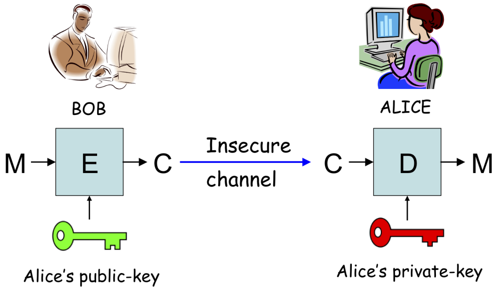
Public-key cryptography

- Invented by Diffie and Hellman in 1976. Revolutionized the field.
- Each user now has two keys
  - A public key
  - A private key
  - Should be hard to compute the private key from the public key.
- Enables:
  - Asymmetric encryption
  - Digital signatures
  - Key exchange, identification, and many other protocols.



Public-key encryption

- Public-key encryption (or asymmetric encryption)
  - Solves the key distribution issue

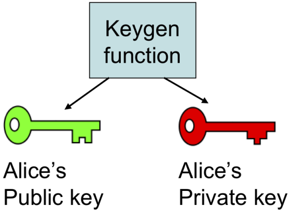


RSA

- Key generation:
  - Generate two large distinct primes p and q of same bit-size $k/2$, where k is a parameter.
  - Compute $n = p \cdot q$ and $\phi = (p-1)(q-1)$.
  - Select a random integer e such that $\gcd(e, \phi) = 1$
  - Compute the unique integer d such that

$$e \cdot d \equiv 1 \pmod{\phi}$$

    using the extended Euclidean algorithm.
  - The public key is (n, e).
  - The private key is d.



RSA encryption

- Encryption with public-key (n, e)
  - Given a message $m \in [0, n-1]$ and the recipent's public-key (n, e), compute the ciphertext:

$$c = m^e \bmod n$$

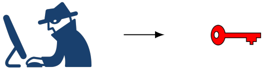
- Decryption with private-key d
  - Given a ciphertext c, to recover m, compute:

$$m = c^d \bmod n$$

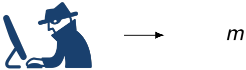
- Security is based on the hardness of factorization
  - Given $n = p \cdot q$, no known efficient algorithm to recover the primes p and q.
  - Public modulus n must be large enough: at least 1024 bits. 2048 bits is better.

Security models

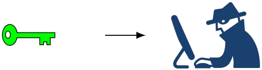
- To be rigorous when speaking about security, one must specify: the attacker's goal and the attacker's power.
- The attacker's goal
  - Does he need to recover the private key ?



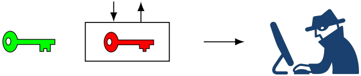
  - or only decrypt a particular ciphertext (or less) ?



- The attacker's power
  - Does he get only the user's public-key ?

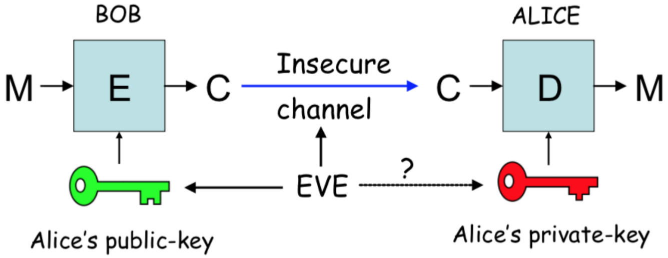


  - or more ?



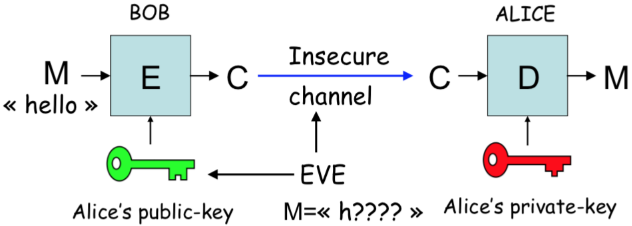
Attacker's goal

- One may think that the adversary's goal is always to recover the private key.
  - complete break
  - may be too ambitious in practice



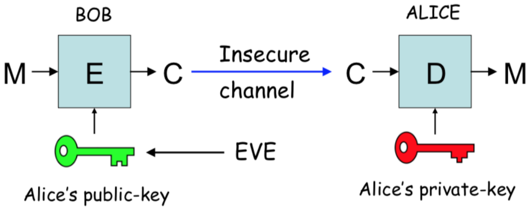
Attacker's goal

- More modest goal: being able to decrypt one ciphertext.
  - or recover some information about a plaintext (for example, the first character)



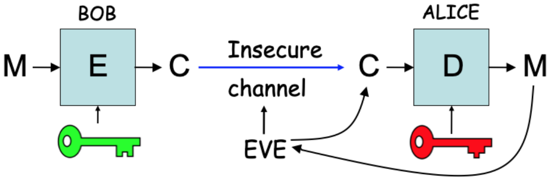
Attack model

- Specify the power of the attacker
- Public-key only attack: the attacker gets only the public-key
  - Called chosen plaintext attack (CPA) because the adversary can encrypt any plaintext of his choice.
  - Weakest adversary



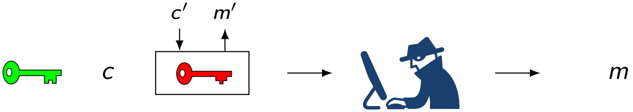
Chosen ciphertext attack (CCA)

- Most powerful attack
- The attacker can obtain decryption of messages of his choice
- May be realistic in practice
  - attacker gets access to a decryption machine
  - encryption algorithm used in a more complex protocol in which users can obtain decryption of chosen ciphertexts.



Chosen ciphertext attack against textbook RSA

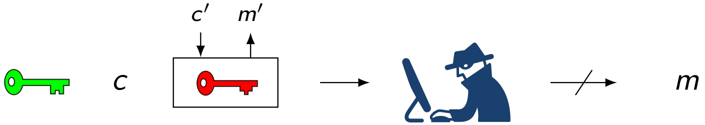
- Chosen-ciphertext attack:
  - Given ciphertext c to be decrypted
  - Generate a random r
  - Ask for the decryption of the random looking ciphertext $c' = c \cdot r^e \pmod{n}$
  - One gets $m' = (c')^d = c^d \cdot (r^e)^d = c^d \cdot r = m \cdot r \pmod{n}$
  - This enables to compute $m = m'/r \pmod{n}$



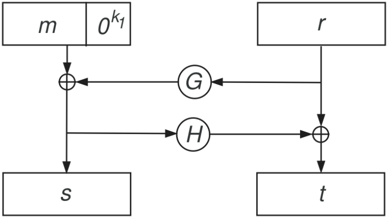
- Conclusion: do not use textbook RSA encryption !

Strongest security notion for public-key encryption

- Indistinguishability under adaptive chosen ciphertext attack (IND-CCA2)
  - Formalized in 1991 by Rackoff et Simon
  - A ciphertext should give no information about the corresponding plaintext, even under an adaptive chosen-ciphertext attack.
  - Has become standard security notion for encryption.



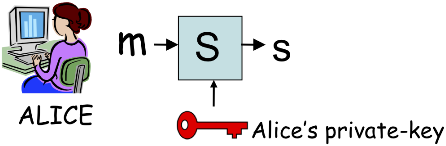
- OAEP (Bellare and Rogaway, E'94)
  - IND-CCA2, assuming that RSA is hard to invert.
  - PKCS #1 v2.1



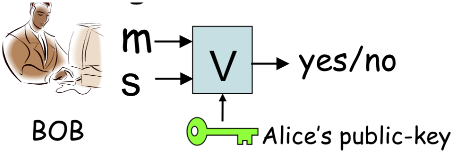
$$c = (s \| t)^e \bmod N$$

Digital signatures

- A digital signature σ is a bit string that depends on the message m and the user's public-key pk
  - Only Alice can sign a message m using her private-key sk



  - Anybody can verify Alice's signature of the message m given her public-key pk



The RSA signature scheme

- Key generation :
  - Public modulus: $N = p \cdot q$ where p and q are large primes.
  - Public exponent : e
  - Private exponent: d, such that $d \cdot e = 1 \pmod{\phi(N)}$
- To sign a message m, the signer computes :
  - $s = m^d \bmod N$
  - Only the signer can sign the message.
- To verify the signature, one checks that:
  - $m = s^e \bmod N$
  - Anybody can verify the signature

Attacks against textbook RSA signatures

- Given $\sigma_1 = (m_1)^d \bmod N$ and $\sigma_2 = (m_2)^d \bmod N$:
  - one can compute the signature of $m_1 \cdot m_2$ without knowing d:

$$\sigma = (m_1 \cdot m_2)^d = (m_1)^d \cdot (m_2)^d = \sigma_1 \cdot \sigma_2 \pmod N$$

- One cannot use plain RSA signature
  - Hash-and-sign paradigm: the message is first hashed

$$m \longrightarrow H(m) \longrightarrow 1001\ldots0101\|H(m)$$

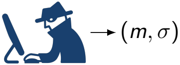
$$\downarrow$$

$$\sigma = \left(1001\ldots0101\|H(m)\right)^d \pmod N$$

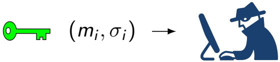
Attack scenario for signature schemes

- We must specify the adversary's goal and the adversary's power.
- Adversary's goal
  - Controlled forgery: the adversary can produce the signature of any message

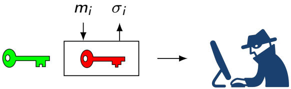
  - Existential forgery: the adversary can produce the signature of a (possibly meaningless) message



- Adversary's power
  - Known message attack: the adversary obtains a set of pairs message/signature

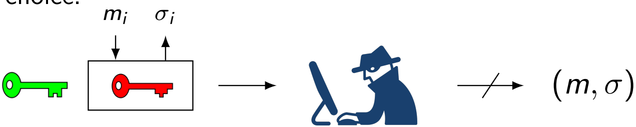


  - Chosen message attack: the adversary can obtain the signature of any message of his choice, adaptively.



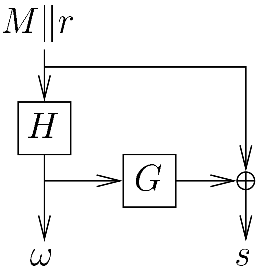
Strongest security notion for signature scheme

- Combines weakest goal with strongest adversary
- Existential unforgeability under an adaptive chosen message attack
  - Defined by Goldwasser, Micali and Rivest in 1988
  - It must be infeasible for an attacker to forge the signature of a message, even if he can obtain signatures of messages of his choice.



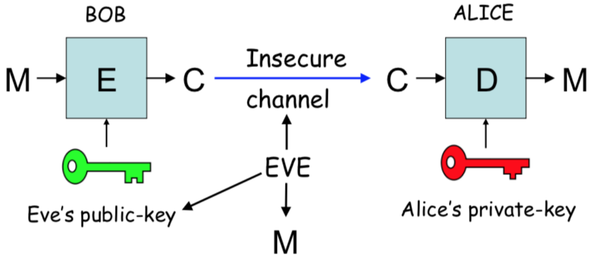
The PSS signature scheme

- PSS (Bellare and Rogaway, Eurocrypt'96)
  - IEEE P1363a and PKCS#1 v2.1.
  - 2 variants: PSS and PSS-R (message recovery)
  - Provably secure against chosen-message attacks, in the random oracle model.
  - PSS-R: $\mu(M,r) = \omega \| s$, $\sigma = \mu(M,r)^d \bmod N$



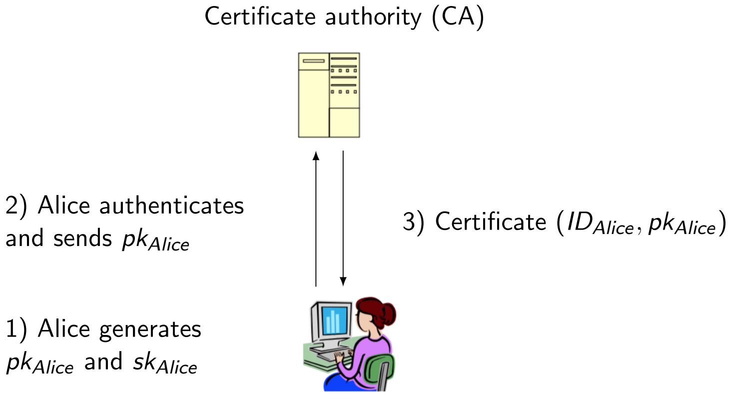
Public-key infrastructure

- Public-keys need to be authenticated
  - Bob needs to be sure that the public-key belongs to Alice.
  - Otherwise, impersonation attack



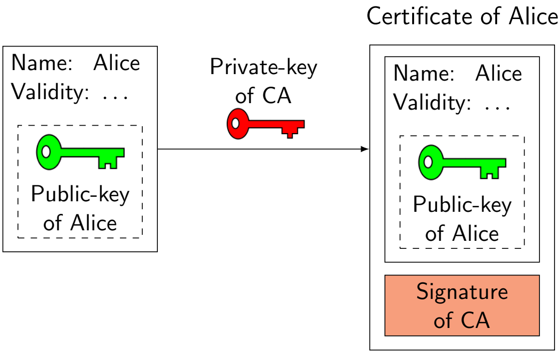
Public-key Infrastructure (PKI)

- A central authority binds public-keys to identities.
  - Public-key is stored in a certificate provided by the central authority
  - Used to prevent impersonation attack

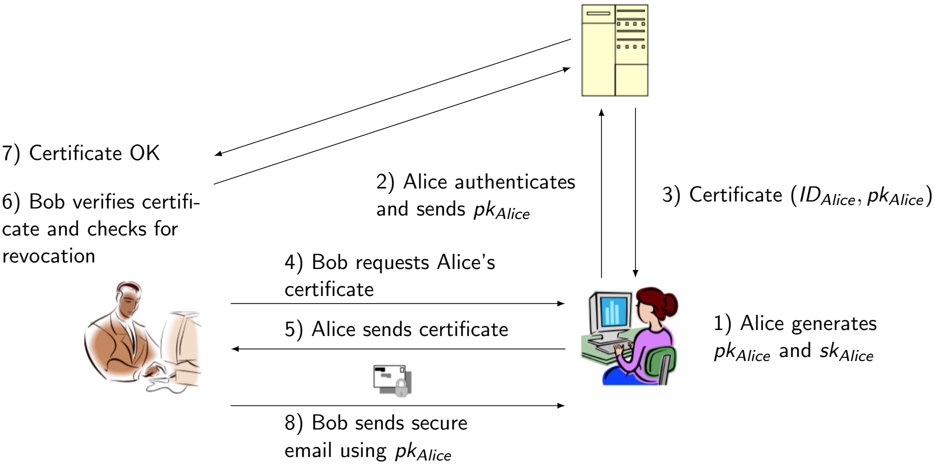


Public-key certificate

- The signature of the certificate authority (CA) binds together a public-key with an identity in the certificate.
    - Proves ownership of a public-key
    - Bob can be sure that the public-key belongs to Alice by checking the signature using the CA public-key.
    - The CA is trusted by all participants.
    - Most common certificate standard: X.509

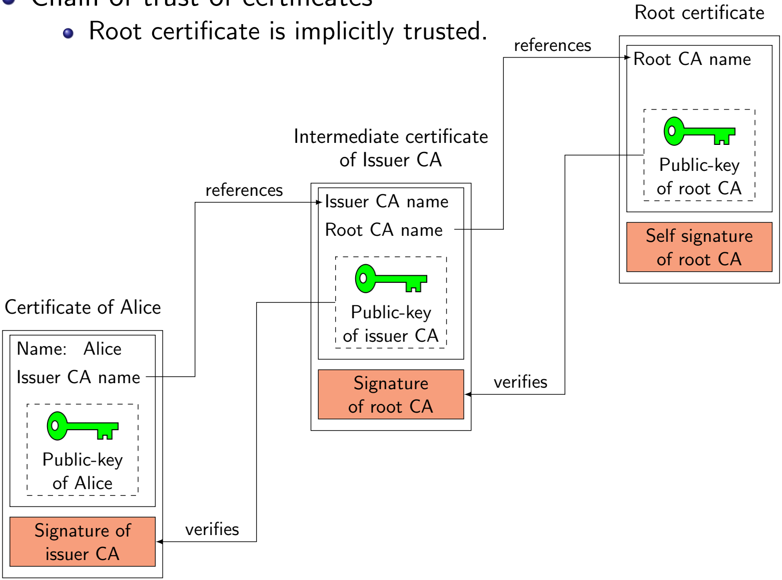


PKI encryption



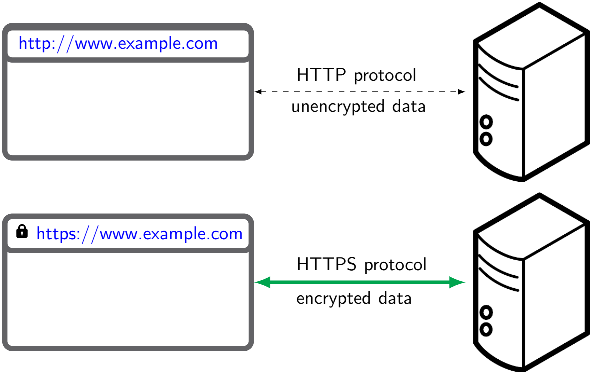
Hierarchy of certificates

- Chain of trust of certificates
  - Root certificate is implicitly trusted.



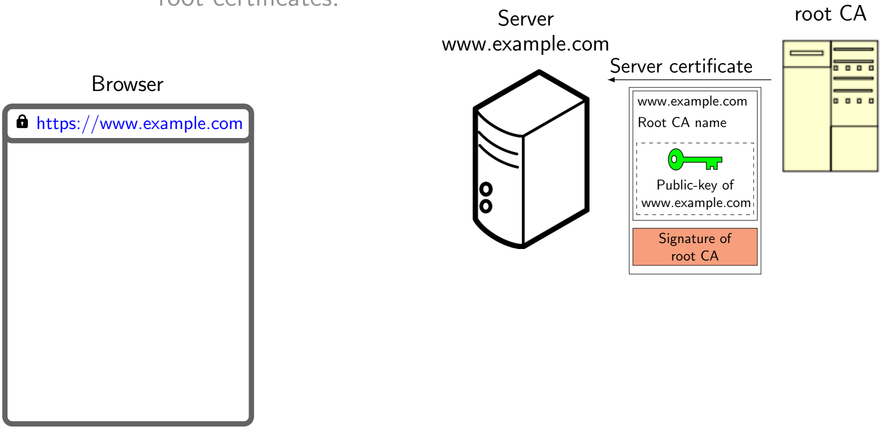
The HTTPS protocol

- Hypertext Transfer Protocol Secure (HTTPS) is a protocol for securely browsing the web.
  - Communication is encrypted using Transport Layer Security (TLS); formerly, Secure Sockets Layer (SSL).
  - Validates the authenticity of an HTTPS web server, and ensures confidentiality and integrity of communications.



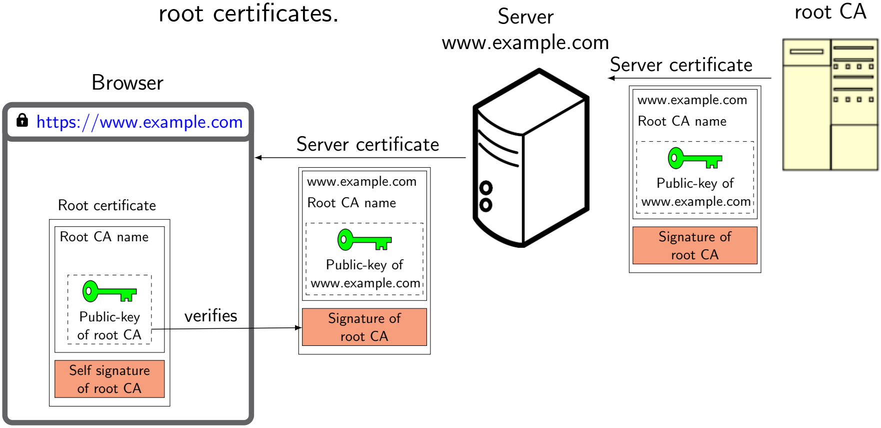
Server authentication in HTTPS

- CA issues a certificate to the server to authenticate the server's public-key
  - The server expects the CA's certificate to be contained in most clients web browser.
  - One needs to trust the browser's publisher to include correct root certificates.



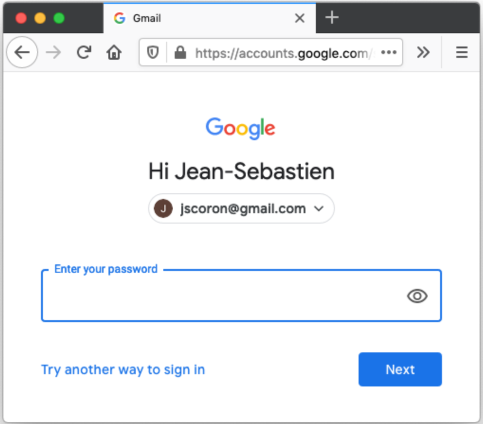
Server authentication in HTTPS

- CA issues a certificate to the server to authenticate the server's public-key
  - The server expects the CA's certificate to be contained in most clients web browser.
  - One needs to trust the browser's publisher to include correct root certificates.



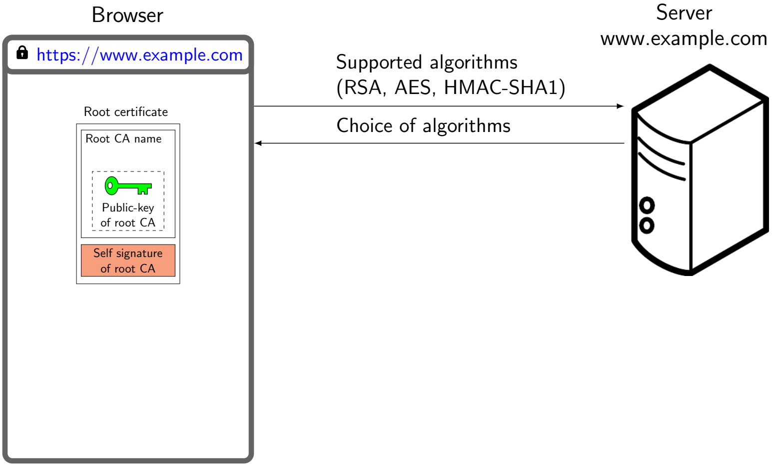
Client authentication in HTTPS

- Generally, only the server is authenticated
  - Mutual authentication requires a client certificate.
  - Most services use passwords to authenticate users, instead of client certificates.



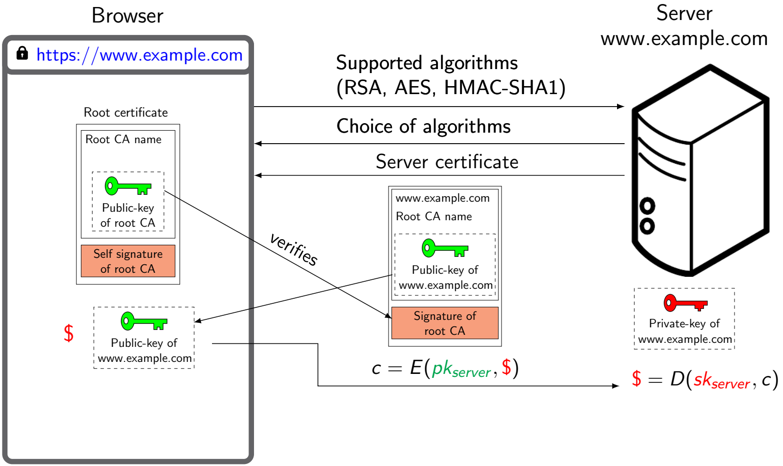
The TLS protocol

- Three steps
  - Negotiation for algorithms used.
  - Certificate verification and PK encryption for session key.
  - Symmetric encryption for traffic encryption.



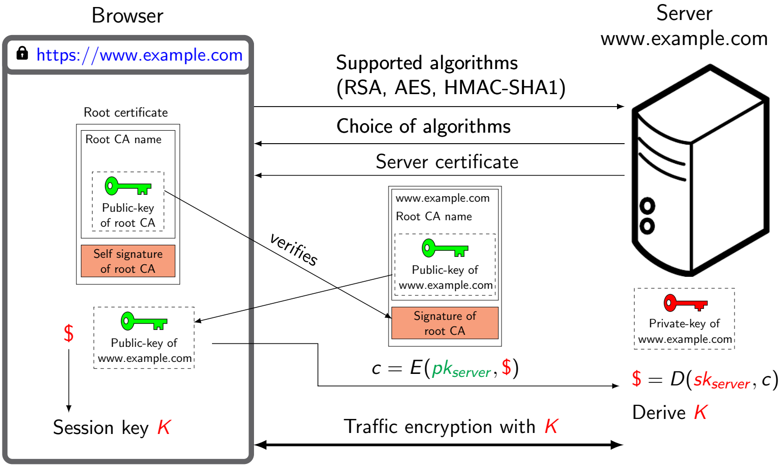
The TLS protocol

- Three steps
  - Negotiation for algorithms used.
  - Certificate verification and PK encryption for session key.
  - Symmetric encryption for traffic encryption.



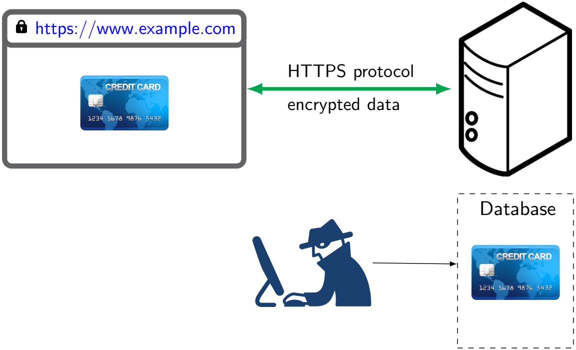
The TLS protocol

- Three steps
  - Negotiation for algorithms used.
  - Certificate verification and PK encryption for session key.
  - Symmetric encryption for traffic encryption.



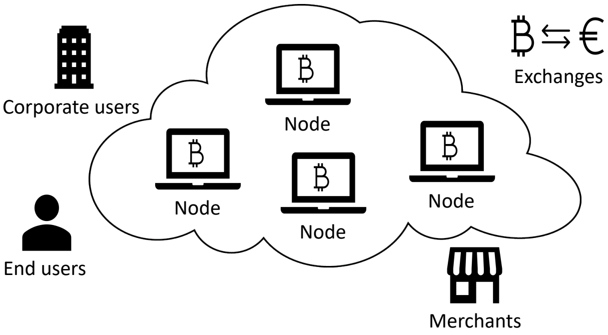
Credit card via HTTPS

- HTTPS only protects the credit card number during transit between the user's computer and the server
  - Does not protect against an attack on the server
- Attack on the server usually easier than interception in transit
  - Credit card number often saved in a database in merchant site
  - Attacks generally concentrate on the server and database



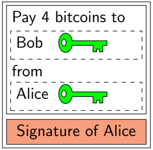
Bitcoin

- Decentralized payment system, invented by Satoshi Nakamoto in 2008.
  - "Bitcoin: a peer-to-peer electronic cash system"
- Network of thousands of computers.
  - No trusted authority. Pseudonymous.
  - Permissionless: anybody can run a Bitcoin node.



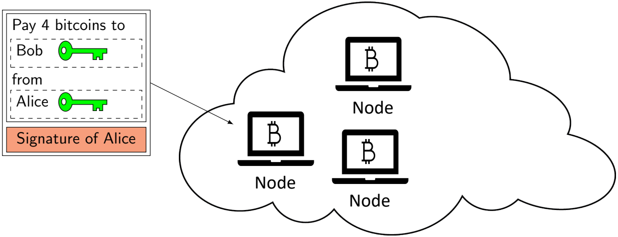
Payment of Bitcoin (simplified)

- Assume that Alice wants to transfer 4 bitcoins to Bob
  - Alice prepares the transaction
    - The amount to transfer (4 bitcoins), the address of the recipient, her digital signature
  - Alice sends the transaction to the network
    - The network verifies the transaction: Alice has 4 bitcoins, signature is valid.
  - Transaction is included in the blockchain
    - Alice's transaction is appended to the transaction history, creating a new *block*.



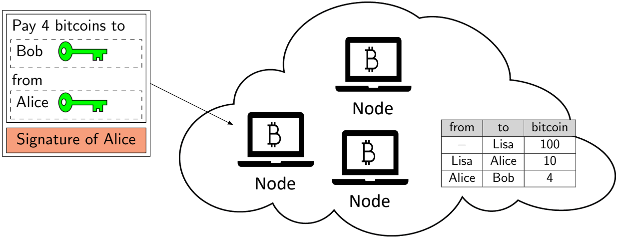
Payment of Bitcoin (simplified)

- Assume that Alice wants to transfer 4 bitcoins to Bob
  - Alice prepares the transaction
    - The amount to transfer (4 bitcoins), the address of the recipient, her digital signature
  - Alice sends the transaction to the network
    - The network verifies the transaction: Alice has 4 bitcoins, signature is valid.
  - Transaction is included in the blockchain
    - Alice's transaction is appended to the transaction history, creating a new *block*.



Payment of Bitcoin (simplified)

- Assume that Alice wants to transfer 4 bitcoins to Bob
  - Alice prepares the transaction
    - The amount to transfer (4 bitcoins), the address of the recipient, her digital signature
  - Alice sends the transaction to the network
    - The network verifies the transaction: Alice has 4 bitcoins, signature is valid.
  - Transaction is included in the blockchain
    - Alice's transaction is appended to the transaction history, creating a new *block*.



from	to	bitcoin
–	Lisa	100
Lisa	Alice	10
Alice	Bob	4

The Bitcoin ledger: simplified model

from	to	bitcoin
—	Lisa	100
Lisa	Alice	10
Alice	Bob	4

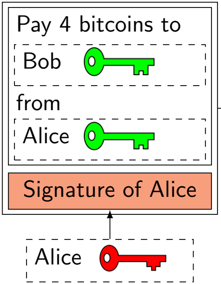
- New transactions are appended at the end of the spreadsheet
- Balance of Alice: sum of “to” minus sum of “from”
  - balance of Alice: 6 bitcoins
  - Alice can transfer 1 bitcoin to Charlie
  - new balance is 5 bitcoins

The Bitcoin ledger: simplified model

from	to	bitcoin
–	Lisa	100
Lisa	Alice	10
Alice	Bob	4
Alice	Charlie	1

- New transactions are appended at the end of the spreadsheet
- Balance of Alice: sum of "to" minus sum of "from"
  - balance of Alice: 6 bitcoins
  - Alice can transfer 1 bitcoin to Charlie
  - new balance is 5 bitcoins

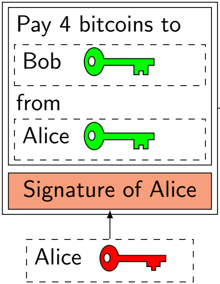
Certifying transactions with signatures



from	to	bitcoin
—	Lisa	100
Lisa	Alice	10
Alice	Bob	4

- Alice signs the transaction to pay 4 bitcoins to Bob
  - Bob is identified by his public-key
  - Anybody can verify the transaction using the public-key of Alice.
- If Alice looses her private-key, she cannot spend her bitcoins.

Certifying transactions with signatures



from	to	bitcoin
—	Lisa	100
Lisa	Alice	10
Alice	Bob	4

- Alice signs the transaction to pay 4 bitcoins to Bob
  - Bob is identified by his public-key
  - Anybody can verify the transaction using the public-key of Alice.
- If Alice looses her private-key, she cannot spend her bitcoins.

Pseudonymous payment

- Names are replaced by public-keys
  - Payments are made via public-keys instead of names.

from	to	bitcoin
–	Lisa	100
Lisa	Alice	10
Alice	Bob	4
Alice	Charlie	1

- One replaces the public-keys by their hash to save space: the *public-key hash* (PKH).
- One can use a unique address for each payment to improve privacy.

Pseudonymous payment

- Names are replaced by public-keys
  - Payments are made via public-keys instead of names.

from	to	bitcoin
—	`45ab...1c32`	100
`45ab...1c32`	`8c24...fe5a`	10
`8c24...fe5a`	`7a53...b3ac`	4
`8c24...fe5a`	`3c6e...02a3`	1

- One replaces the public-keys by their hash to save space: the *public-key hash* (PKH).
- One can use a unique address for each payment to improve privacy.

Preparing a Bitcoin transaction

Confirmed transactions

txid	1	sig	pk	5	PKH_1
txid	0	sig	pk		

txid	2	sig	pk	3	PKH_x
				8	PKH_2

$txid_1$	0	9	PKH_3
$txid_2$	1	4	PKH_4

- A transaction can have multiple inputs
  - to spend bitcoins from previous transactions
  - reference txid and output index.
- A transaction can have multiple outputs
  - number of bitcoins and output PKH.

Preparing a Bitcoin transaction

Confirmed transactions

txid	1	sig	pk	5	PKH_1
txid	0	sig	pk		

txid	2	sig	pk	3	PKH_x
				8	PKH_2

$txid_1$	0	9	PKH_3
$txid_2$	1	4	PKH_4

- A transaction can have multiple inputs
  - to spend bitcoins from previous transactions
  - reference txid and output index.
- A transaction can have multiple outputs
  - number of bitcoins and output PKH.

Preparing a Bitcoin transaction

Confirmed transactions

txid	1	sig	pk	5	PKH_1
txid	0	sig	pk		

txid	2	sig	pk	3	PKH_x
				8	PKH_2

txid_1	0	9	PKH_3
txid_2	1	4	PKH_4

- A transaction can have multiple inputs
  - to spend bitcoins from previous transactions
  - reference txid and output index.
- A transaction can have multiple outputs
  - number of bitcoins and output PKH.

Preparing a Bitcoin transaction

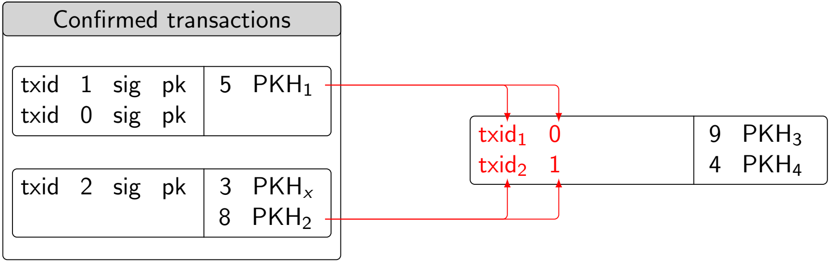
Confirmed transactions

txid	1	sig	pk	5	PKH_1
txid	0	sig	pk		

txid	2	sig	pk	3	PKH_x
				8	PKH_2

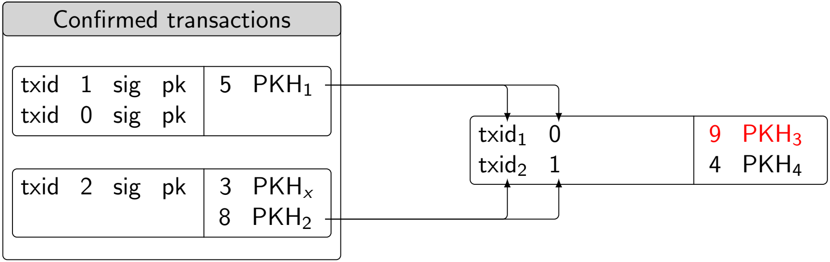
$txid_1$	0	9	PKH_3
$txid_2$	1	4	PKH_4

- Unspent transaction outputs (UTXOs)
  - coins received by Alice that she can spend ($5 + 8 = 13$ Ƀ)
  - by referencing the transaction (txid) and the output index (idx) in the transaction.
- Alice pays 9 Ƀ to Bob's address PKH_3 (output index 0)
  - and 3 Ƀ to herself (change) in a fresh address PKH_4.



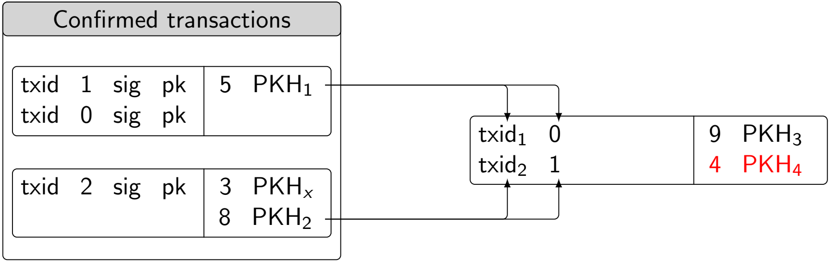
- Unspent transaction outputs (UTXOs)
  - coins received by Alice that she can spend ($5 + 8 = 13$ Ƀ)
  - by referencing the transaction (txid) and the output index (idx) in the transaction.
- Alice pays 9 Ƀ to Bob's address PKH_3 (output index 0)
  - and 3 Ƀ to herself (change) in a fresh address PKH_4.

Preparing a Bitcoin transaction



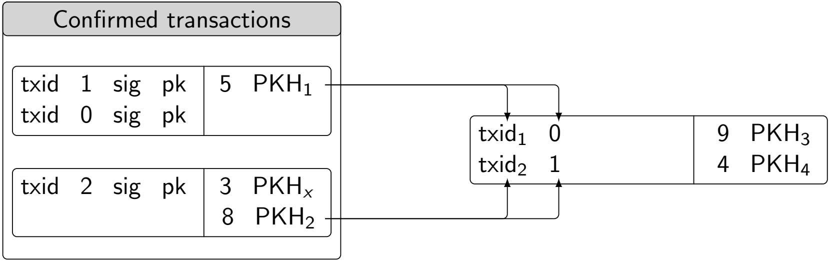
- Unspent transaction outputs (UTXOs)
  - coins received by Alice that she can spend ($5 + 8 = 13$ Ƀ)
  - by referencing the transaction (txid) and the output index (idx) in the transaction.
- Alice pays 9 Ƀ to Bob's address PKH$_3$ (output index 0)
  - and 3 Ƀ to herself (change) in a fresh address PKH$_4$.

Preparing a Bitcoin transaction

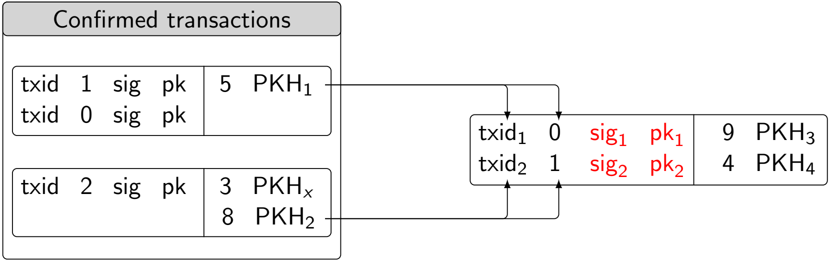


- Unspent transaction outputs (UTXOs)
  - coins received by Alice that she can spend ($5 + 8 = 13$ Ƀ)
  - by referencing the transaction (txid) and the output index (idx) in the transaction.
- Alice pays 9 Ƀ to Bob's address PKH_3 (output index 0)
  - and 3 Ƀ to herself (change) in a fresh address PKH_4.

Signing a transaction



- Alice knows the private-keys corresponding to the addresses PKH$_1$ and PKH$_2$
  - She can "unlock" the two UTXOs by signing the transaction.
  - She inserts the public-keys pk_1 and pk_2 so that people can verify.
  - Each signature commits the entire transaction.
- Anyone can verify the transaction



- Alice knows the private-keys corresponding to the addresses PKH$_1$ and PKH$_2$
  - She can "unlock" the two UTXOs by signing the transaction.
  - She inserts the public-keys pk_1 and pk_2 so that people can verify.
  - Each signature commits the entire transaction.
- Anyone can verify the transaction

Account-based vs value-based system

- Account-based system

from	to	bitcoin
PKH_y	PKH_1	5
PKH_z	PKH_x	3
PKH_z	PKH_2	8

  - Each PKH is an account
  - One must keep the balance of each account

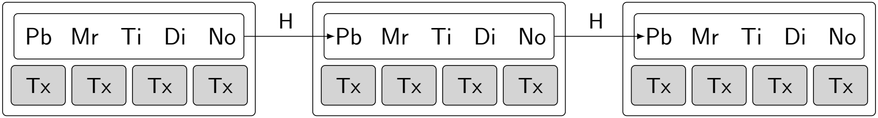
- Value-based system (Bitcoin)

txid	idx	output
$txid_1$	0	5 PKH_1
$txid_2$	0	3 PKH_x
$txid_2$	1	8 PKH_2

  - Each transaction output is a "coin" that can be spent only once.
  - One must keep track of the UTXO set

The blockchain

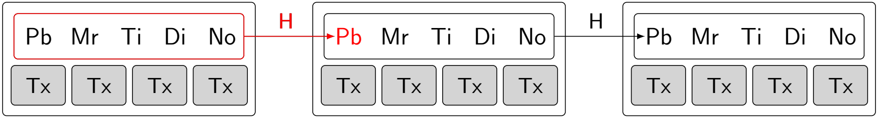
- The blockchain is a sequence of blocks of transactions
  - connected through cryptographic hashes



- The block header contains
  - the hash of the previous block header (prev_block)
  - the combined hash of the transactions (merkle_root)
  - a timestamp, the target difficulty, and the nonce.
- Security of the blockchain
  - Given the hash in the last block header, one cannot modify any previous block
  - one cannot modify any transaction in a previous block

The blockchain

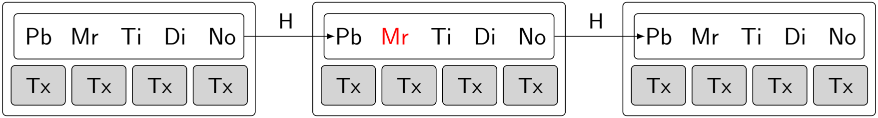
- The blockchain is a sequence of blocks of transactions
  - connected through cryptographic hashes



- The block header contains
  - the hash of the previous block header (prev_block)
  - the combined hash of the transactions (merkle_root)
  - a timestamp, the target difficulty, and the nonce.
- Security of the blockchain
  - Given the hash in the last block header, one cannot modify any previous block
  - one cannot modify any transaction in a previous block

The blockchain

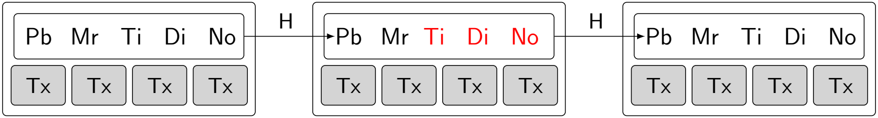
- The blockchain is a sequence of blocks of transactions
  - connected through cryptographic hashes



- The block header contains
  - the hash of the previous block header (prev_block)
  - the combined hash of the transactions (merkle_root)
  - a timestamp, the target difficulty, and the nonce.
- Security of the blockchain
  - Given the hash in the last block header, one cannot modify any previous block
  - one cannot modify any transaction in a previous block

The blockchain

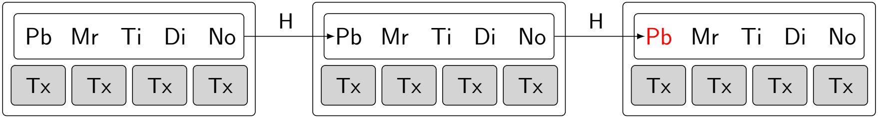
- The blockchain is a sequence of blocks of transactions
  - connected through cryptographic hashes



- The block header contains
  - the hash of the previous block header (prev_block)
  - the combined hash of the transactions (merkle_root)
  - a timestamp, the target difficulty, and the nonce.
- Security of the blockchain
  - Given the hash in the last block header, one cannot modify any previous block
  - one cannot modify any transaction in a previous block

The blockchain

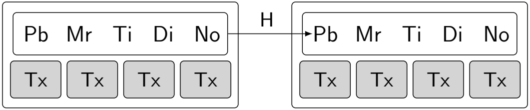
- The blockchain is a sequence of blocks of transactions
  - connected through cryptographic hashes



- The block header contains
  - the hash of the previous block header (prev_block)
  - the combined hash of the transactions (merkle_root)
  - a timestamp, the target difficulty, and the nonce.
- Security of the blockchain
  - Given the hash in the last block header, one cannot modify any previous block
  - one cannot modify any transaction in a previous block

Proof of work

- Miners compete to create the next block
  - Proof of work: requires to compute a huge number of cryptographic hashes
  - Miners are rewarded by the block subsidy and transaction fees.



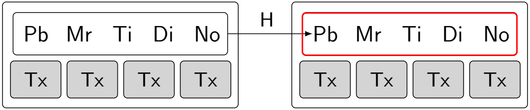
- Proof of work
  - The hash of the block header must be below the target

```
block_id:  0000000000000000000094002a8b505cd2509bcbfe9f5da0a3d7ccd209887c134
target  :  000000000000000000170cfe0000000000000000000000000000000000000000
```

  - by adjusting the 32-bit nonce.
  - Anybody can verify the proof of work by hashing the block header.

Proof of work

- Miners compete to create the next block
  - Proof of work: requires to compute a huge number of cryptographic hashes
  - Miners are rewarded by the block subsidy and transaction fees.



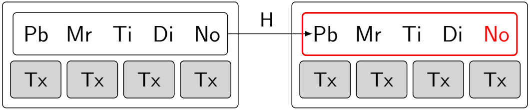
- Proof of work
  - The hash of the block header must be below the target

```
block_id:  000000000000000000094002a8b505cd2509bcbfe9f5da0a3d7ccd209887c134
target  :  0000000000000000000170cfe000000000000000000000000000000000000000
```

  - by adjusting the 32-bit nonce.
  - Anybody can verify the proof of work by hashing the block header.

Proof of work

- Miners compete to create the next block
  - Proof of work: requires to compute a huge number of cryptographic hashes
  - Miners are rewarded by the block subsidy and transaction fees.



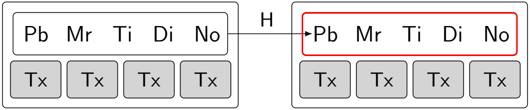
- Proof of work
  - The hash of the block header must be below the target

```
block_id:  0000000000000000000094002a8b505cd2509bcbfe9f5da0a3d7ccd209887c134
target  :  0000000000000000000170cfe000000000000000000000000000000000000000
```

  - by adjusting the 32-bit nonce.
  - Anybody can verify the proof of work by hashing the block header.

Proof of work

- Miners compete to create the next block
  - Proof of work: requires to compute a huge number of cryptographic hashes
  - Miners are rewarded by the block subsidy and transaction fees.



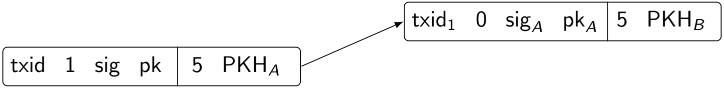
- Proof of work
  - The hash of the block header must be below the target

```
block_id:  0000000000000000000094002a8b505cd2509bcbfe9f5da0a3d7ccd209887c134
target  :  000000000000000000170cfe0000000000000000000000000000000000000000
```

  - by adjusting the 32-bit nonce.
  - Anybody can verify the proof of work by hashing the block header.

Preventing double spending

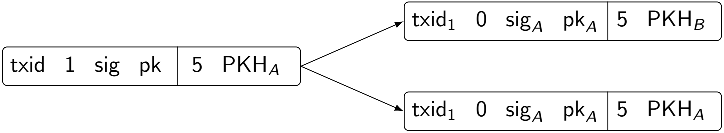
- Double spending
  - Alice prepares a transaction of 5 Ƀ to Bob in exchange of his car



  - She also prepares a double-spend transaction to herself.
  - Both transactions are valid separately, but they cannot be both included in the blockchain.

Preventing double spending

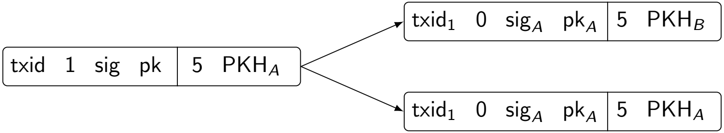
- Double spending
  - Alice prepares a transaction of 5 Ƀ to Bob in exchange of his car



  - She also prepares a double-spend transaction to herself.
  - Both transactions are valid separately, but they cannot be both included in the blockchain.

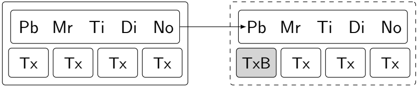
Preventing double spending

- Double spending
  - Alice prepares a transaction of 5 Ƀ to Bob in exchange of his car



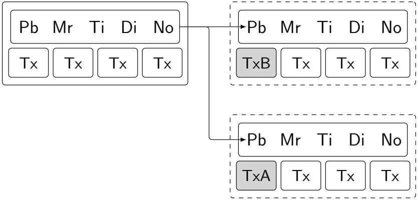
  - She also prepares a double-spend transaction to herself.
  - Both transactions are valid separately, but they cannot be both included in the blockchain.

Preventing double spending



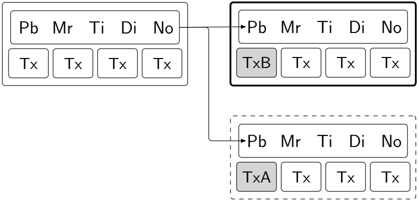
- Double spending
  - Alice sends the payment of 5 Ƀ to Bob (TxB) to all miners.
  - She also secretly mines a block with the double-spend transaction (TxA)
  - The payment to Bob is now included in the next block. Bob gives his car to Alice.
  - Alice finds the proof of work with the double-spend transaction
  - She finds the proof of work for another block. She publishes the two blocks.
  - The miners start mining on the longest chain.
  - Alice can keep her 5 Ƀ.

Preventing double spending



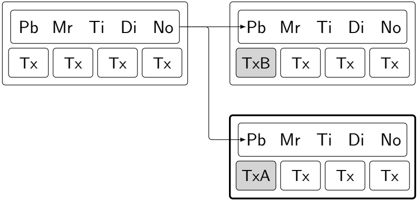
- Double spending
  - Alice sends the payment of 5 Ƀ to Bob (TxB) to all miners.
  - She also secretly mines a block with the double-spend transaction (TxA)
  - The payment to Bob is now included in the next block. Bob gives his car to Alice.
  - Alice finds the proof of work with the double-spend transaction
  - She finds the proof of work for another block. She publishes the two blocks.
  - The miners start mining on the longest chain.
  - Alice can keep her 5 Ƀ.

Preventing double spending



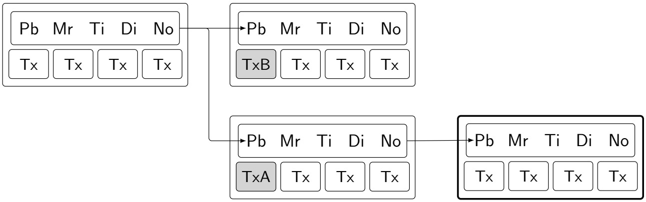
- Double spending
  - Alice sends the payment of 5 Ƀ to Bob (TxB) to all miners.
  - She also secretly mines a block with the double-spend transaction (TxA)
  - The payment to Bob is now included in the next block. Bob gives his car to Alice.
  - Alice finds the proof of work with the double-spend transaction
  - She finds the proof of work for another block. She publishes the two blocks.
  - The miners start mining on the longest chain.
  - Alice can keep her 5 Ƀ.

Preventing double spending



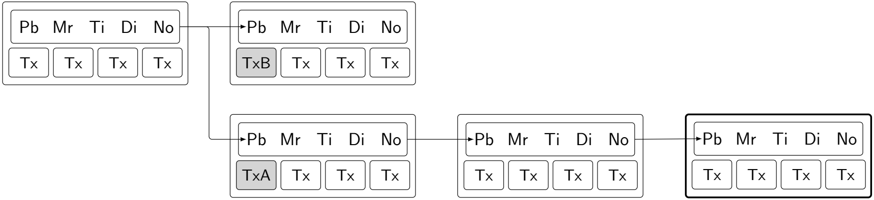
- Double spending
  - Alice sends the payment of 5 Ƀ to Bob (TxB) to all miners.
  - She also secretly mines a block with the double-spend transaction (TxA)
  - The payment to Bob is now included in the next block. Bob gives his car to Alice.
  - Alice finds the proof of work with the double-spend transaction
  - She finds the proof of work for another block. She publishes the two blocks.
  - The miners start mining on the longest chain.
  - Alice can keep her 5 Ƀ.

Preventing double spending



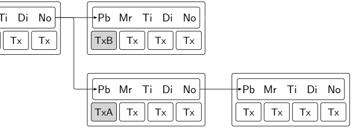
- Double spending
  - Alice sends the payment of 5 Ƀ to Bob (TxB) to all miners.
  - She also secretly mines a block with the double-spend transaction (TxA)
  - The payment to Bob is now included in the next block. Bob gives his car to Alice.
  - Alice finds the proof of work with the double-spend transaction
  - She finds the proof of work for another block. She publishes the two blocks.
  - The miners start mining on the longest chain.
  - Alice can keep her 5 Ƀ.

Preventing double spending



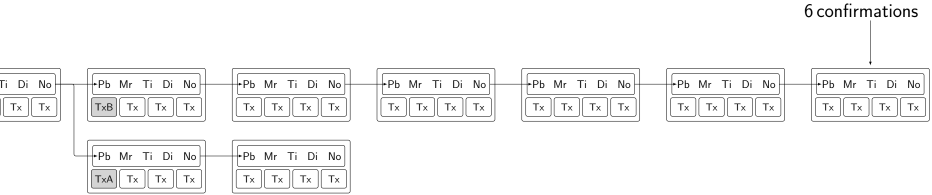
- Double spending
  - Alice sends the payment of 5 Ƀ to Bob (TxB) to all miners.
  - She also secretly mines a block with the double-spend transaction (TxA)
  - The payment to Bob is now included in the next block. Bob gives his car to Alice.
  - Alice finds the proof of work with the double-spend transaction
  - She finds the proof of work for another block. She publishes the two blocks.
  - The miners start mining on the longest chain.
  - Alice can keep her 5 Ƀ.

Preventing double spending



- To prevent double-spend attacks, the recipient should wait for more confirmations
  - For example, at least 6 confirmations: 5 blocks have been mined after the block with TxB
  - If the hash-rate of Alice is a small fraction of the total hash-rate, the probability of a double-spend becomes negligible.

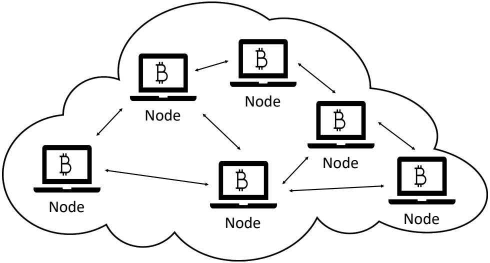
Preventing double spending



- To prevent double-spend attacks, the recipient should wait for more confirmations
  - For example, at least 6 confirmations: 5 blocks have been mined after the block with TxB
  - If the hash-rate of Alice is a small fraction of the total hash-rate, the probability of a double-spend becomes negligible.

Conclusion

- Bitcoin is a decentralized payment protocol
  - Transactions and blocks are relayed through a peer-to-peer network
  - No central source of authority needed



- Other concepts in Bitcoin
  - Difficulty adjustment: a block mined every 10 mins.
  - Payment script: more flexibility.
  - Lightweight wallet: fast verification using Merkle tree.