UNIVERSITÄT BONN

IT SECURITY

CYBER SECURITY OF DISTRIBUTED AND RESOURCE LIMITED SYSTEMS

THORSTEN AURISCH

FRAUNHOFER FKIE, KOM

About Me

- Studied physics at the University of Bonn
- Get PhD in Computer Science
- At Fraunhofer FKIE since 1998
- Research areas autonomous cyber security mechanisms, cyber resilience, key management
- Supervise labs, seminars and theses

Content

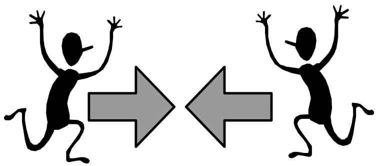
- Motivation
- Chapter 1: Confidentiality in distributed systems
- Chapter 2: Integrity, authenticity in distributed systems
- Chapter 3: Secure distributed documentation

Motivation

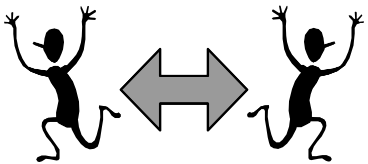
- This is a high-speed walk through selected topics
- There is a full lecture on cyber security of distributed and resource limited systems
- If you want to know more, visit the lecture

MOTIVATION: SECURITY FOR TWO PARTIES

START

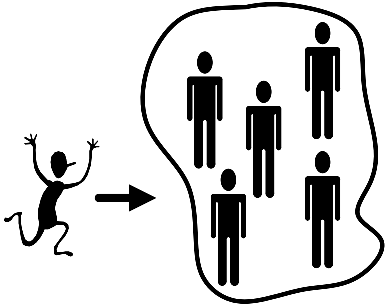


END

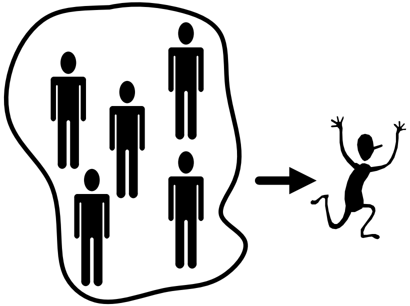


Motivation: Membership Operations in Distributed Systems

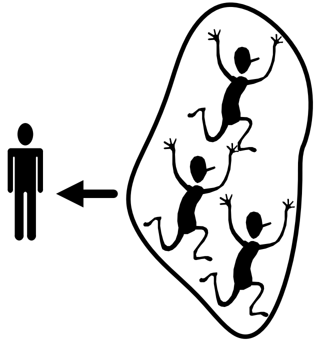
User JOIN



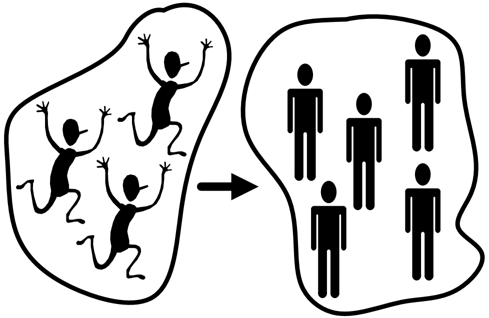
User LEAVE



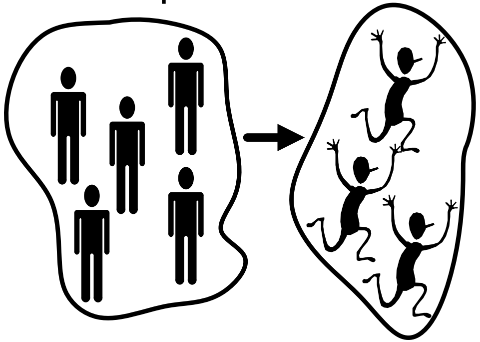
User EJECT



Group MERGE



Group PARTITION



UNIVERSITÄT BONN

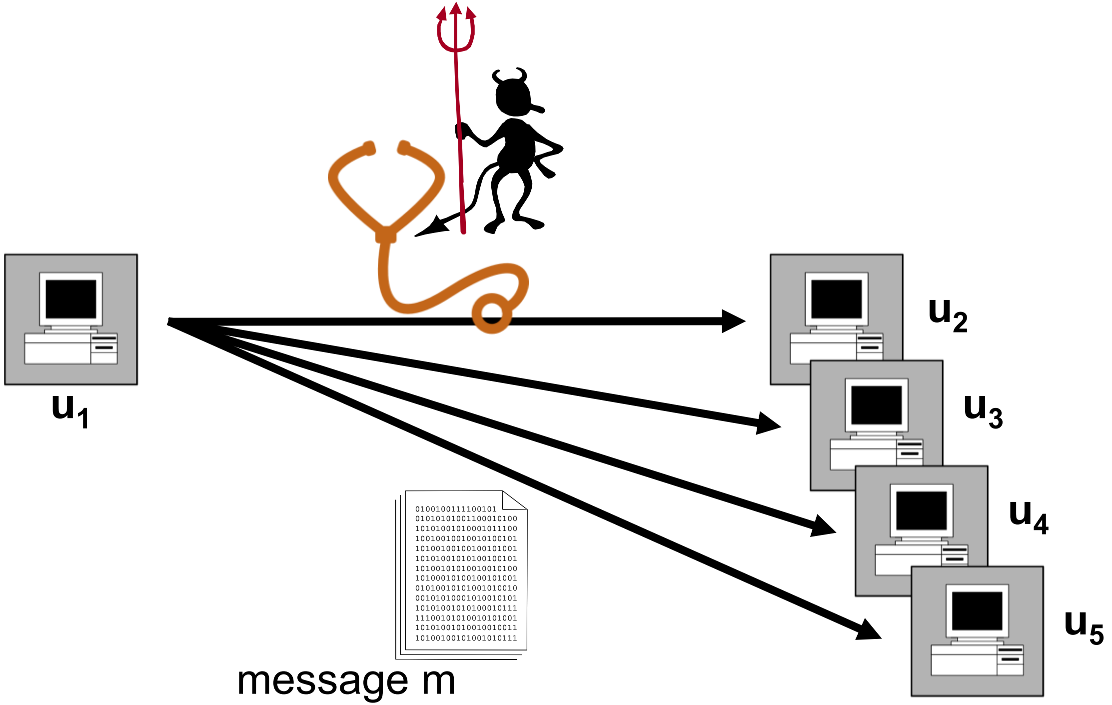
Motivation: Distributed Security Mechanisms

- There are good reasons to use distributed security mechanisms
  - Secrets are too important to be kept by one user
  - It is easier to trust many than one user
  - Robustness is an important factor if multiple users are involved

Chapter 1: Learning Objective

- We want to understand the protection mechanisms against eavesdropping

Example:
chat system

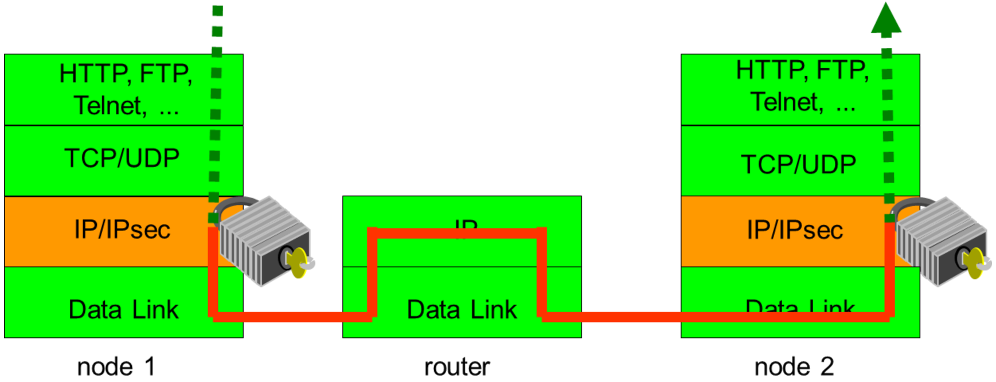


XMPP

[psaik]

CHAPTER 1: CONFIDENTIALITY IS PROVIDED BY ENCRYPTION

- Confidentiality can be provided by encryption E(m,k) with a key k
- <u>Example:</u> IPsec is a security protocol that specifies how network traffic is encrypted
  - IP Authentication Header (AH) defines a method for IP packet authentication
  - IP Encapsulating Security Payload (ESP) defines a method for IP packet encryption (and payload authentication)
- However, a <u>dynamic key</u> is need to handle composition changes in the system



UNIVERSITÄT BONN

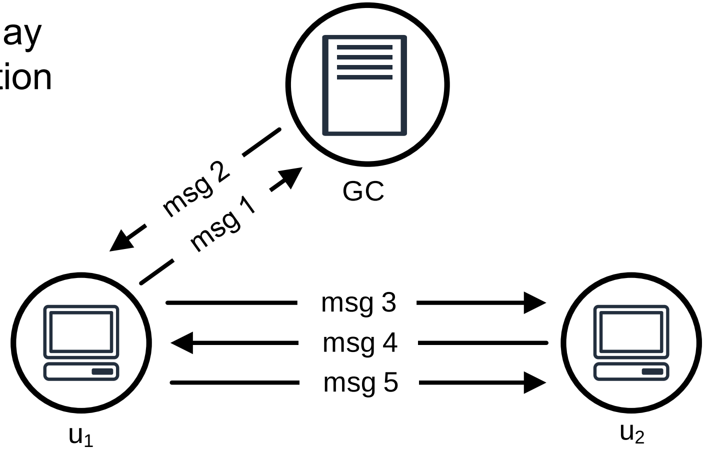
Chapter 1: Needham-Schroeder Protocol – Dynamic Pairwise Keys (1)

- A dynamic key can be established for each user pair by the Needham-Schroeder protocol
- Example: Pairwise key k_{12} for the users u_1 and u_2 by a Group Controller (GC)
  - u_1 with ID_{u1}, u_2 with ID_{u2}
  - u_1 / GC share $k_{1,GC}$, u_2 / GC share $k_{2,GC}$

nonce as replay attack protection

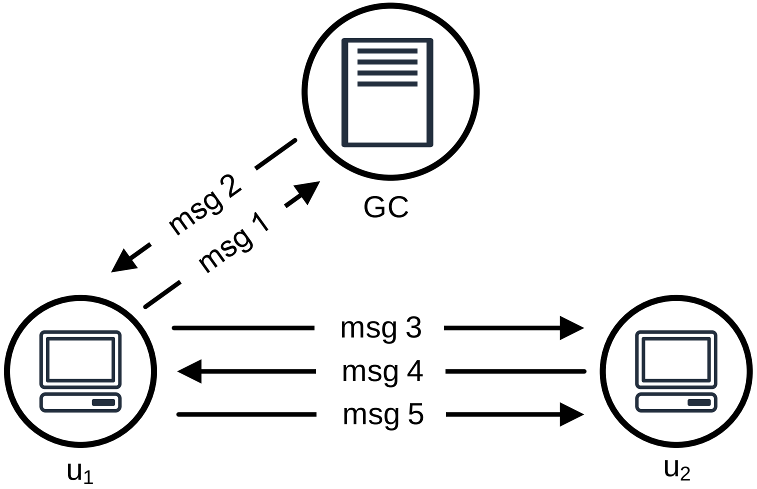
1. $u_1 \rightarrow GC$: ID_{u1}, ID_{u2}, n_{u1}
2. $u_1 \leftarrow GC$: $E(\{n_{u1}, k_{12}, ID_{u2}, \mathbf{E(\{k_{12}, ID_{u1}\}, k_{2,GC})}\}, k_{1,GC})$
3. $u_1 \rightarrow u_2$: $E(\tilde{n}_{u1}, k_{12}), \mathbf{E(\{k_{12}, ID_{u1}\}, k_{2,GC})}$
4. $u_1 \leftarrow u_2$: $E((\tilde{n}_{u1}-1, n_{u2}), k_{12})$
5. $u_1 \rightarrow u_2$: $E((n_{u2}-1), k_{12})$

ticket



Chapter 1: Needham-Schroeder Protocol – Dynamic Pairwise Keys (2)

- Disadvantages of the Needham-Schroeder protocol
  - The Group Controller is a single point of failure
  - Handling of network separation and fusion is not supported
  - Pairwise key establishment has a limited scalability



Chapter 1: Scalable Security

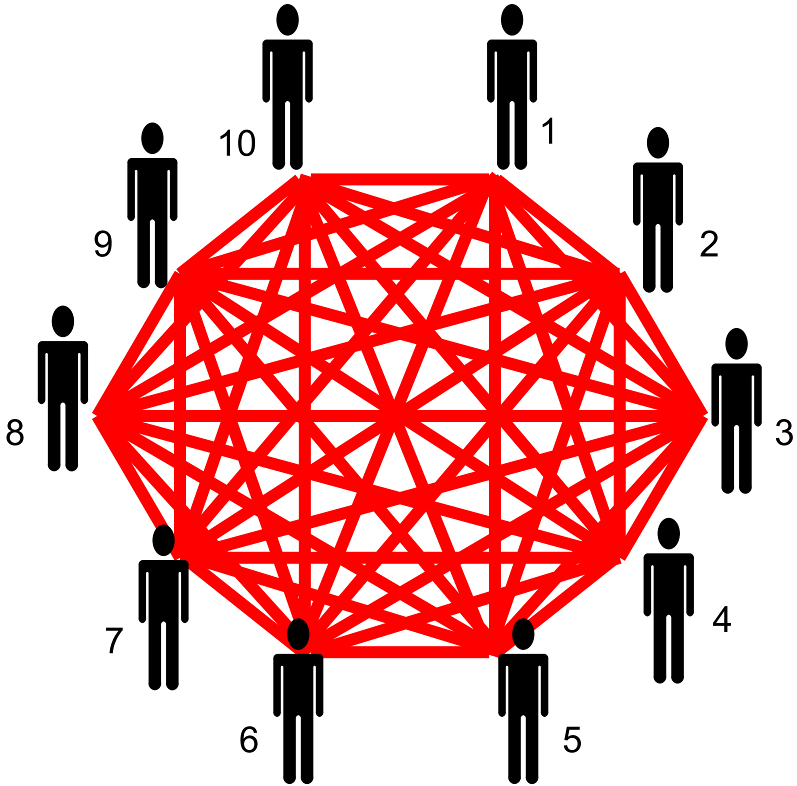
- Using the Needham-Schroeder protocol means each user pair needs a key
- The number of pairwise keys in a distributed system of n users can be calculated with:

$$\#keys = f(n) = \frac{n \cdot (n-1)}{2}$$

- 10 users need 45 secret keys
- The number of key grows quadratically with number of users

$$f(n) \in O(n^2)$$

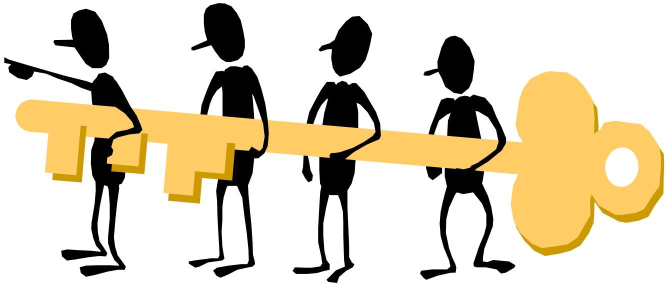
Big O notation to classify algorithms efficiency



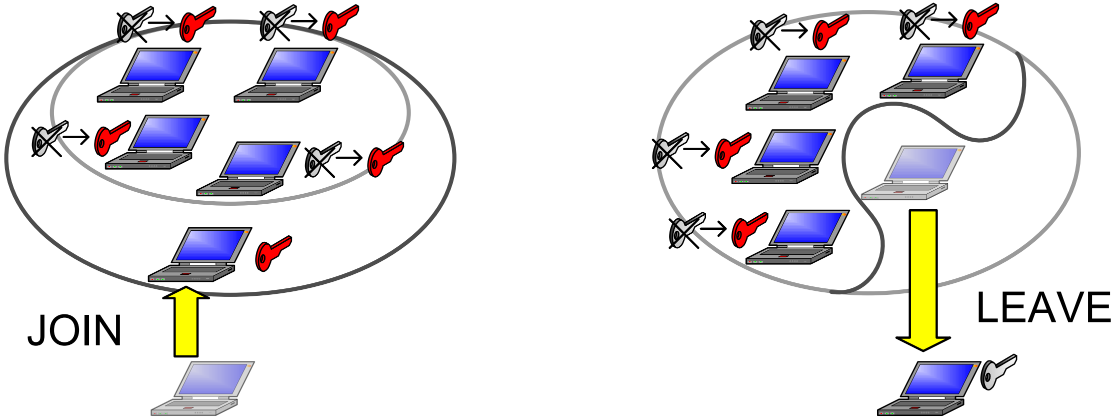
Chapter 1: Group Key for Efficiency

- Efficient confidentiality in distributed systems can be achieved by a shared <u>group key</u> for the encryption E(m,k) of a security protocol

- The group key is only provided for authorized users



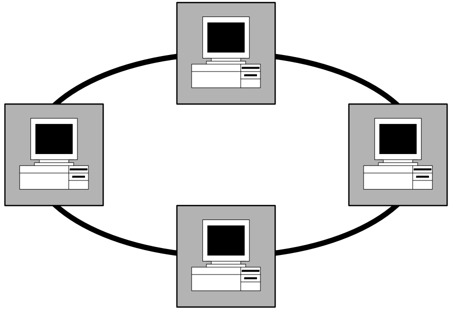
Chapter 1: Group Key Security Requirements



- Only authorized users receive the group key (key secrecy)
- In dynamic groups key secrecy is guaranteed by altering the group key (rekeying)
  - Authorized users can join the group but receive no old group key (forward secrecy)
  - Users can leave the group but receive no new group key (backward secrecy)

Chapter 1: Distributed Group Key Management for Confidentiality

- <u>Concept</u>: Every user contributes to the session key for encryption
  - The key establishment and update mechanisms are based on an iterative application of the Diffie-Hellman algorithm
- Advantages
  - It is easier to trust many users than one GC
  - The key establishment is done without a single point of failure
  - The mechanisms can handle network partition and merge
- Disadvantages
  - Difficult key establishment in networks with high packet losses

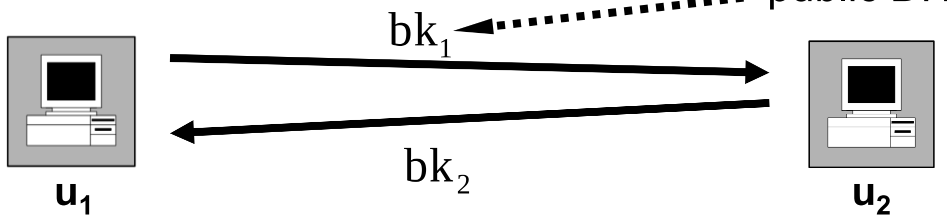


Chapter 1: Refresher - "Distributed" Two Party Key Management - Diffie-Hellman Algorithm (DH)

group with multiplication modulo p as the operation

- Generating a shared secret key over public channels
- $\mathbb{Z}^*_p$={1, ..., p – 1}, p = prime, g = generator of $\mathbb{Z}^*_p$

public DH key = blind key

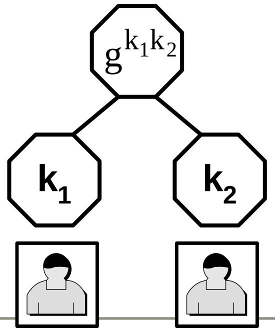


k_1 = secret key u_1

$bk_1 = g^{k_1} \bmod p \ := BK(k_1)$

$k_{12} = (g^{k_2})^{k_1} \bmod p$
$:= DH(bk_2, k_1)$

DH displayed as key tree



k_2 = secret key u_2

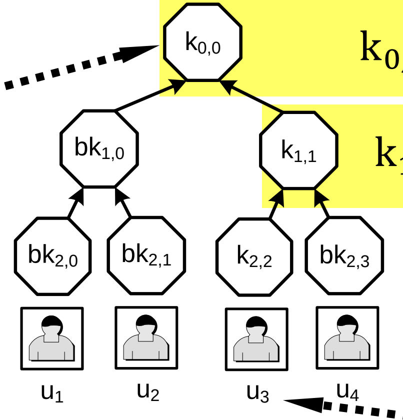
$bk_2 = g^{k_2} \bmod p := BK(k_2)$

$k_{12} = (g^{k_1})^{k_2} \bmod p$
$:= DH(bk_1, k_2)$

Chapter 1: Tree-based Group Diffie-Hellman (TGDH) Protocol – Basis

- All users u_i are arranged in a tree, generate keys k_i / blind keys bk_i according to the DH algorithm, and assign the individual keys to the leaves of the tree
- Every user knows the blind keys of all other tree nodes
- Root key calculation by an iterative application of the Diffie-Hellman algorithm

Root key is used for application data encryption



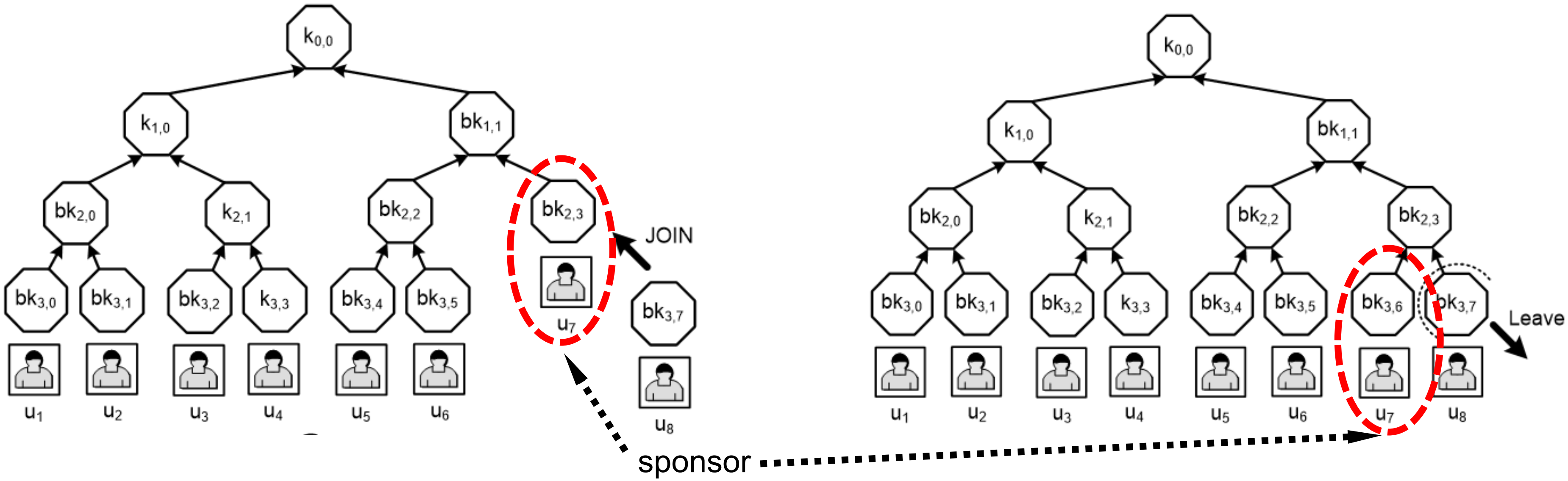
$$k_{0,0} = (bk_{1,0})^{k_{1,1}} \bmod p \coloneqq DH(bk_{1,0}, k_{1,1})$$

$$k_{1,1} = (bk_{2,3})^{k_{2,2}} \bmod p \coloneqq DH(bk_{2,3}, k_{2,2})$$

Root key calculation of u_3 by applying the DH algorithm twice

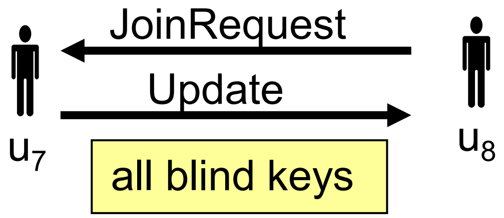
Chapter 1: TGDH – Handling Membership Operations

- A <u>sponsor</u> is selected for handling the membership operation (JOIN, LEAVE)
  - JOIN: The sponsor is the user whose leaf is split
  - LEAVE: The sponsor is the user who is the sibling in the key tree

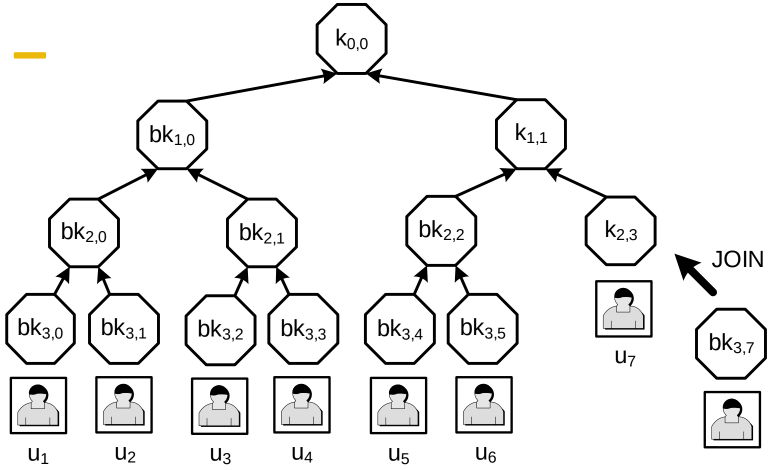


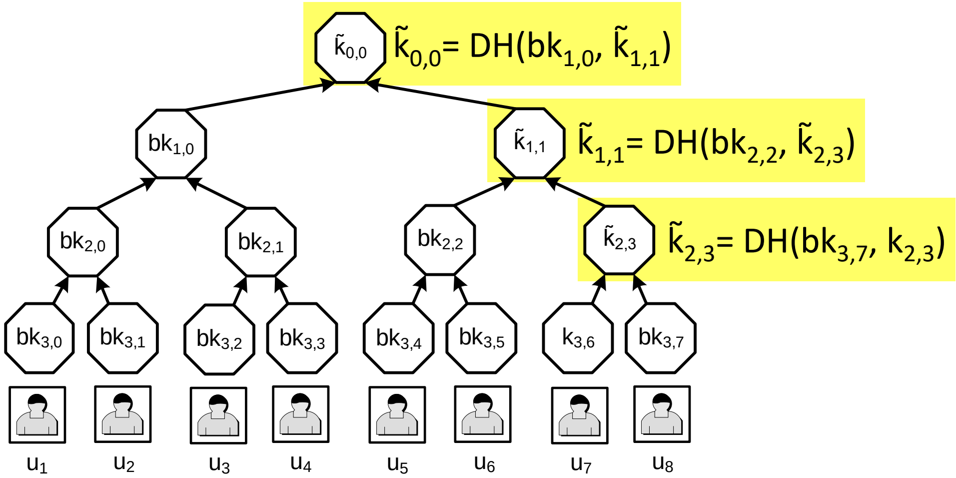
Chapter 1: TGDH – JOIN (1)

0. Group u_i $i \in \{1,2,3,4,5,6,7\}$
   JOIN u_8 with $k_{3,7}$, $bk_{3,7}$
   New sponsor u_7

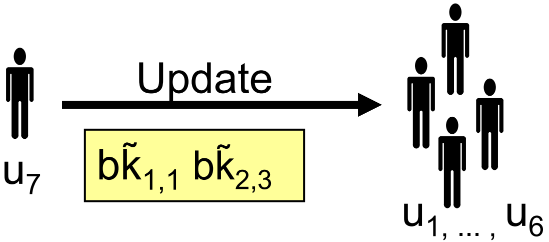
1. 

u_7 group key calculation:
$\tilde{k}_{2,3}$, $\tilde{k}_{1,1}$, $k\tilde{b}_{1,1}$, $\tilde{k}_{0,0}$

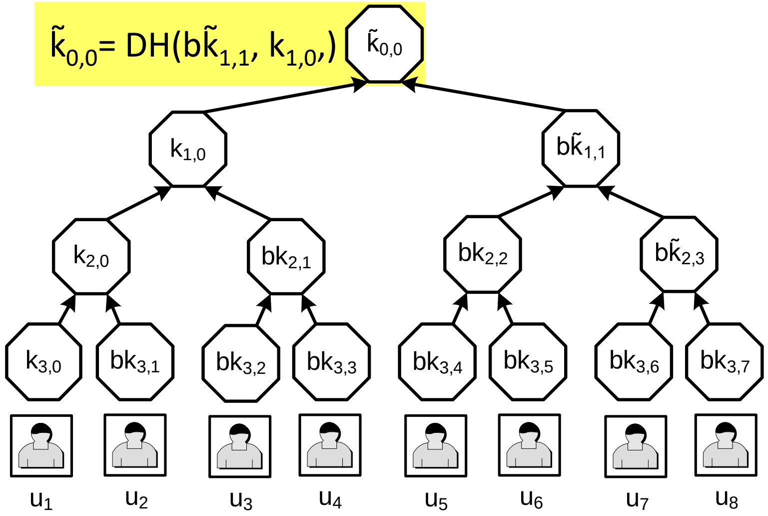




CHAPTER 1: TGDH – JOIN (2)

2. 

3. e.g. u_1 group key calculation (receive $b\tilde{k}_{1,1}$)
   $\tilde{k}_{0,0} = DH(b\tilde{k}_{1,1}, k_{1,0})$

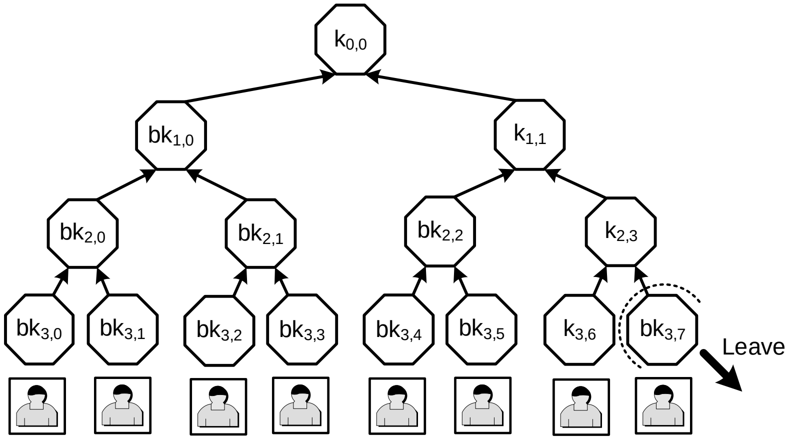


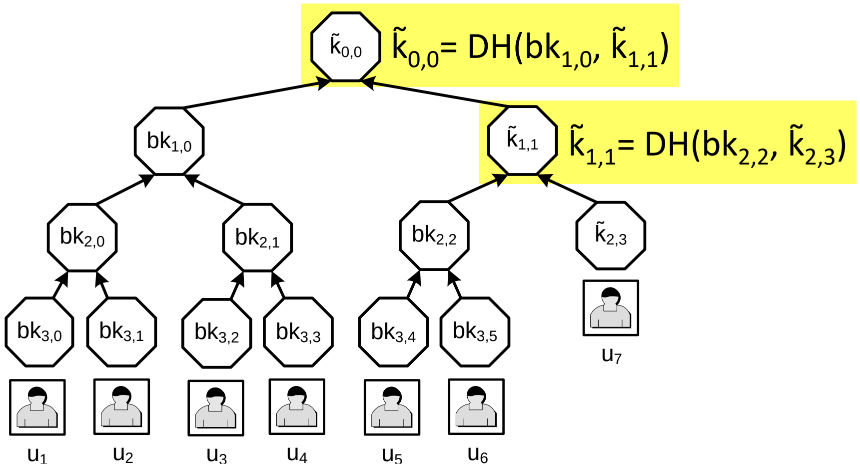
Chapter 1: TGDH – Leave (1)

0. Group u_i $i \in \{1,2,3,4,5,6,7,8\}$
   LEAVE u_8
   New sponsor u_7

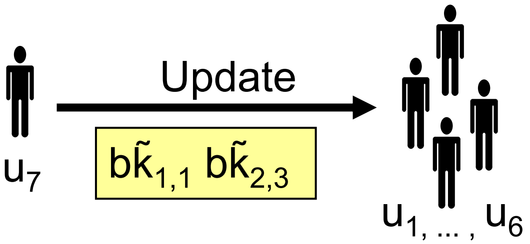
1. 

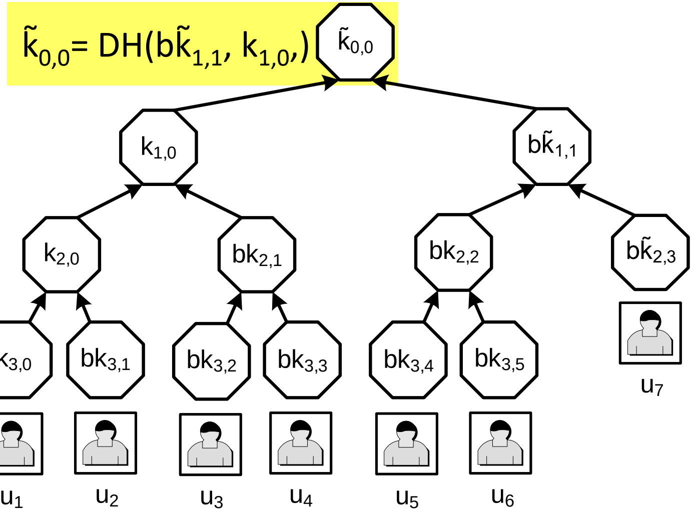
   u_7 group key calculation (refresh $\tilde{k}_{2,3}$, $b\tilde{k}_{2,3}$):
   $\tilde{k}_{1,1}$, $k\tilde{b}_{1,1}$, $\tilde{k}_{0,0} = DH(bk_{1,0}, \tilde{k}_{1,1})$





CHAPTER 1: TGDH – LEAVE (2)

2. 

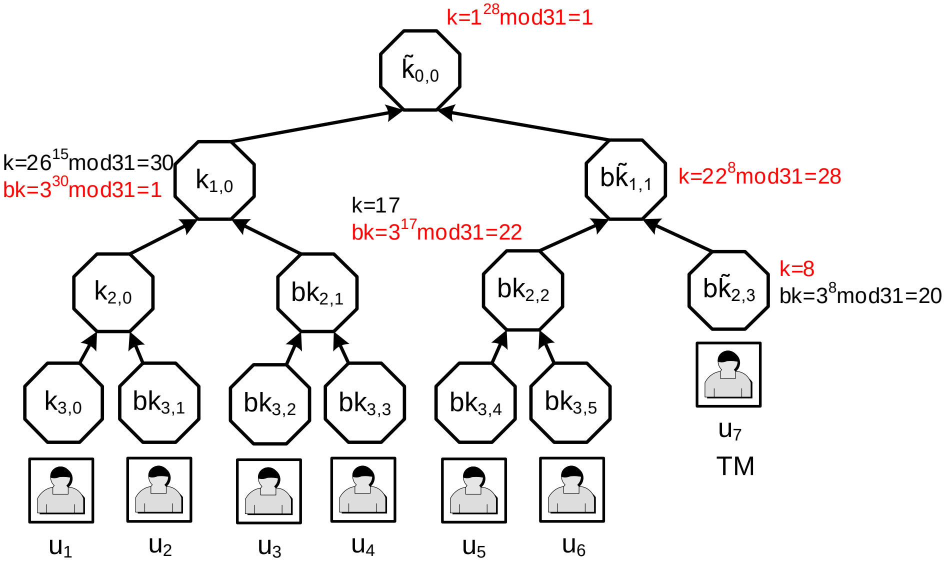


3. e.g. u_1 group key calculation (receive $b\tilde{k}_{1,1}$)
$\tilde{k}_{0,0}= DH(b\tilde{k}_{1,1}, k_{1,0})$

CHAPTER 1: TGDH – LEAVE EXAMPLE CALCULATION

0. Group u_i $i \in \{1,2,3,4,5,6,7,8\}$
   LEAVE u_8
   New sponsor u_7 , g = 3, p = 31
1. 

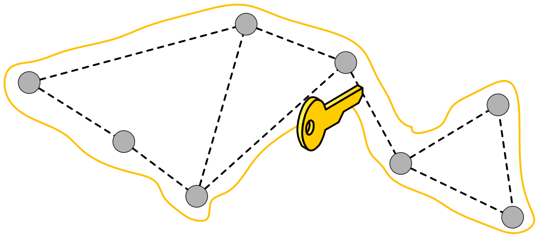
   u_7 group key calculation:
   Refresh $\tilde{k}_{2,3}$, $b\tilde{k}_{2,3}$
   $\tilde{k}_{1,1}$= DH($bk_{2,2}$, $\tilde{k}_{2,3}$), $b\tilde{k}_{1,1}$= BK($\tilde{k}_{1,1}$)
   $\tilde{k}_{0,0}$= DH($bk_{1,0}$, $\tilde{k}_{1,1}$)



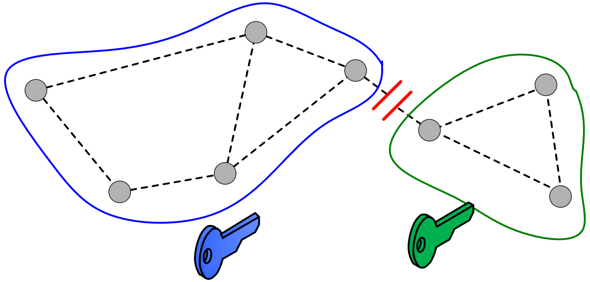
2. ……

Chapter 1: TGDH in Mobile Ad-hoc Networks

- Network fusion
  - The network fusion is supported by a key tree merge
    → Generation of a new common root
    → Inserting the smaller tree in the bigger tree



- Network separation
  - A key tree split is used in order to support a network separation
    → Multiple REJOIN if the separation can not be handled by a tree split



CHAPTER 1: SUMMARY

- Confidentiality in distributed systems is provided by encryption E(m,k) with a group key k

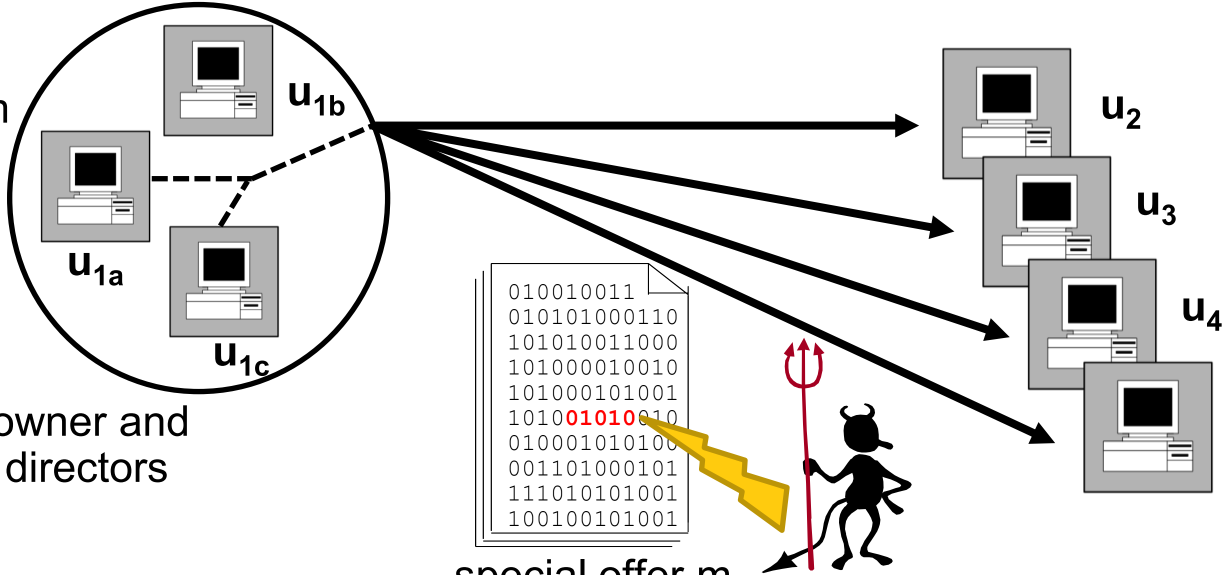
- Key management is a fundamental component for providing confidentiality

- Distributed group key management, e.g. TGDH, is based on the iterative application of the Diffie-Hellman algorithm

Chapter 2: Learning Objective

- We want to understand protection mechanisms against content modifications where "all" or "a certain number" of users contribute

Example:
chat system



company owner and managing directors

special offer m

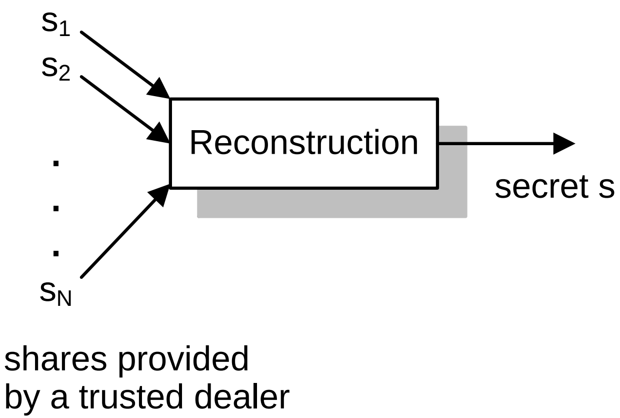
XMPP

[psaik]

Chapter 2: Secret Sharing for Distributed Signature Generation/Verification

- The secret signature key may be too important to be kept by a single user

- Secret sharing can be also be used for distributed signature calculation
  - "Trusted" dealer for initialization
- Linear secret sharing
  - n-out-of-n users can reconstruct the secret
  - Additive reconstruction: $s = s_1 + \ldots + s_n$
- Threshold secret sharing (k,n)
  - k-out-of-n users can reconstruct the secret
  - Reconstruction by Lagrange interpolation



Chapter 2: Distributed Integrity Protection- n-out-of-n Signature Algorithm with RSA

- The user u_i , i = 1,…,n jointly calculates the signature of a message m
- Public key e, partial secret keys d_i of the user u_i, modulus n (ñ = p·q, p,q large primes)

Signing: RSA encryption of the hash (with the partial keys)

$$M = H(m) < \tilde{n}$$

$$s_i = M^{d_i} \bmod \tilde{n}; \quad \text{user } u_i, \quad i = 1, \dots, n$$

Verification: RSA decryption of the hash

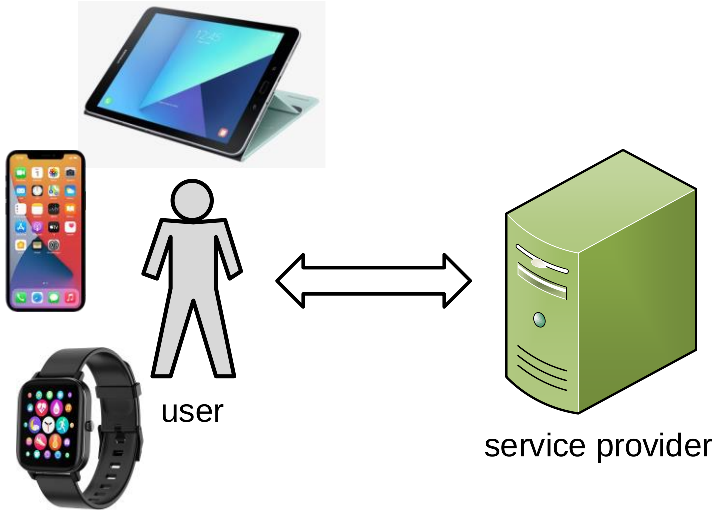
$$s = \prod_{i=1}^{n} s_i$$

$$H(m) \stackrel{?}{=} H'(m) = s^e \bmod \tilde{n}$$

signature reconstruction

Chapter 2: Example - Collaborative Entity Authentication

- Collaborative authentication an alternative approach for remote access to sensitive information and services
  - A private key is shared among the personal devices, e.g. smartphones, tables, smartwatches, fitbit devices of a user
  - The devices must collaborate to authenticate a user to remote sensitive services



CHAPTER 2: EXAMPLE - E-VOTING SYSTEM

- Concept
  - Voter with vote x
  - Multiple authorities A_i $(i = 1,\dots,n)$ confirm and count the votes

- Basic algorithms
  - n-out-of-n RSA signature
  - Blind signature

Chapter 2: Blind Signature

- Authority signs a message m without knowing the content
  - The user "blinds" a message m with random number r (and the authority‘s public key)
  - The authority calculates the signature of the blinded message
  - The user can “unblind" the signature of the blinded message using r^{-1}

Example: Blind RSA Signature Algorithm

Authority: secret key d, public key e, modulus ñ (ñ = p·q, p,q large primes)

User: $r = Rnd(), \gcd(r, ñ) = 1,$

$m´ = blind_e(m, r) \Rightarrow m´ = r^e \cdot m \bmod ñ$ ($r^e \bmod ñ$ is blinding factor)

Authority: $s´ = Sig(blind_e(m, r), d) \Rightarrow s´ = r \cdot m^d \bmod ñ$

User: $Unblind(s´,r) \Rightarrow s = s´ \cdot r^{-1} \bmod ñ$

Chapter 2: Example - E-Voting System (Linear Secret Sharing, Blind Signatures) (1)

- Voter with vote x
- Authority A_i, i = 1,…,n with secret (private) keys (d_i, ñ), public key (e, ñ)

1. Voting, blinding, informing the authorities

x = vote, r = Rnd()

$x' = \text{Blind}_e(x, r) \Rightarrow x' = r^e \cdot x \bmod \tilde{n}$

$x'\colon V \rightarrow A_i$, i = 1,…,n

2. Signing the blinded vote by the authorities

$s_i' = \text{Sig}(x', d_i)$

$\Rightarrow s_i' = x'^{d_i} \bmod \tilde{n}$

$s_i'\colon V \leftarrow A_i$, i = 1,…,n

vote and sign

Chapter 2: Example - E-Voting System (Linear Secret Sharing, Blind Signatures) (2)

- Voter with vote x
- Authority A_i, i = 1,…,n with secret (private) key (d_i, ñ), public key (e, ñ)

3. Unblinding signature by the voter

$$s' = \prod_{i=1}^{n} s'_i$$

s = Unblind(s´,r) => $s = s' \cdot r^{-1} \bmod ñ$

s, x: $V \rightarrow A_i$, $i \in \{1,\ldots,n\}$

anonymous communication

4. Verification, vote counting by an authority

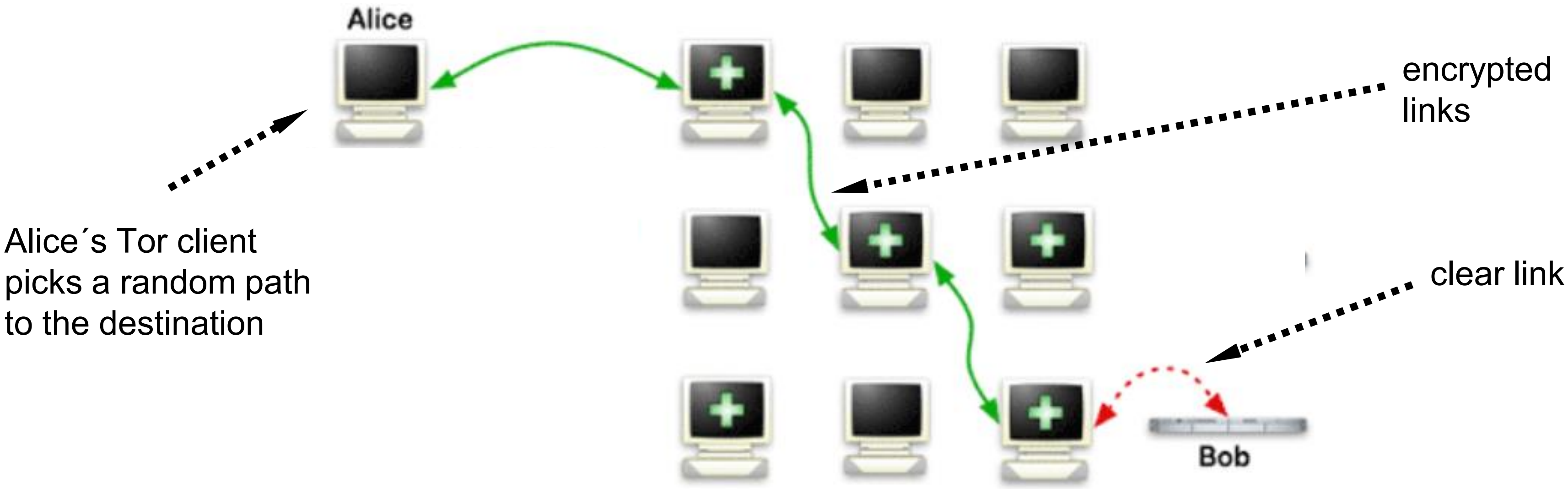
A_i, $i \in \{1,\ldots,n\}$

$s \overset{?}{=} Ver(x, e)$, Count(x)

deliver, verify and count

Chapter 2: Remark – Anonymous Internet Communication

- TOR (http://tor.eff.org)
  - Second-generation onion routing network
  - Specifically designed for low-latency anonymous Internet communication



Chapter 2: Threshold Secret Sharing (1)

- Shamir`s Secret Sharing is a (k, n) threshold algorithm
  - "Trusted" dealer
  - Users u_i (i=1,…n)
  - Secret a_0 (e.g. <u>secret key</u>)

ring mod m (m>n, e.g. m=2^k)
usually
field mod p (p=2q+1, p,q=primes)

- A dealer chooses randomly k-1 positive integers from ($\mathbb{Z}_p$, +, ·) and create a polynomial f(x) which contains the secret

$$f(x) = a_0 + a_1 x + \ldots + a_{k-1} x^{k-1} \bmod p$$

- A dealer gives every user u_i a share (x_i, s_i=f(x_i) mod p)
  - The input x_i is public (mostly x_i=i with i = user id)
  - The output s_i must be kept secret

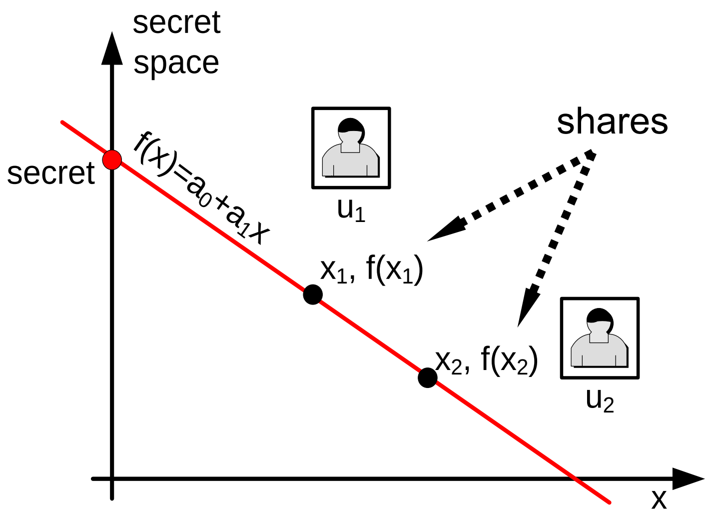
Chapter 2: Threshold Secret Sharing (2)

- The secret s can be reconstructed from every subset of k share

- Restoring the secret a_0 is done by Lagrange interpolation with k points (x_i, y_i) where $y_i = s_i = f(x_i) \bmod p$

$$f(0) = a_0 = \sum s_i \cdot \boxed{\prod_{i \neq j} \frac{-x_j}{x_i - x_j} \bmod p}$$

Lagrange interpolation polynomial with x=0



Restoring a secret in a (2, n) secret sharing

CHAPTER 2: EXAMPLE - THRESHOLD SIGNATURES WITHIN DNSSEC

- The Domain Name System (DNS) associates information, e.g. IP address, with domain names
- A Resource Record (RR) is the basic information unit
- DNSSEC is the Domain Name System with security features
  - In the Resource Record SIG a RR signature is stored

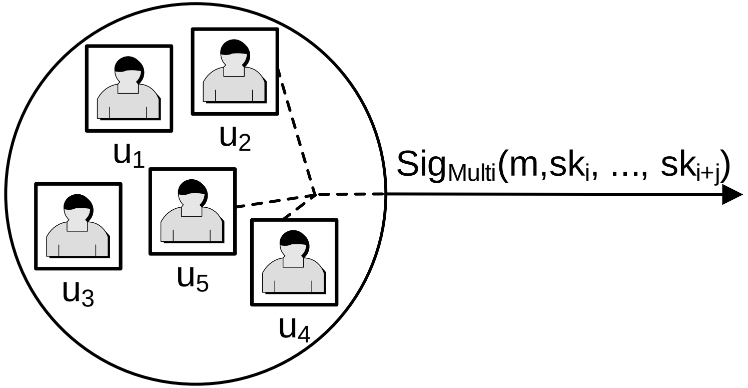
```
www.example.de.   1285  A  1.2.3.4
www.example.de.   1285  IN                ; class of the RR
                        RRSIG             ; RR of the type RRSIG
                        A                 ; type of the signed RR
                        3                 ; „signature" algorithm
                        3                 ; number of name components
                        1285
                        ( 20040327122207
                          20040226122207
                          22004           ; key tag
                          example.de.     ; signer name
                          BM=)8.BfsWf&%X  ; signature )
```

Resource Record SIG for www.example.de

- The secret signature key for the root zone is split into 7 pieces
- Signatures are possible with 5 pieces , e.g. by Threshold Elgamal Signature algorithm

CHAPTER 2: MULTI-SIGNATURE

- A multi-signature algorithm enables multiple users to sign a message
  - A receiver can verify a multi-signature but must know the signer´s identities/keys
  - A user can be easily added to the group of possible signers
- A multi-signature is shorter than the collection of individual signatures

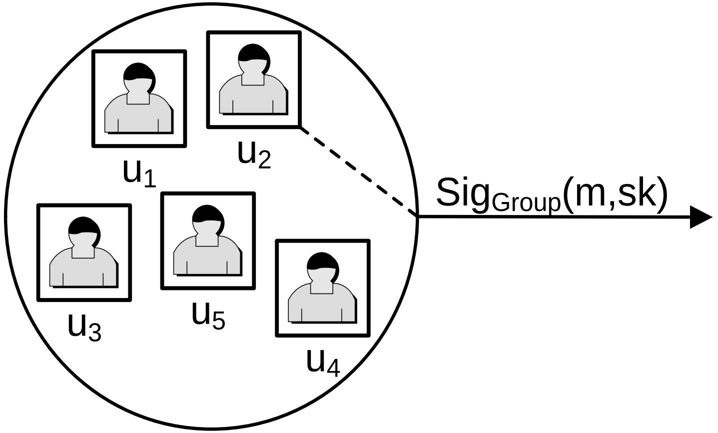


e.g. company owner and managing directors creates and signs a contract

- Example: Guillou-Quisquater signature algorithm

CHAPTER 2: GROUP SIGNATURE

- A group signature procedure enables a user to sign a message m on behalf of the group
  - A receiver can verify the group signature but must not know the identity of the signer
  - The signer's identity can be revealed
- Group signature scheme consist of 6 algorithms
- <u>Example</u>: Elgamal Signature algorithm with Group Controller contributions



e.g. sales department creates and signs an offer

CHAPTER 2: SUMMARY

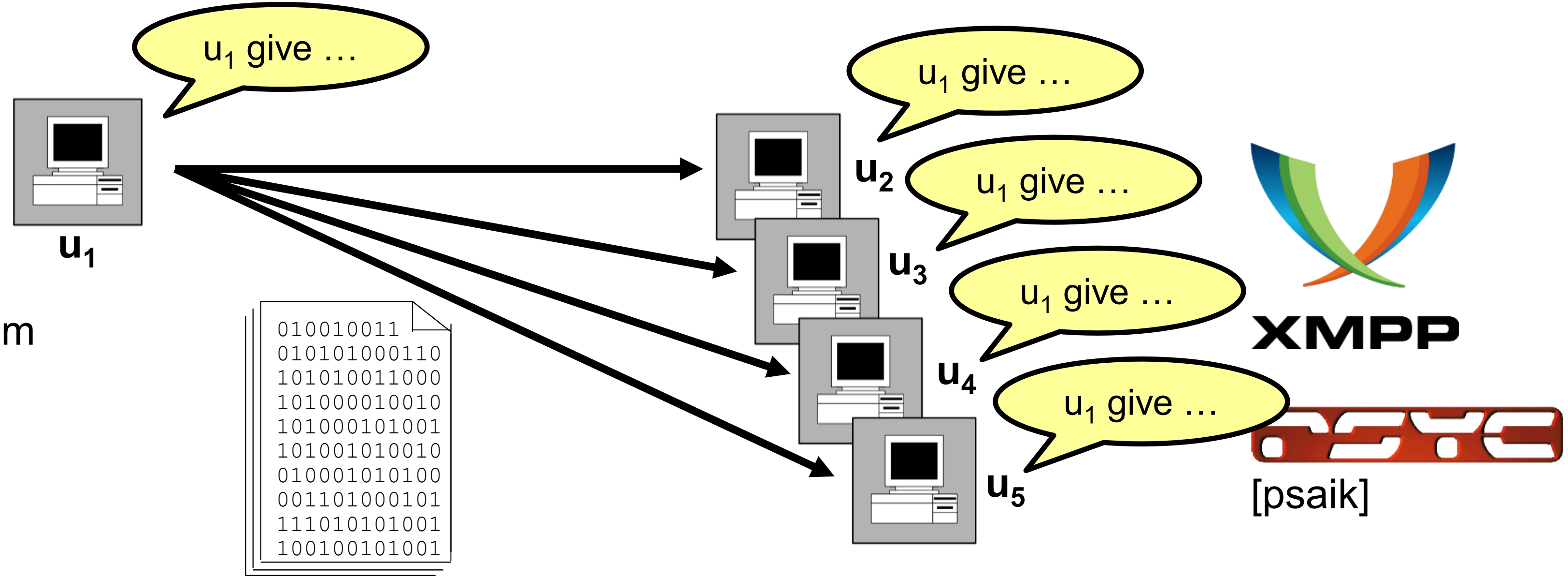
- Linear sharing provides a n-out-of-n sharing of a secret
- Within a (k,n) threshold sharing k of n potential users can reconstruct the secret
- Shamir`s Secret Sharing is a (k, n)-threshold algorithm
- Secret sharing can be also be used for distributed signature calculation

Chapter 3: Learning Objective

- We want to understand the documentation of transactions in distributed system with no trusted third party

Example:

chat system

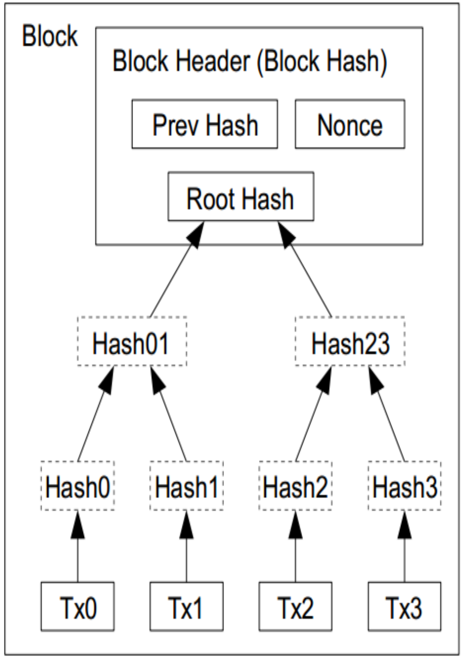


message: 10 € for all recipients

Chapter 3: Bitcoin Blockchain

- Bitcoin is a (crypto-)currency based on a blockchain-based ledger
- A block of the chain contains of a block header and a list of transactions

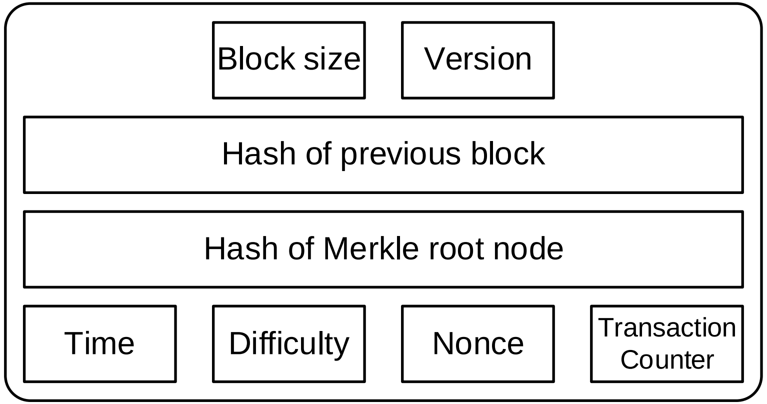
- A Merkle tree is constructed in which every leaf contains the hash of one transaction included in the block
  - The root hash of the tree is included in the block header
  - Merkle tree enables an efficient proof of membership



Transactions Hashed in a Merkle Tree

Chapter 3: Bitcoin Block Details

- The SHA-256 hash of the previous block that creates the chaining
  - The hash used for chaining is calculated from the version until the nonce field of the block header
- The hash of the root node of a Merkle tree with the transactions
- The nonce is required for the consensus mechanism
  - The difficulty is a parameter



Block header

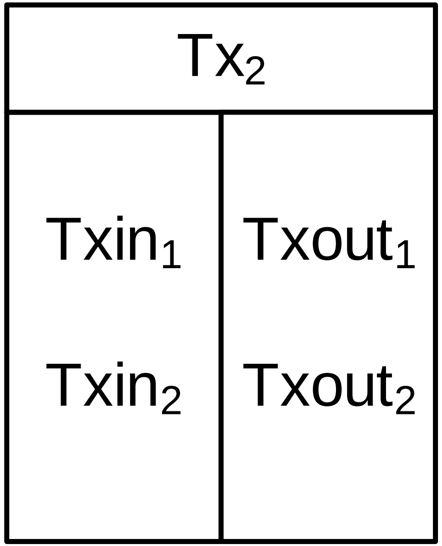
CHAPTER 3: ACCOUNT-BASED LEDGER VS TRANSACTION-BASED LEDGER

- Account-based ledger
  - Focuses on accounts and their balances
  - Stores balances of accounts
- Transaction-based ledger
  - Focuses on individual transactions and their documentation
  - Stores a sequence of transactions

- The Bitcoin blockchain is a transaction-based ledger
  - Documentation of <u>transactions</u> is more efficient than tracking balances of accounts
  - Definition of conditional transactions using Bitcoin Script is possible

Chapter 3: Transactions in Bitcoin

- Transactions (Tx) have a number of inputs and a number of outputs
  - Inputs (Txin): Former outputs, that are being consumed
  - Outputs (Txout): Creation of new coins and transfer of coins
- Each transaction has a unique identifier (TxID)
  - Each transaction output has a unique identifier TxID[#txout]
  - Example: 5[1] = Second Txout of the sixth transaction
- Transactions are signed by the creator



Transaction Tx2 with two inputs and two outputs

Chapter 3: Example - Transactions in Bitcoin

	0	Txin: Ø Txout: 25.0 → Alice
Tx of Alice	1	Txin: 0[0] Txout: 17.0 → Bob, 8.0 → Alice signed by Alice
Tx of Bob	2	Txin: 1[0] Txout: 8.0 → Carol, 9.0 → Bob signed by Bob
Tx of Alice	3	Txin: 1[1] Txout: 6.0 → David, 2.0 → Alice signed by Alice

Transactions where new coins are created have no Txin

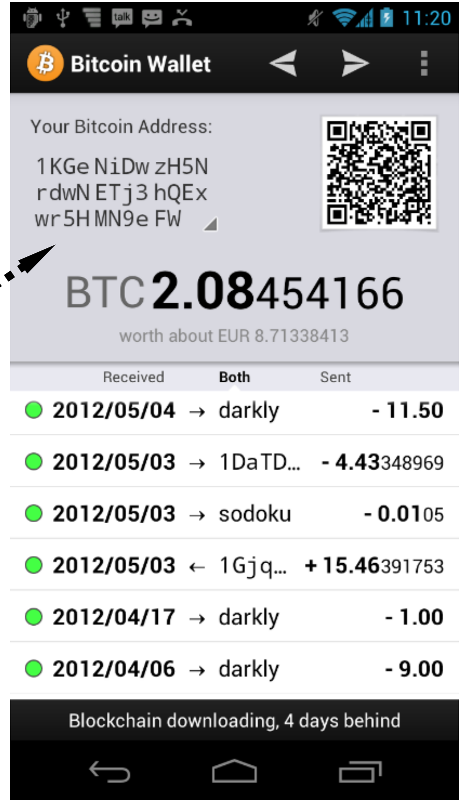
Either all or none of the coins have to be consumed

Hashes of the public keys are used for addressing

Manage a list of unspent transaction outputs (UTXO)

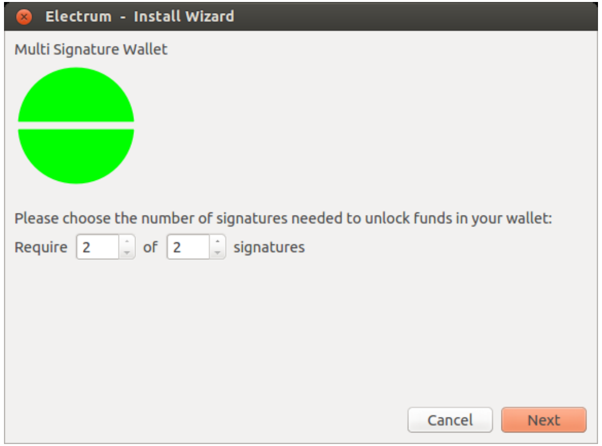
Chapter 3: Bitcoin Wallet

- A wallet program is needed to send and receive bitcoins
- The wallet program creates the public/private key pairs
- A 160-bit hash of the public key is the corresponding Bitcoin address
- Provides a list of transactions of the Bitcoin address



CHAPTER 3: MULTI-SIGNATURE ADDRESS

- Multi-signature address are used to collaboratively control transactions
- To spend coins from a n-of-m address, n cosigners need to sign a transaction
  - Sharing of the secret signature key
  - Using a multi-signature algorithm
- A partially signed transactions has to be transferred to the cosigner wallets
  - Manual transfer (e.g. via file on a usb stick)
  - Cosigner Pool Plugin for electronic exchange



Create a 2 of 2 multi-signature **Electrum wallet**

CHAPTER 3: BITCOIN OPERATION

- Actors: user, mining node
- <u>Step 1</u>: Users create, sign and then broadcast their transactions
  - The mining nodes cache them in the memory pool (mempool)
- <u>Step 2</u>: A mining node creates a candidate (new) block with <u>verified</u> transactions
  - All transactions of block must be authentic (digital signature checking)
  - All transactions of block must be valid (transaction checking)
- <u>Step 3</u>: The candidate block is shared across the Bitcoin network
- <u>Step 4</u>: All mining nodes try to solve a hard search puzzle for the candidate block
- <u>Step 5</u>: A mining node who solves the hard search puzzle broadcasts the valid block
- <u>Step 6</u>: Other mining nodes accept the new block and using the new block hash as the previous hash for the next candidate block

Chapter 3: Step 6 - Distributed Consensus of the Mining Nodes

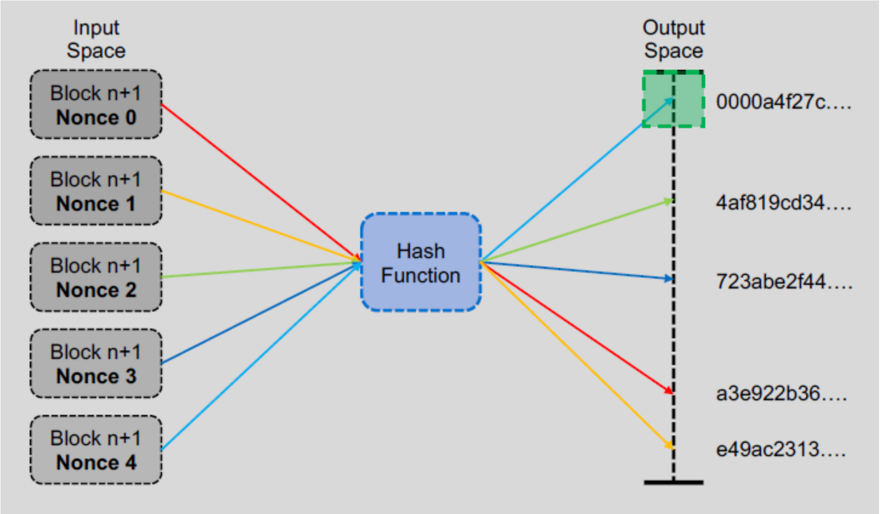
- The Bitcoin network has to agree on the information in the blockchain
  - Which of the proposed transactions are valid?
  - In which order do the transactions appear in the blockchain?

- The Bitcoin network selects a random mining node to propose a valid block using proof-of-work (PoW)
  - Proof-of-work means solve a hard search puzzle
  - The process of creating a valid block is also called mining

- The proof-of-work is used as decentralized consensus mechanism on the next block

Chapter 3: Search Puzzle - Proof-of-work

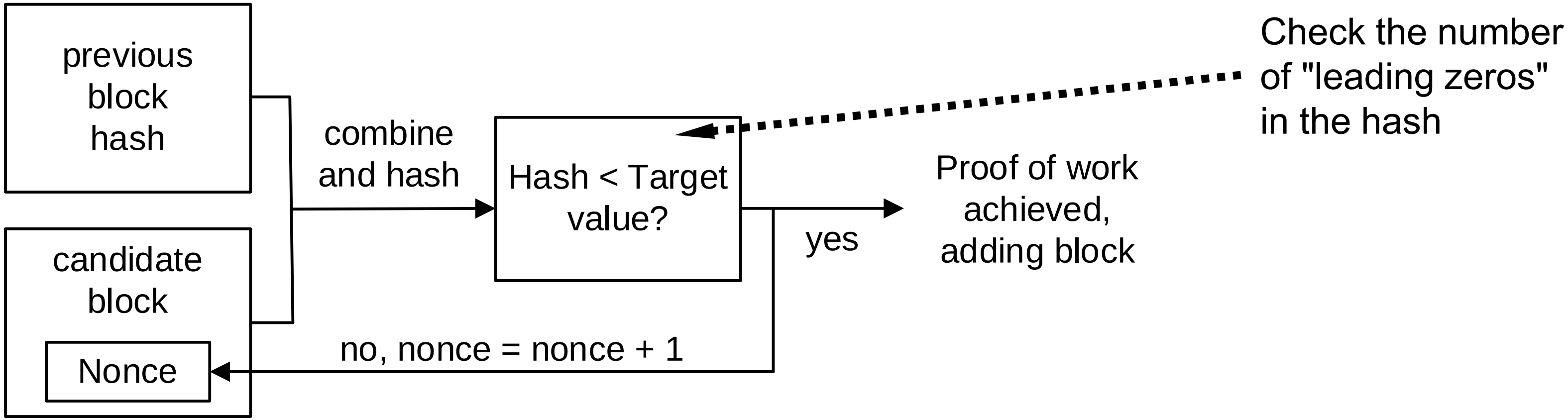
- A search puzzle is a mathematical problem which requires searching a large space to find a solution
- There are no shortcuts in finding the solution
- Solving the puzzle requires finding an input so that the output falls within the set Y
- The target set Y is defined as {0, 1, …, d}
  - The puzzle has to find an input that the result of the hash function is smaller than the <u>difficulty d</u>



Search puzzle to find a nonce so that the hash result is smaller then d

Chapter 3: Proof-of-Work in Bitcoin

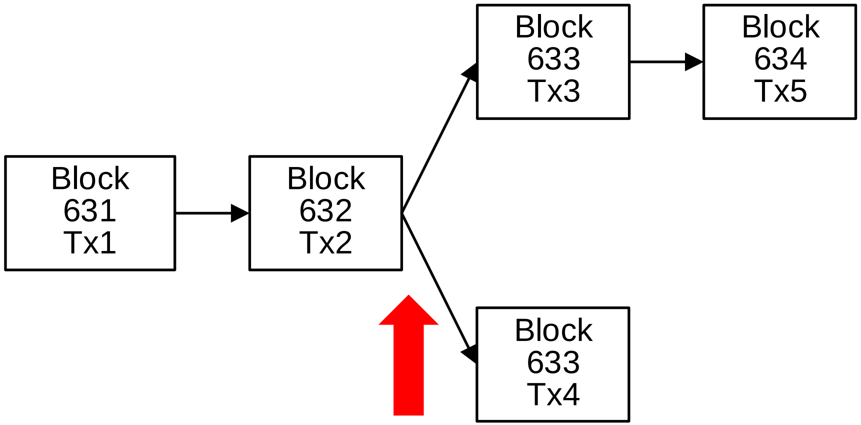
- Proof-of-work has to find a nonce so that the hash result is smaller then d



nonce = 0 0 leading zeros = 4c8f1205f49e70248939df9c7b7…
nonce = 12 1 leading zero = **0**5017256be77ad2985b36e75e…
nonce = 112 2 leading zeros = **00**ae7e0956382f55567d0ed931…
nonce = 3728 3 leading zeros = **000**b5a6cfc0f076cd81ed3a606b…

CHAPTER 3: BRANCHING OF A BLOCKCHAIN-BASED DISTRIBUTED LEDGER

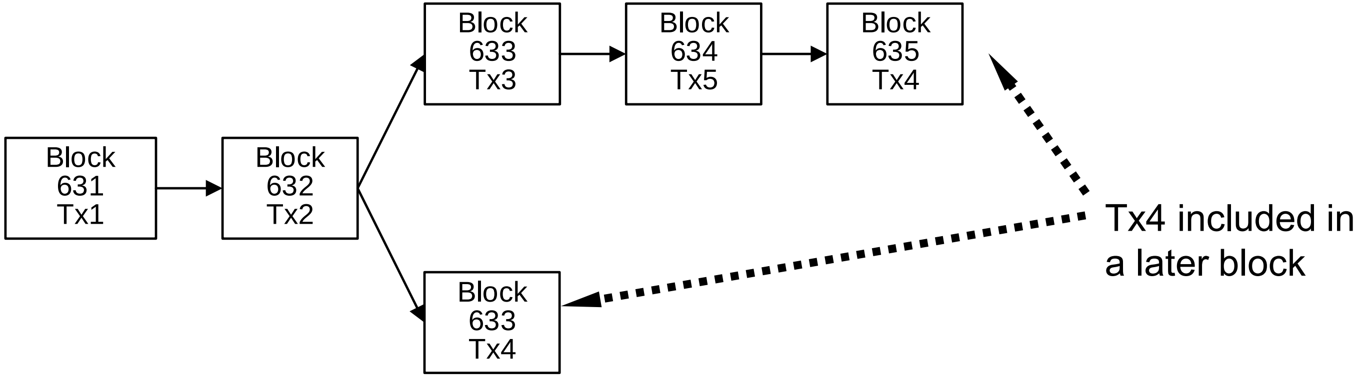
- When two mines found two valid blocks at the same time a branching of the blockchain is possible
- If branching occurs, the longest chain is accept as the valid version because the longest chain took the most effort to build



Blockchain fork at block 632

Chapter 3: Transactions in Orphan Blocks

- A valid block that has been broadcasted but has not been included in the longest blockchain is called an orphan block
- Including a block causes the removal of the included transactions from the mempool
  - Unconfirmed transactions are still stored in the mempool
- The transactions in an orphan block are considered included later



Transactions in an orphan block

Chapter 3: Summary

- To document certain transactions in a group the distributed ledger technology can be used

- Blockchain-based distributed ledger operation consists of six steps

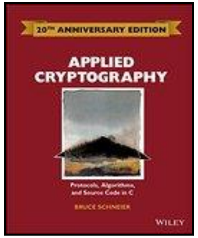
- Bitcoin is a (crypto-)currency based on a Blockchain-based public distributed ledger

Concluding Remark

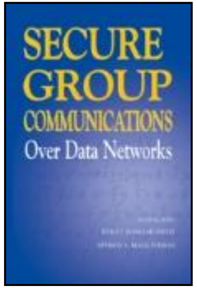
- This was a high-speed walk through selected topics of the lecture on cyber security of distributed and resource limited systems
- If you want to know more, visit the lecture (e.g. SS 2025)
  - Kill chain, defense strategies for distributed systems
  - Key management
  - Distributed signatures
  - Cyber resilience in the case of partially successful cyber attacks
  - Distributed ledger technology (blockchain)
  - IoT security
  - Cyber security in software-defined networks
  - Artificial intelligence in cyber security

Books

- B. Schneier
  Applied Cryptography
  John Wiley & Sons, 1994



- X. Zou, B. Ramamurthy, M. S. Spyros
  Secure Group Communications Over Data Networks
  Springer-Verlag New York, 2005



- A. Perrig
  Secure Broadcast Communication in Wired and Wireless Networks
  Springer, 2003

Literature

- R. M. Needham, M. D. Schroeder
  Using encryption for authentication in large networks of computers
  Communications of the ACM, Volume 21, Issue 12, 1978
- A. Shamir
  How to Share a Secret
  Communications of the ACM, Volume 22, Issue 11, 1979
- M. Kucharczyk
  Blind Signatures in Electronic Voting Systems
  Communications in Computer and Information Science, 79, 1970
- Y. Desmedt
  Threshold cryptosystems
  Advances in Cryptology, AUSCRYPT '92, 1992

Exercise

- Exercise sheet “Cyber security of distributed and resource limited systems”
- Issue
  - Via Sciebo
- Discussion
  - October 31, 2024, 12:30 - 01:15 p.m.
  - Please complete the tasks so that you can show the solution on the board at the next exercise and submit your solution as pdf to the gitlab
- Tasks
  - Task 1: Explain in your own words
  - Task 2: Distributed group key management
  - Task 3: Secret Sharing

Opinion poll

Lightning Surveys

Institute of Computer Science IV, University of Bonn in cooperation with the Fraunhofer FKIE

Dr. Thorsten Aurisch
FKIE / KOM
Zanderstraße 5
D-53177 Bonn, Germany
Phone: +49 (228) 50212-505
Mail: thorsten.aurisch@fkie.fraunhofer.de