QoS and Security

Mestrado em Engenharia de Computadores e Telemática

2021/2022

Quality of Service

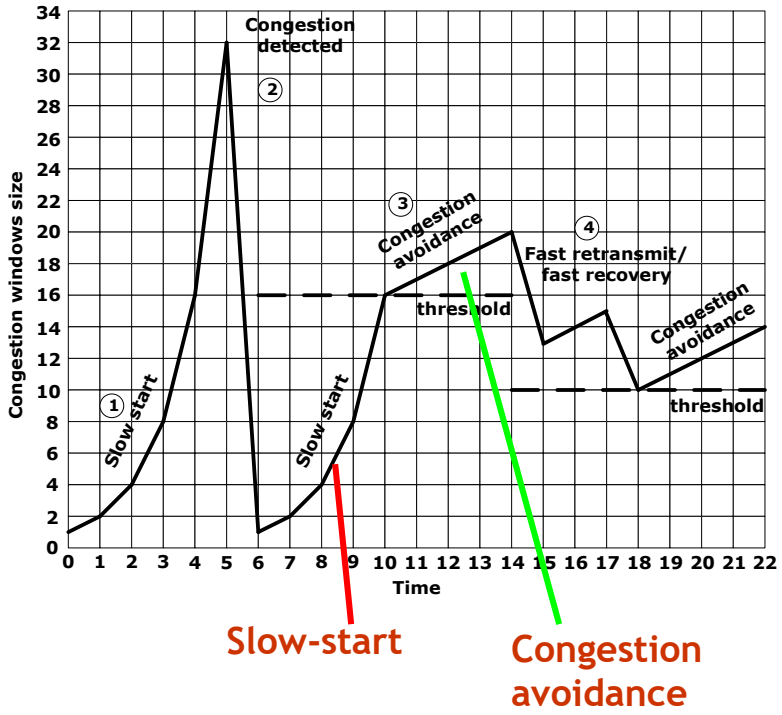
TCP- and UDP-based applications

Problem: Evaluate TCP

- Why does TCP perform *poorly* in ad-hoc/vehicular networks?
  - Developed for wire-line networks
  - Assume all losses are due to congestion
- Many TCP variants have been proposed
  - How good are they? Are they sufficient?
- Are there any other alternatives?
  - Are non-TCP protocols the solution?

Overview of TCP concepts

- Conventional TCP: Tahoe, Reno, New-Reno
- Sending rate is controlled by
  - Congestion window (*cwnd*): limits the # of packets in flight
  - Slow-start threshold (*ssthresh*): when congestion avoidance starts
- Loss detection
  - 3 duplicate ACKs (faster, more efficient)
  - Retransmission timer expires (slower, less efficient)
- Overview of congestion control mechanisms
  - Slow-start phase: *cwnd* starts from 1 and increases exponentially
  - Congestion avoidance (CA): *cwnd* increases linearly
  - Fast retransmit and fast recovery: Triggered by 3 duplicate ACKs



What is different in ad-hoc?

1. Mobility
   - Route stability and availability
2. High bit error rate
   - Packets can be lost due to “noise”
3. Unpredictability/Variability
   - Difficult to estimate time-out, RTT, bandwidth
4. Contention: packets compete for airtime
   - Intra-flow and inter-flow contentions
5. Long connections have poor performance
   - ❑ More than 4 hops throughput drops dramatically

Why does TCP fail in ad-hoc networks?

- TCP misinterprets route failures as congestion
  - Effects: Reduce sending rate
  - Buffered packets (Data and ACKs) at intermediate nodes are dropped
  - Sender encounters timeouts
    - Under prolonged disconnection, a series of timeouts may be encountered
- TCP misinterprets wireless errors as congestion
  - Effects: Incorrect execution of congestion control → performance drops
  - Wireless channel is error-prone compared to wireline
    - Fading, interference, noise

Why does TCP fail in ad-hoc networks?

- Delay spike causes TCP to invoke unnecessary retransmissions
  - Effects: Performance drops and many unnecessary retransmissions exist
  - Variability: Spikes are not uncommon here
  - Spikes throw off parameter estimation and tuning
    - RTO, window size, slow-start threshold
- Inefficiency due to the loss of retransmitted packet
  - Effects: performance drops significantly under high loss environment
  - Loosing a *retransmitted* packet hurts
    - TCP can recover from one loss (fast retransmission)
  - Wired networks: packet loss rate is low
  - Here, high packet loss makes the problem significant

Classification of Transport protocols

- TCP variants try to improve the performance by the following ways
  - Estimating the available bandwidth
  - Exploiting buffering capability

TCP-Vegas

- TCP Vegas senses the congestion in the network before any packet loss occurs, and instantly it decreases the window size. So, TCP Vegas handles the congestion without any packet loss occur.
- Uses *diff* = *expected rate* - *actual rate* to regulate the sending rate
- *Expected rate* = window size divided by baseRTT,
- *Actual rate* = window size divided by currentRTT
- baseRTT = It is the minimum RTT measured so far.

- *Expected rate* is always larger then the *Actual rate*

TCP-Vegas

- Rate-based congestion control
  - diff = expected rate – actual rate
  - If diff < a, Vegas increases *cwnd* linearly
  - If diff > b, Vegas decreases *cwnd* linearly
  - If a< diff < b, Vegas keeps *cwnd* unchanged
- Modified slow-start
  - Allows *cwnd* to grow exponentially only in every other *RTT*
  - If diff > c, Vegas switches from slow-start to congestion avoidance
- New retransmission
  - Reads and records transmission time
  - When DUPACK arrives, checks if it is expired
  - Retransmits without waiting for third DUPACK

Exploiting Buffering Capabilities

- Buffering capability and Sequence information (TCP-BuS)
  - Uses explicit route failure notification (routing error) to detect route failures
  - When route failure occurs, intermediate nodes buffer the pending packets and TCP sender doubles the retransmission timeout (*RTO*) value
  - Makes use of special messages such as localized query and reply to find a path
    - Messages modified to carry TCP connection and segment information
  - Avoid timeouts and unnecessary retransmissions
  - Pro: Reduce the number of timeout events → reduce the number of retransmissions
  - Con: Require assistance from intermediate nodes
    - Special routing protocol is used

QoS in UDP: trade-offs

- In IntServ-like mechanisms
  - Application specifies traffic and QoS parameters
  - A resource reservation protocol estimates and reserves sufficient resources at each node on the path
- In a multi-hop wireless network, however
  - It is hard to estimate available resources
    - Shared medium
      - All traffic in the transmission range reduces available bandwidth
    - Dynamic bandwidth due to node mobility and contention
  - It is hard to do resource reservation
    - Shared medium reservation requires global coordination
    - Violations can occur as bandwidth fluctuates
  - Resource reservation is pinned to a route
    - Must be redone whenever route changes
    - This might NOT be a good idea
      - Time for recovering a route and reserve resources in the new route

QoS in UDP: trade-offs

- In DiffServ-like mechanisms
  - Application chooses a class of service
  - Network needs admission control to avoid class overload
  - Examples
    - Assured Forwarding Per-hop behavior assures per-hop throughput
    - Expedited Forwarding Per-hop behavior assures per-hop low delay
- In a multi-hop wireless network, however
  - It is hard to do admission control
    - Flows do not go through common ingress nodes
  - It is hard to maintain assurances
    - Flow distribution varies as routes change
    - Bandwidth fluctuates

QoS Routing

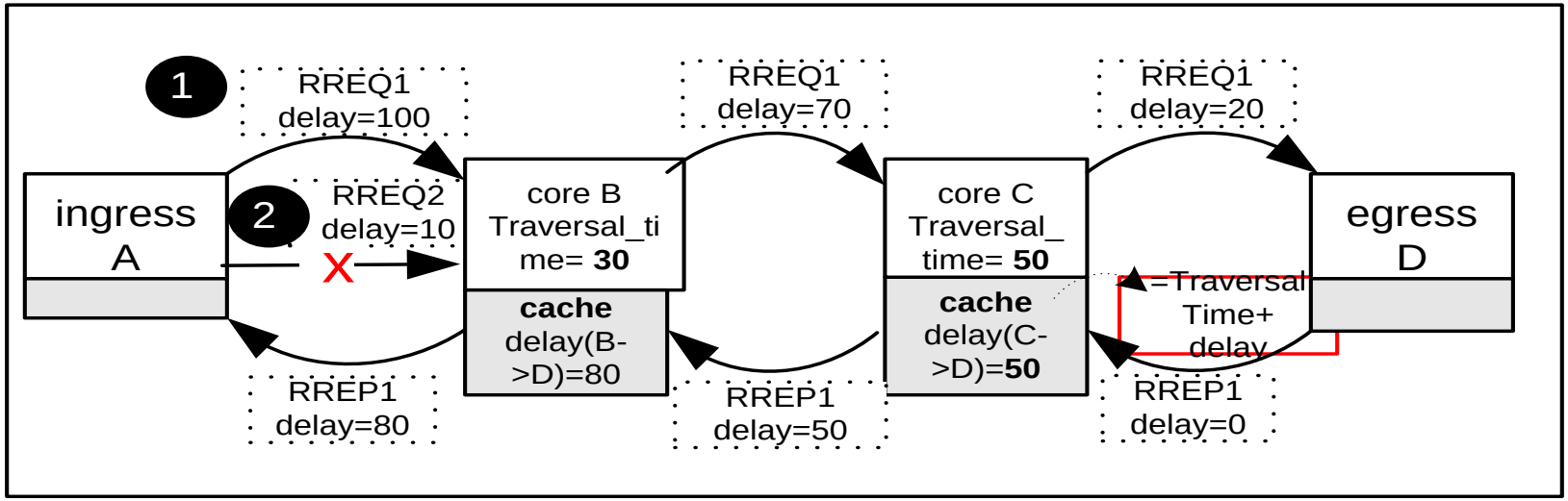
- Routing is an essential component for QoS
- It can inform a source node of the bandwidth and QoS availability of a destination node and of the path to the destination node
- Add QoS requirements in routing metrics
  - Difficult to route maintenance
  - Overhead of QoS routing
  - Reserved resources may not be guaranteed
  - Needs to be responsive to nodes mobility

QoS for AODV

- Add extensions to the route messages (RREQ, RREP)
- Node that receives a RREQ + QoS extension must be able to meet the service requirement in order to rebroadcast the RREQ (if not in cache)
- In order to handle the QoS extensions, some changes need to be done on the routing tables
- AODV current fields
  - Destination sequence number, Interface, Hop count, Next hop, List of precursors
- AODV new fields (4 new fields)
  - Maximum delay
  - Minimum available bandwidth
  - List of sources requesting delay guarantees
  - List of sources requesting bandwidth guarantees

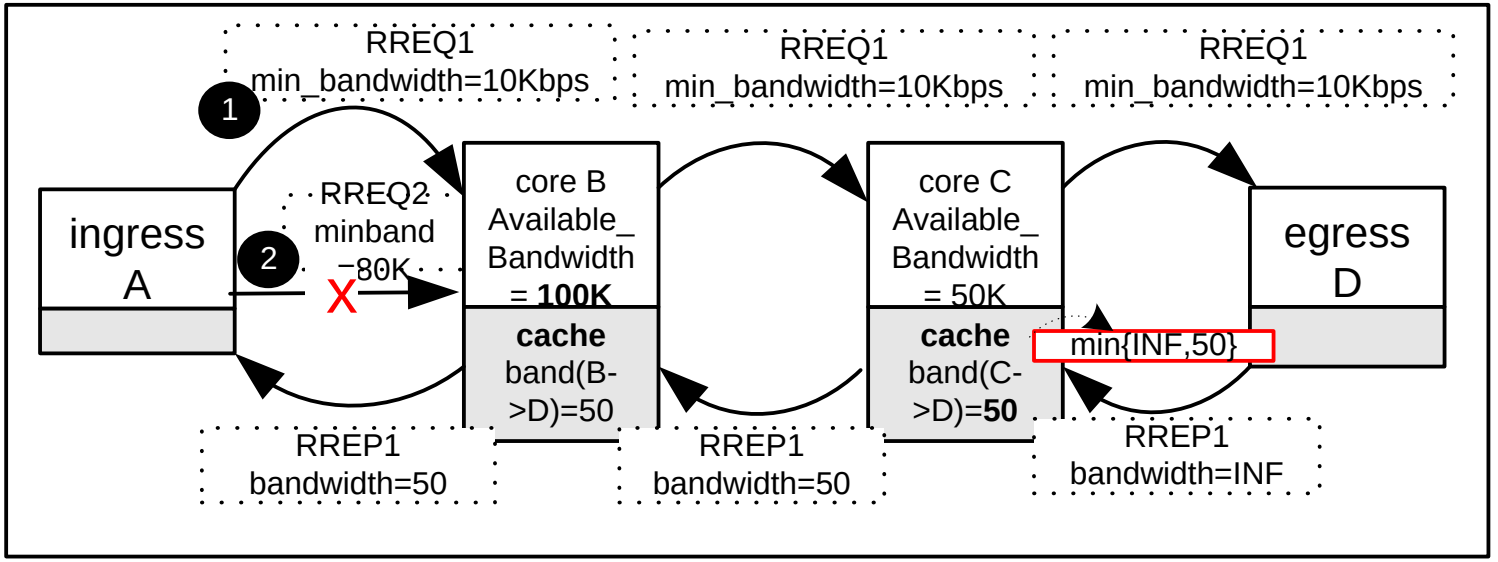
QoS for AODV- Delay

- Handling Delay
  - Maximum delay extension
  - List of sources requesting delay guarantees
  - Used during route discovery process



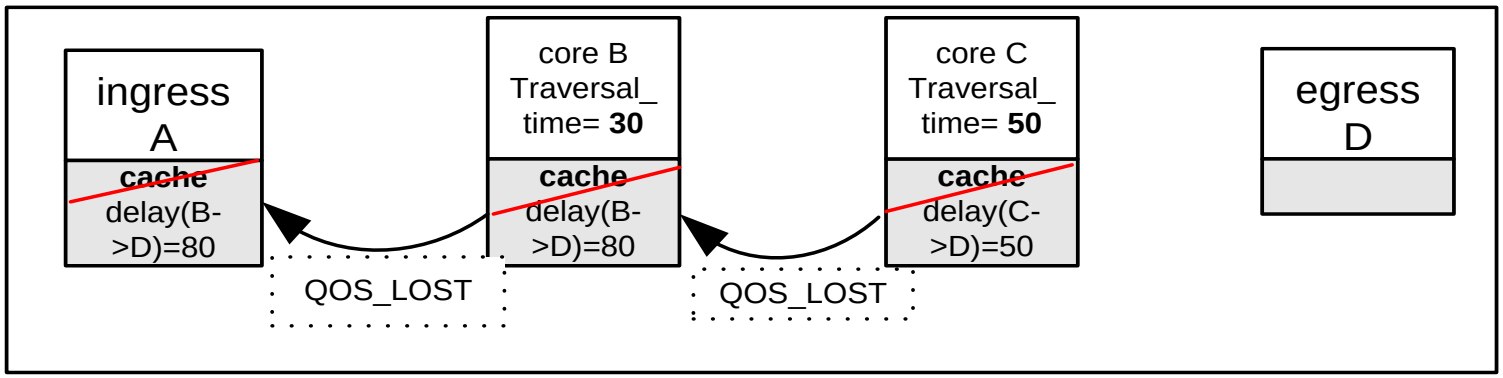
QoS for AODV- Bandwidth

- Handling Bandwidth
  - Similar to delay requests
  - RREQ can include both types
  - Utilized during route discovery process



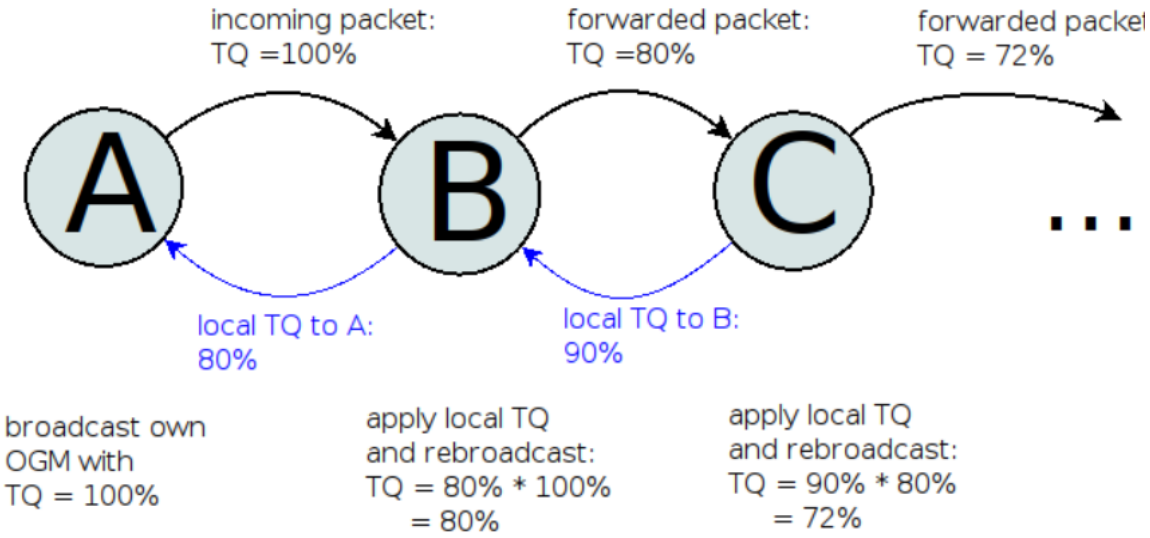
QoS for AODV- Loosing QoS

- Loosing QoS parameters
  - If, after establishment, a node detects that the QoS cannot be maintained any more, it originates an ICMP QoS_LOST message to all depending nodes
    - Reason to keep a List of Sources requesting delay/bandwidth guarantees
- Reasons for loosing QoS parameters
  - Increased load of a node
  - Why would a node take over more jobs than it can handle?



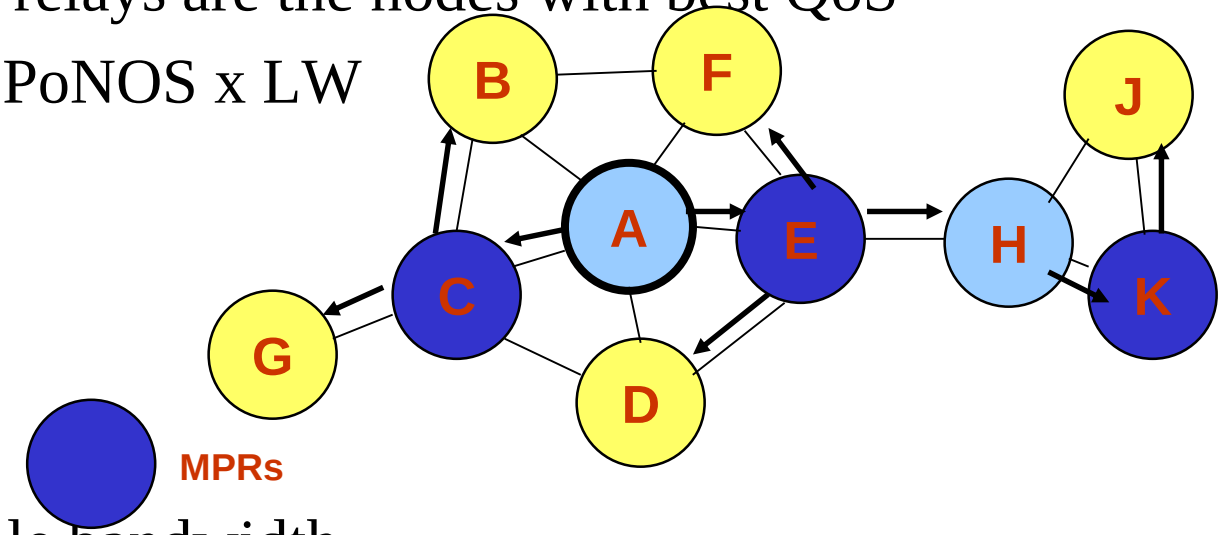
Transmission Quality (Batman v.3)

- To add the local link quality in the TQ value the following calculation is performed:
- TQ = TQ_{incoming} * TQ_{local}
- Example: Node A broadcasts the packet with TQ max. Node B receives it, applies the TQ calculation and rebroadcasts it. When node C gets the packet it knows about the transmit quality towards node A.



QoS-OLSR

- Multi-Point relays are the nodes with best QoS
- QoS=AB x PoNOS x LW



- AB: available bandwidth
- PoNOS: Percentage of Neighbors on Other Street, presents the level of neighbors' diversity.
- LW: Lane Weight is used to favor the selection of MPRs from lanes that carry the majority of the traffic flow to increase their stability.

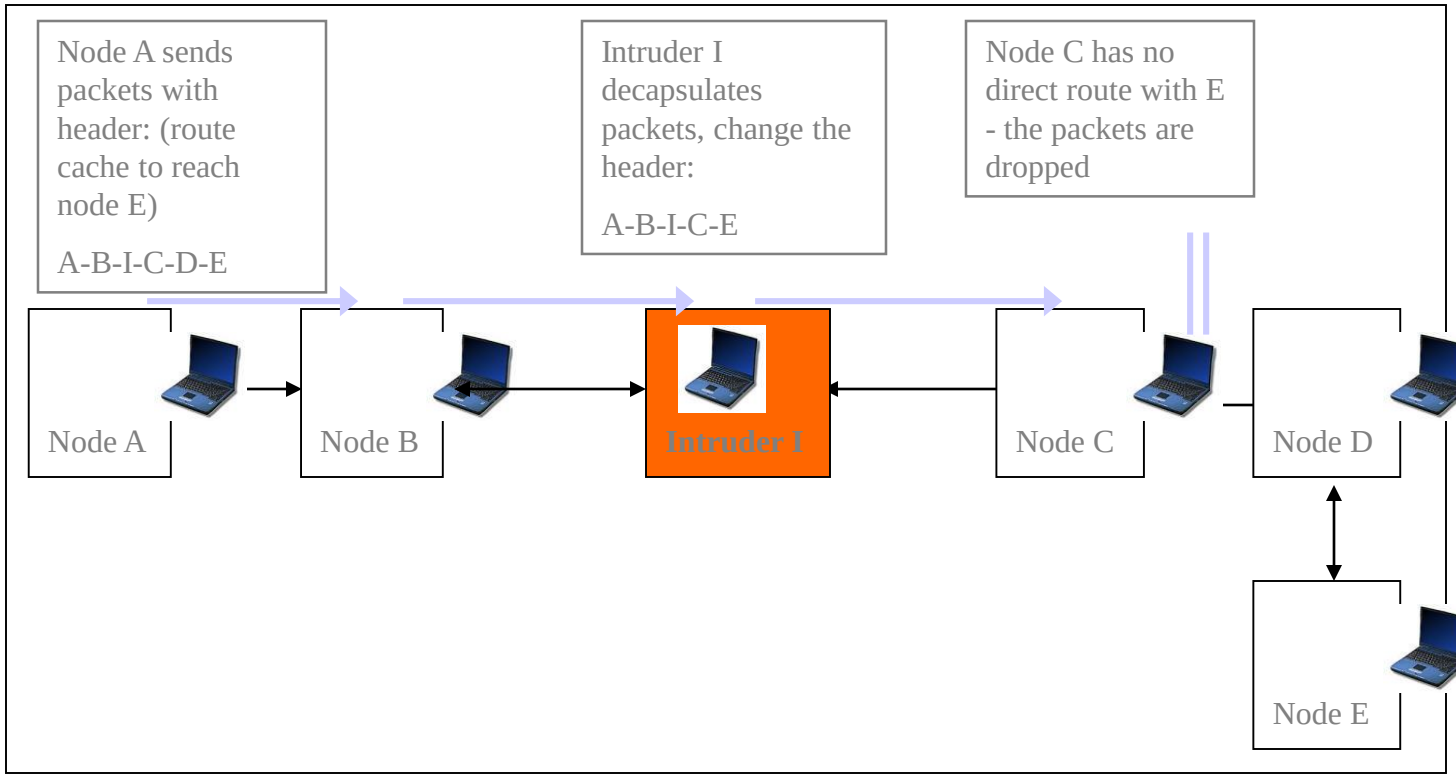
Security

Possible routing attacks: attacks using modification

- Malicious node announces better routes than the other nodes in order to be inserted in the network
  - Redirection by changing the route sequence number
  - Redirection with modified hop count
  - Denial Of Service (DOS) attacks with modified source routes
    - A malicious node is inserted in the network through one of the previous techniques
    - The malicious node changes the packet headers it receives
    - The packets will not reach the destination
    - The transmission is aborted

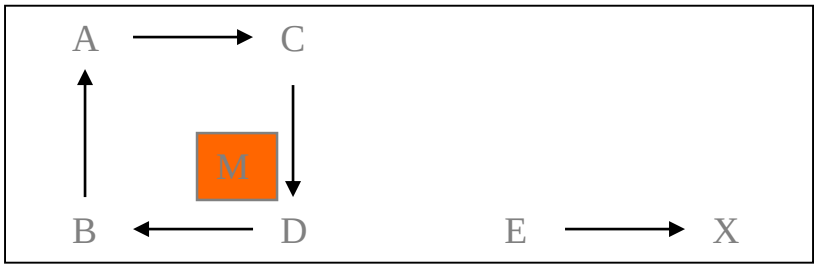
Possible routing attacks: attacks using modification

- DOS attacks with modified source routes



Possible routing attacks: attacks using impersonation

- Usurpation of the identity of another node to perform changes
  - Spoofing MAC address of other nodes
  - Forming loops by spoofing MAC address
    - A malicious node M can listen to all nodes
    - It changes its MAC address to the MAC address of another node
    - It announces to several nodes a shorter path to reach X



    - X is now unreachable because of the loop formed

Possible routing attacks: attacks using fabrication

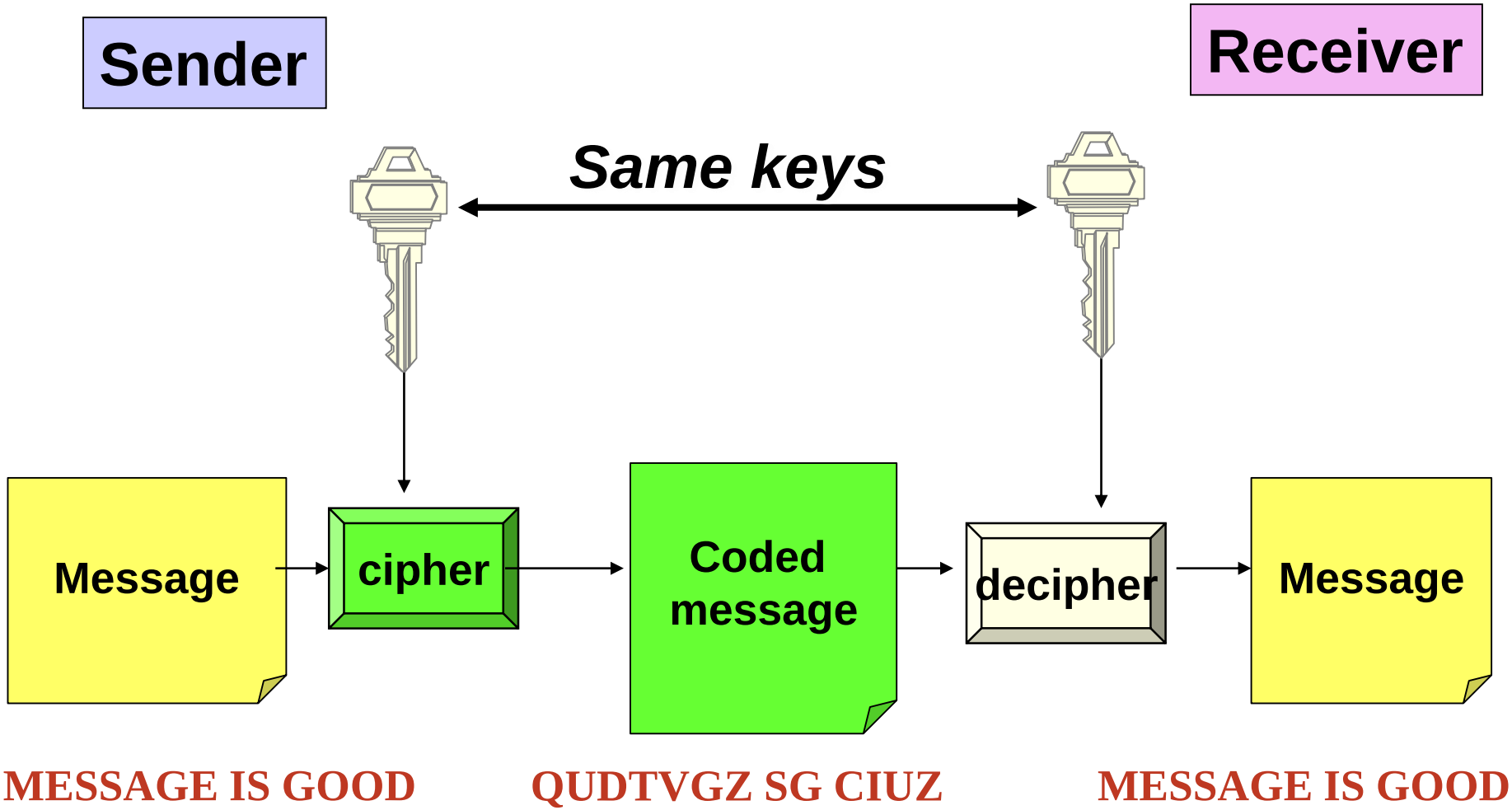
- **Generates traffic to disturb the good operation of an ad-hoc network**
  - **Falsifying route error messages**
    - Isolate nodes
  - **Corrupting routing state**
    - Hacker can easily broadcast a message with a spoofed IP address such that the other nodes add this new route to reach a special node S
    - The malicious node will receive the packets intended to S
  - **Routing table overflow attack**
    - Hacker can send in the network a lot of route to non-existent nodes until overwhelm the protocol
  - **Replay attack**
    - Hacker sends old advertisements to a node
  - **Black hole attack**
    - Hacker advertises a zero metric route for all destinations
    - All the nodes around it will route packets towards it

Key Management - basics

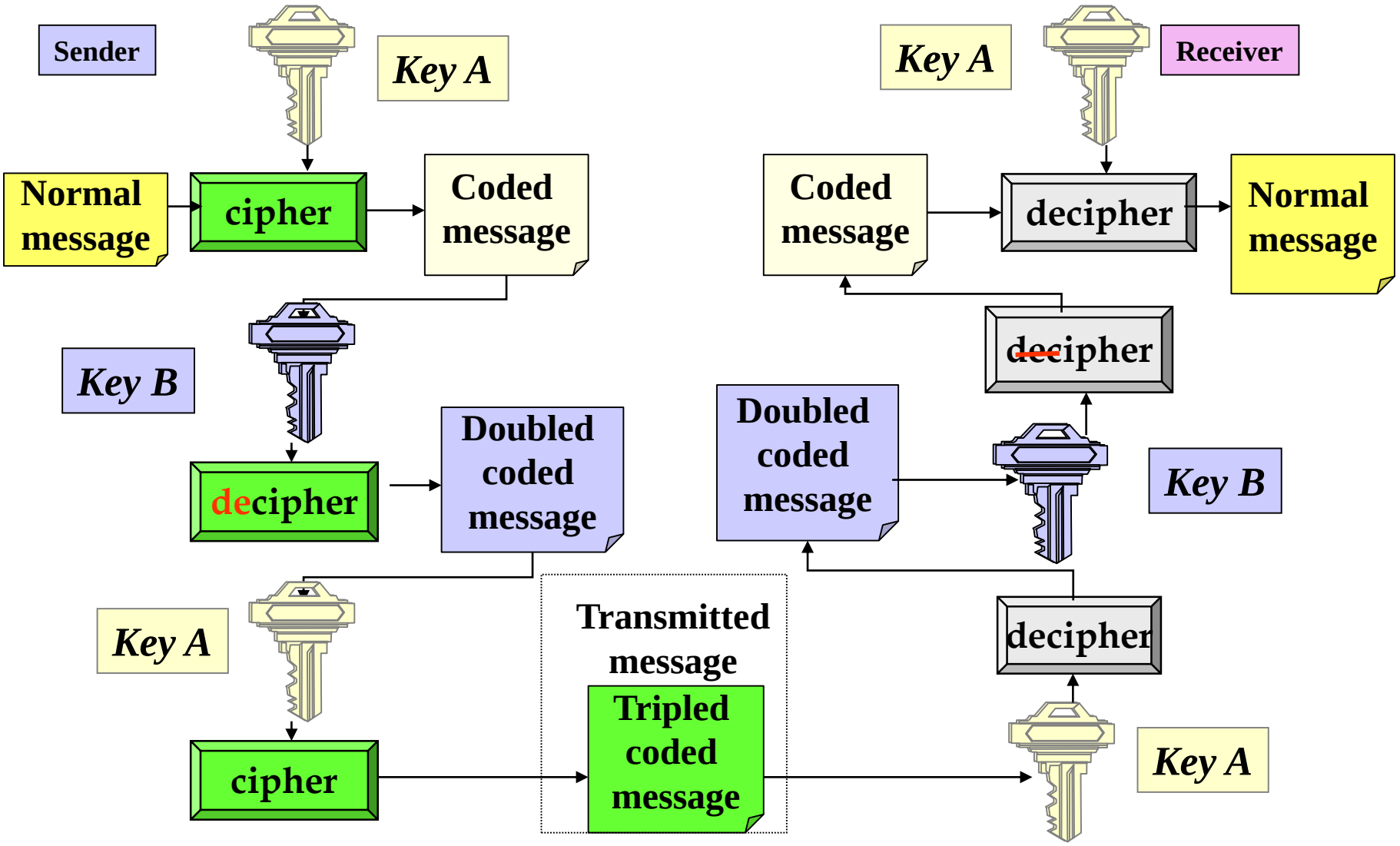
Symmetric cipher

- Advantages
  - Fast and relatively secure
    - Provides integrity and privacy
  - Larger key length ⇳ larger security
- Disadvantages
  - Requires the share of a secret key
    - How?
  - Complex administration and non-scalable
    - It is needed to distributed keys
    - A key for each receiver

Symmetric cipher



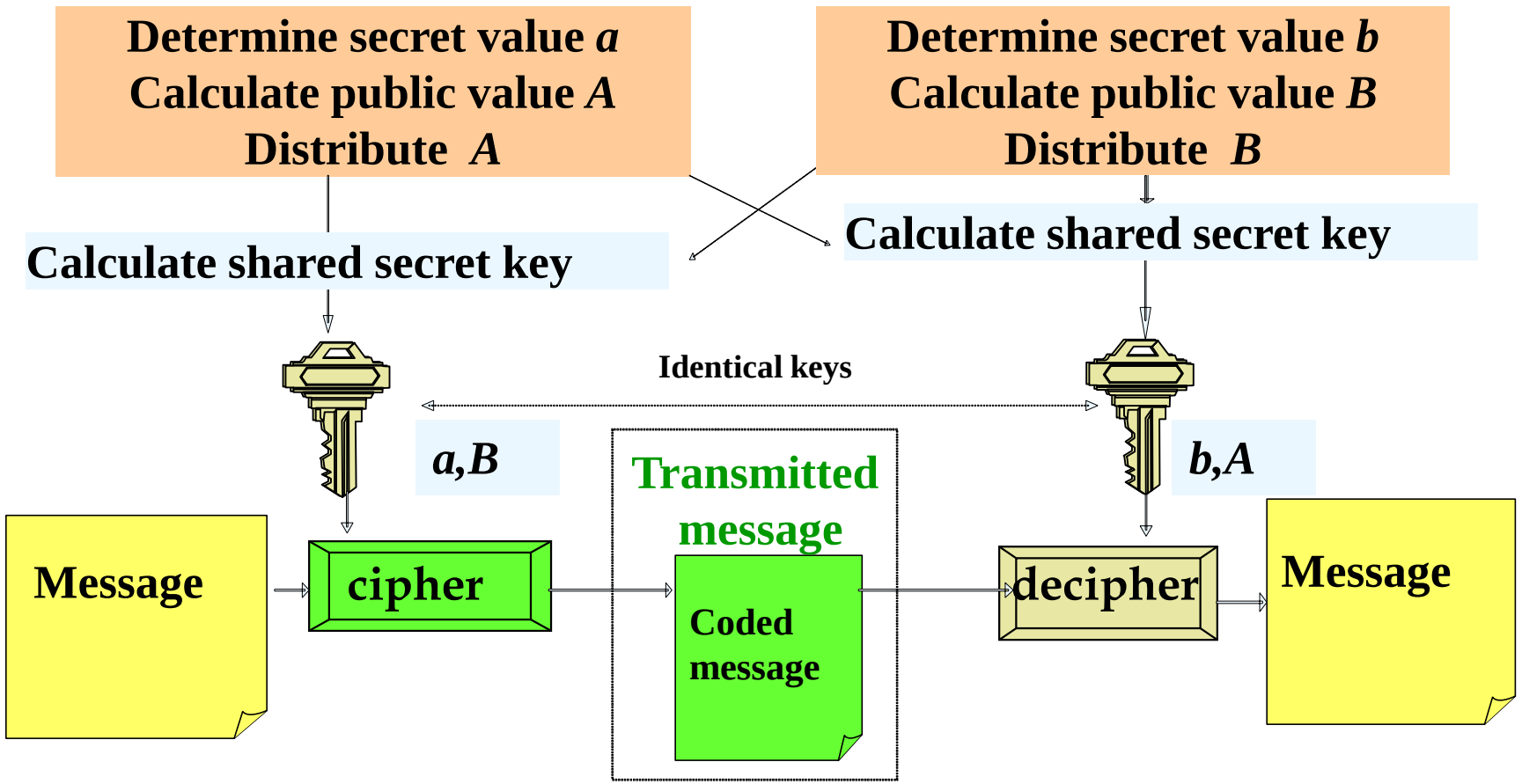
Triple symmetric mechanisms (e.g. “3-DES”)



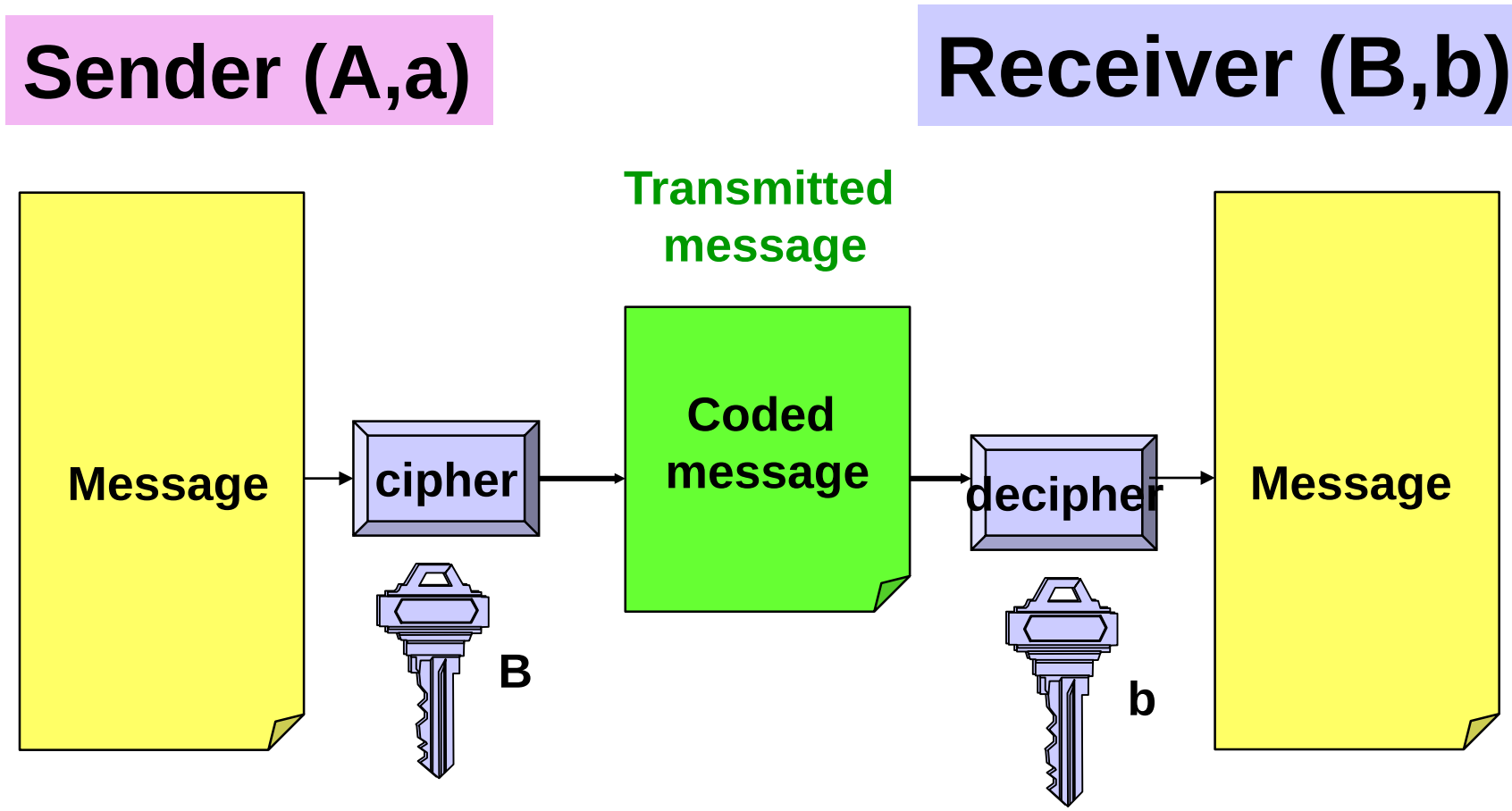
Asymmetric cipher

- Also known as PKE - public key encryption
- Advantages
  - It is not needed to share secret keys *à priori*
  - It is scalable and versatile
- Disadvantages
  - Generally computationally intensive
  - It may require a certificate authority
  - Private keys have to be confidential

Diffie-Hellman

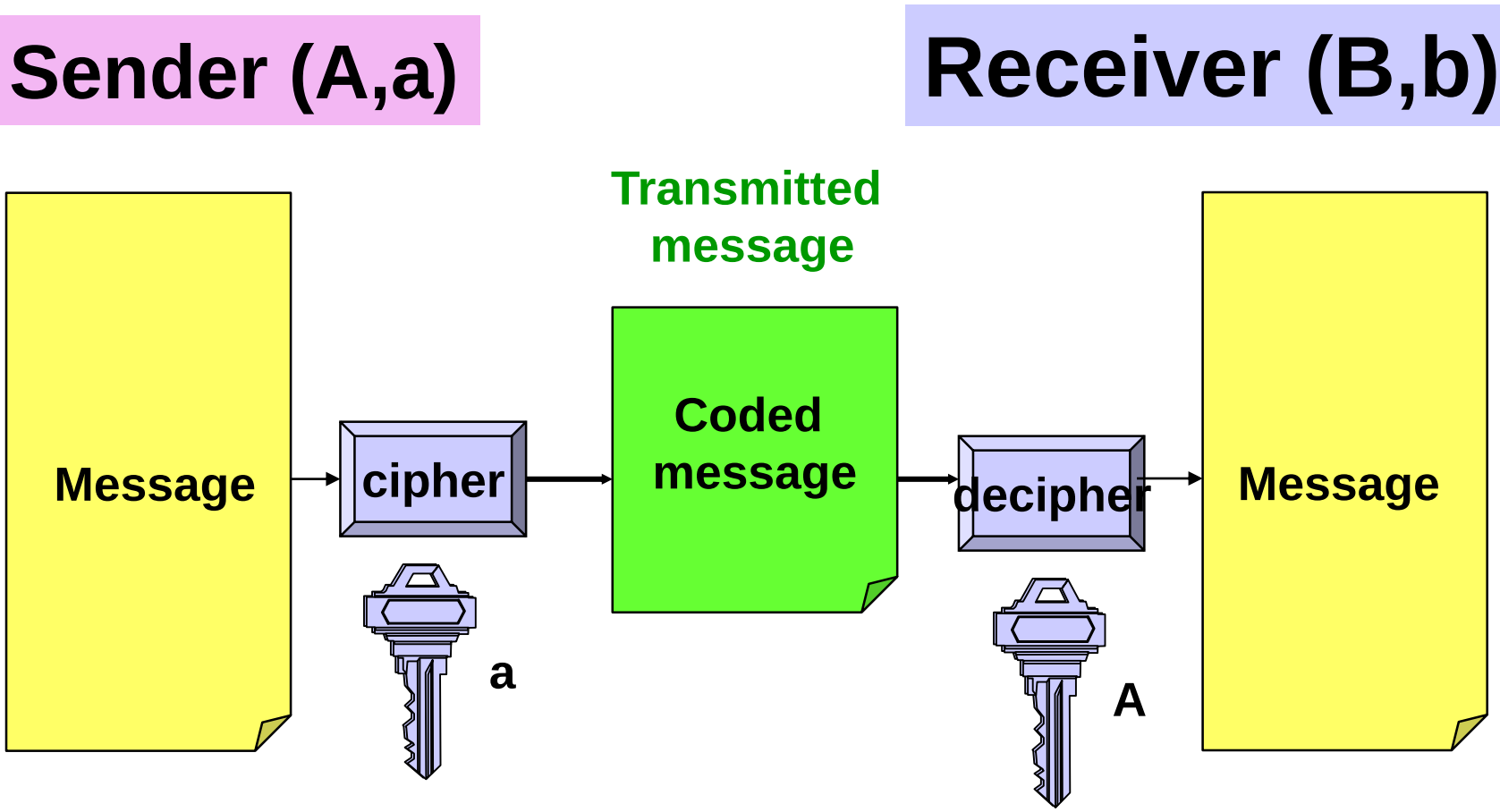


Public-private pair for confidentiality



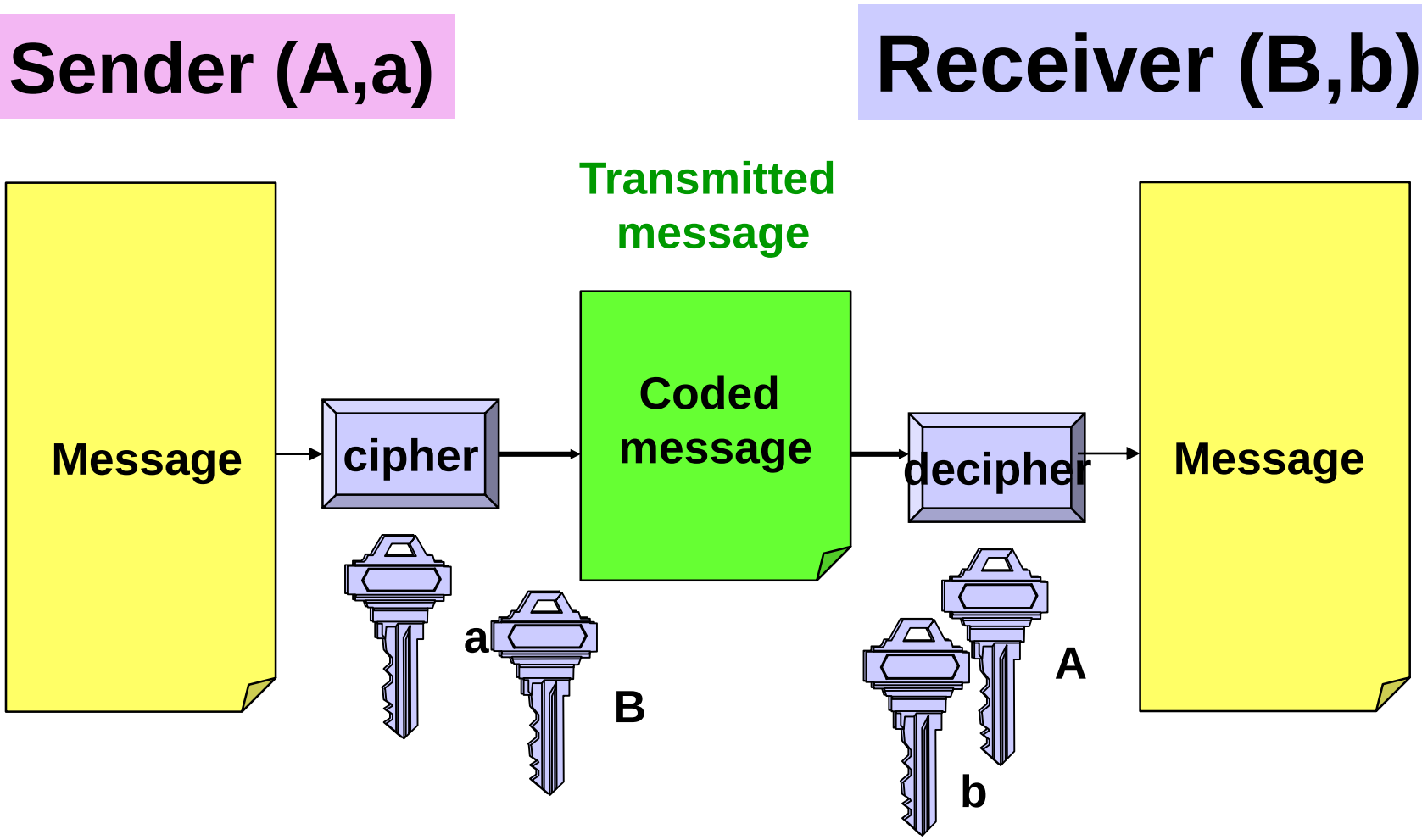
Guarantees confidentiality without sender guarantee

Public-private pair for authentication



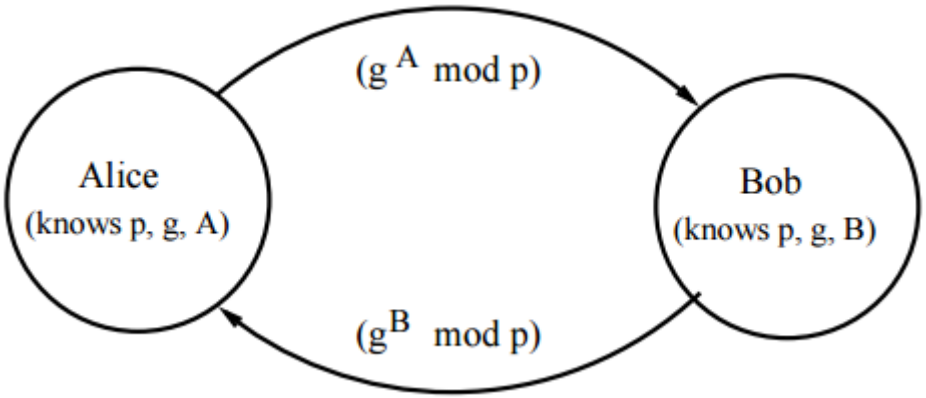
Authenticates the sender, since only the sender will have the secret key a, corresponding to the public key A

Public-private pair for confidentiality and authentication



Guarantees confidentiality and authenticates the sender, since only the sender will have the secret key a, corresponding to the public key A

Diffie-Hellman



- Alice and Bob agree on a prime number **p** and a base **g**.
- Alice chooses the secret number **a**, and sends to Bob ($\mathbf{g}^{\mathbf{a}} \bmod \mathbf{p}$).
- Bob chooses the secret number **b**, and sends to Alice ($\mathbf{g}^{\mathbf{b}} \bmod \mathbf{p}$).
- Alice calculates $((\mathbf{g}^{\mathbf{b}} \bmod \mathbf{p})^{\mathbf{a}} \bmod \mathbf{p})$.
- Bob calculates $((\mathbf{g}^{\mathbf{a}} \bmod \mathbf{p})^{\mathbf{b}} \bmod \mathbf{p})$.

- Alice and Bob use this value as their session key. **p** and **g** do not have to be protected.

Example Diffie-Hellman

- Alice and Bob choose p = 23 e g = 5.
- Alice chooses a = 6 and sends 5^6 mod 23 = 8.
- Bob chooses b = 15 and sends 5^{15} mod 23 = 19.
- Alice calculates 19^6 mod 23 = 2.
- Bob calculates 8^{15} mod 23 = 2.

- 2 is the shared key.

RSA (Rivest-Shamir-Adleman) Key

- Each user generates the pair public/private key
- Randomly generates 2 large prime numbers - **p**, **q**
- Calculates the modulus of the system **N**=**p**.**q**
  - ø(N)=(p-1)(q-1)
- Selects a random key **e**
    - where 1<**e**<ø(**N**), gcd(**e**,ø(**N**))=1 gcd = greatest common divisor
- Solves the equation to find the key and deciphers **d**
  - **e**.**d**=1 mod ø(**N**) and 0≤**d**≤**N**
- Publishes the public key: KU={**e**,**N**}
- Maintains the private key to decipher: KR={**d**,**p**,**q**}

Example RSA

1. Selects prime numbers: p=17 & q=11
2. Calculates $n = pq$ =17×11=187
3. Calculates ø(n)=(p–1)(q-1)=16×10=160
4. Selects e *:* gcd(e,160)=1; chooses e=7
5. Determines d*: de*=1 mod 160 and d < 160: value is d=23 because 23×7=161= 160+1
6. Publishes public key KU={7,187}
7. Maintains secret key KR={23,17,11}

Use of RSA

- To cipher the message M, the sender:
  - Obtains a public key of the receiver $KU=\{e,N\}$
  - Calculates: $C=M^e \bmod N$, where $0 \leq M < N$
- To decipher C :
  - Uses its private key $KR=\{d,p,q\}$
  - Calculates: $M=C^d \bmod N$
- Message M has to be lower that the modulus N

- Message M = 88 (88<N=187)
- Cipher: $C = 88^7 \bmod 187 = 11$; e=7
- Decipher: $M = 11^{23} \bmod 187 = 88$; d=23

RSA Real

p

12131072439211271897323671531612440428472427633701410925634549312301964
37304208561932419736532241686654101705736136521417171171379797429933487
1062829803541

q

12027524255478748885956220793734512128733387803682075433653899983955179
85098879789986914690080913161115334681705083209602216014636634639181247
0987105415233

n

14590676800758332323018693934907063529240187237535716439958187101987343
87990053589383695714026701498021218180862924674228281570229220767469065
43401224889672472407926969987100581290103199317858753663710862357656510
50788371429711563734278891146353510271203276516651841172685983798867211
18372050855263466187400 53

$\phi(n)$

14590676800758332323018693934907063529240187237535716439958187101987343
87990053589383695714026701498021218180862924674228281570229220767469065
43401224889648313811232279966317301397777852365301547848273478871297222
05858745715289160645926971811926897116355507080264399952954964411681194
75165139381842966835212 80

e - the public key

$$65537$$

d - the private key

894894250092744443682285459217730939196695860658842574454978544564876748396298183909349419732628796167979706089172836798754993315741611138540888132754881105882471930775825272784379065040156806234235500672400424666656542323835029222154936232894721388664458187891279461234078077257026266440910365025372545139713

Encryption/Decryption

Encryption: $1976620216402300889624482718775150^{e} \bmod n$

350521113386730266902124239370533285118807608115799816206428023466858106231098502359430490809733862411137840407947041939782153784997654130836464387847409523069325349451950801838615742252262188798272324539128205968864403775360824656817500744174591514854074458625110234722355608230534977915189288202722577877 86

Decryption:

350521113386730266902124239370533285118807608115799816206428023466858106231098502359430490809733862411137840407947041939782153784997654130836464387847409523069325349451950801838615742252262188798272324539128205968864403775360824656817500744174591514854074458625110234722355608230534977915189288202722577877 86$^{d} \bmod n$

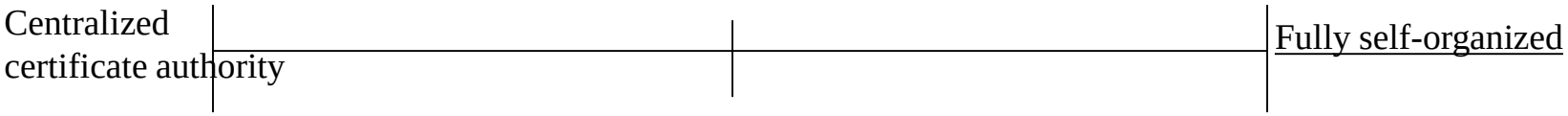
1976620216402300889624482718775150 (which is our plaintext)

Key Management in ad-hoc networks

- Challenges
  - Vulnerable mobile nodes
    - More exposed to physical attacks
  - Mobility-induced unstable network topology
    - Rapid change in connectivity
    - Potential network partition
  - No infrastructure to send key information to nodes

Self-organized public key management

- The scope of key management



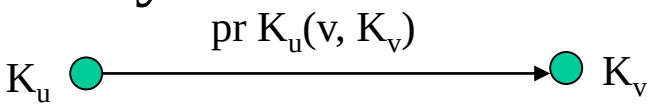
- Public key management system
  - How to get the authentic public key of a user?
- Users issue certificates based on personal acquaintance
  - Certificate: binding between node and its public key: public key, identity and signature of the issuer
  - Certificates stored and distributed by the users themselves
- Updated certificate repository
- Non-updated certificate repository
  - Expired certificates

Certificate Creation

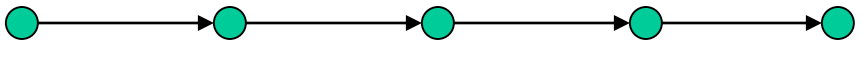
- u gets the public key K_v of v through a side channel
- u creates certificate for v to bind the K_v and v
- u stores the certificate of v in its local repository G_u and sends the certificate to v
- Global certificate graph $\quad pr\ K_u(v, K_v)$
- Connectivity of certificate graph
  - Small world phenomenon
    - Due to social relationships
    - Mobility increases the connectivity
      - Mobile nodes/users exchange their public keys whenever they meet

Certificate graph

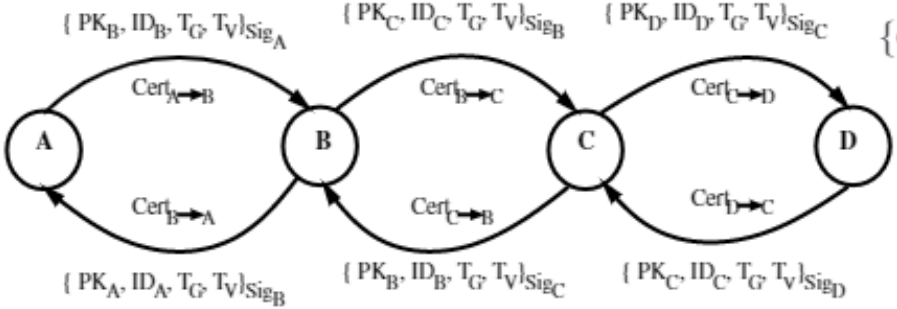
- Vertices: public keys of some nodes
- Edges: public keys certificates issued by users



- User u wants to obtain public key of user v
  - Find a chain of valid public keys certificates leading to v
    - First hop uses edge from u (certificate issued by u)
    - Last hop is a certificate issued by v
    - Intermediate nodes trusted through the previous certificate in the path



$\{Cert_{A\to B}, Cert_{B\to C}, Cert_{C\to D}\}$



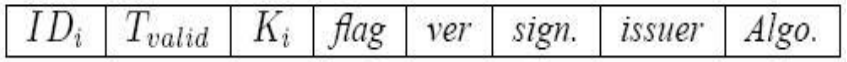
$\{Cert_{D\to C}, Cert_{C\to B}, Cert_{B\to A}\}$

Revocation

- Certificate revocation
  - Explicitly
    - Each node has a list of nodes that request updates for the certificates they issue
  - Implicitly
    - Expiration time of the certificate
    - Simple but need time synchronization of the nodes and decide the expiration time properly
- Key revocation
  - Inform the issuers of its certificate, then the certificate is invalid

Self-securing ad-hoc wireless networks

- Goal
  - Achieve high security assurance
  - High success ratio
  - Efficient communication
- Basic operation
  - Distributed PKI
    - System private key is split to server nodes
    - Quorum of *k* servers produce certificate updating
    - Structure of certificate



  - Operates in phases
    - Server group formation/maintenance
    - Certificate updating/revocation
    - Shared key updating/renewing

Shared secrets

- Threshold secret sharing
  - Each node has a part of the secret
    - Unique ID, derived from the node's address
    - Mechanism for local detection of misbehaving nodes
    - At least K one-hop neighbouring nodes
    - Key pair for each node (public and secret keys)
- Encryption mechanism uses RSA asymmetric keys
- Global Secret Key (SK) and the corresponding Public Key (PK)
  - SK functionality is 'distributed' among nodes
  - Any K nodes holding a partial secret form a distributed Certificate Authority (CA)

Shared secrets

- Partial secret key function of nodes IDs
  - Generation of a polynomial of order K-1, known only in the initial setup
  - K nodes holding a partial secret share recover SK using Lagrange interpolation
  - Coalition of K-1 nodes holding a partial secret share does not have any information about SK
- Node wanting to use the distributed CA
  - Contact K nodes that have a partial secret share
  - K one-hop neighbouring nodes
    - It is easier to collect reliable information about misbehaviour of closer nodes
  - PK is known by all nodes