QoS and Security

Mestrado em Engenharia de Computadores e Telemática

2022/2023

Quality of Service

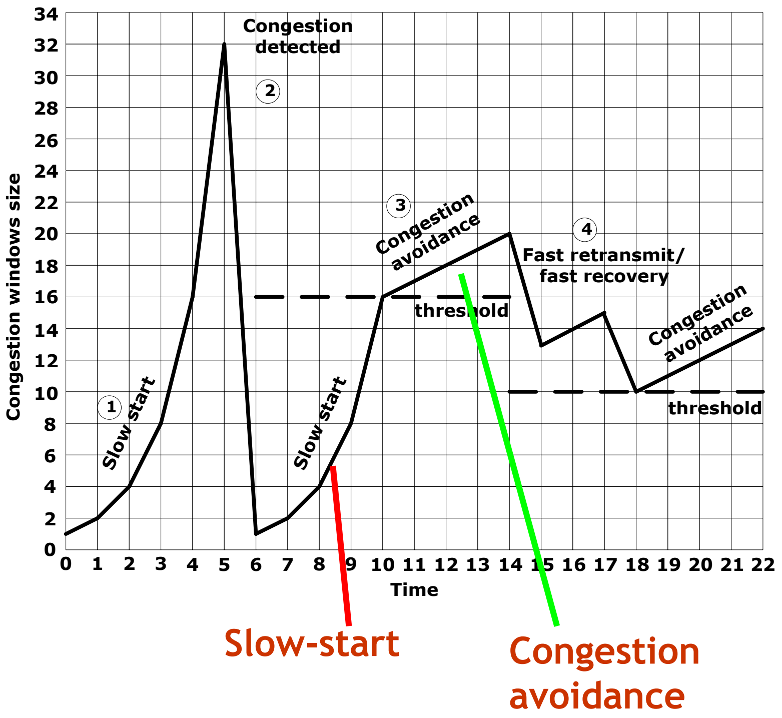
TCP- and UDP-based applications

Problem: Evaluate TCP

- Why does TCP perform *poorly* in ad-hoc/vehicular networks?
  - Developed for wire-line networks
  - Assumes 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 wireless networks, and also 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 wireless and 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 wireless and 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

New possibilities?

TCP-Cubic (1)

- CUBIC is RTT independent
- The window size is a cubic function of the time t, which is passed time since the last congestion occurrence

$$W_{CUBIC} = C\left(t - \sqrt[3]{\frac{\beta . W_{max}}{C}}\right)^3 + W_{max}$$

- where *W_CUBIC* is the *cwnd* size for cubic congestion control mechanism,
- W_max, a *cwnd* size just before last window reduction,
- C, a predefined constant (scaling factor),
- β, a *cwnd* size decrease factor. Window size reduction at the time of loss event is

$$W(t) \leftarrow W(t^*)(1-\beta).$$

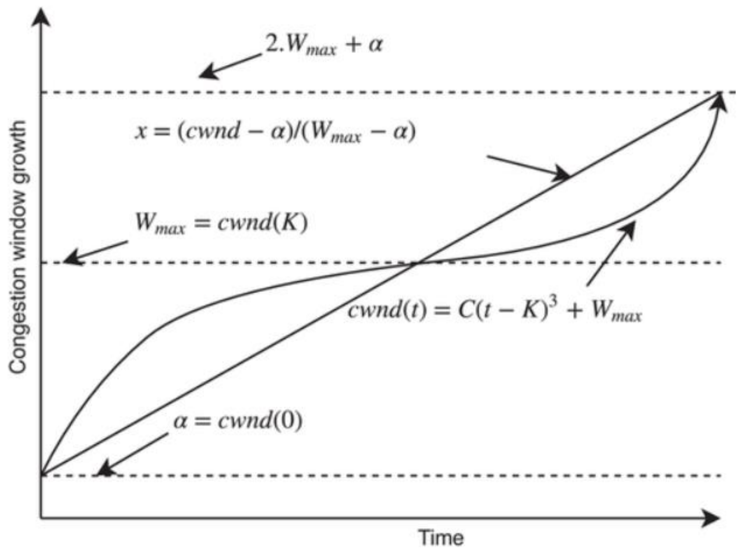
- where W(t*) is the cwnd size at the time t* of packet loss, that is, Wmax.
- When a packet loss is detected, W(t) is reduced as per the previous equation.

https://www.cs.princeton.edu/courses/archive/fall16/cos561/papers/Cubic08.pdf

TCP-Cubic (2)

- Whenever a packet dropping event occurs, the cwnd is reduced by a factor β, otherwise increased by the α for a successful ACK. For TCP with CUBIC, β=0.3, whereas for QUIC with CUBIC, β=0.3/2=0.15

$$K = \sqrt[3]{\frac{\beta . W_{max}}{C}}$$



QUIC

- QUIC was developed and deployed by Google in 2013, but was presented as a standard in 2021 by RFC9000.
- QUIC packets are carried in UDP datagrams to better facilitate deployment in existing systems and networks.
- QUIC handshake combines negotiation of cryptographic (TLS) and transport parameters.
  - It is structured to allow the exchange of application data as soon as possible.

– Application protocols exchange information over a QUIC connection via streams which are ordered sequences of bytes. Two types of streams can be created:
  - Bidirectional streams, which allow both endpoints to send data.
  - Unidirectional streams, which allow a single endpoint to send data.

– Avoids head-of-line blocking across multiple streams
  - When a packet loss occurs, only streams with data in that packet are blocked waiting for a retransmission to be received, while other streams can continue making progress.

– Two levels of data flow control in QUIC:
  - Stream flow control, which prevents a single stream from consuming the entire receive buffer for a connection by limiting the amount of data that can be sent on each stream.
  - Connection flow control, which prevents senders from exceeding a receiver's buffer capacity for the connection by limiting the total bytes of stream data sent on all streams.

https://datatracker.ietf.org/doc/html/rfc9000

QUIC: RTT estimation

- Use separate sequence numbers for data and packet delivery
- Decouple reliability completely from ordered delivery

```
+------------+--------------------------------+
|  Flags (8) |  Connection ID (64) (optional) | ->
+------------+--------------------------------+
+------------------------------------+--------------------------------+
|  Version (32) (client-only, optional) |  Diversification Nonce (256)  | ->
+------------------------------------+--------------------------------+
+------------------------------------+
|  Packet Number (8 - 48)  | ->
+------------------------------------+
+------------+------------+     +------------+
|   Frame 1  |  Frame 2   | ... |  Frame N   |
+------------+------------+     +------------+

        Stream frame
     +------------+--------------------+----------------+
     |  Type (8)  |  Stream ID (8 - 32)  |  Offset (0 - 64)  |
     +------------+--------------------+----------------+
     +------------------------+----------------------------+
     |  Data length (0 or 16) |  Stream Data (data length)  |
     +------------------------+----------------------------+
```

QUIC: RTT estimation

- Decouple reliability completely from ordered delivery
- Packet numbers are monotonically increasing (and hence unique)
  - Distinct from frame offsets within stream

- Use time between the transmission of a packet and its ACK (identifiable using unique packet number) for RTT estimate
  - Regardless of original transmission or retransmission

- Packet loss detected through lack of ACK packets

- However, the congestion control is independent of RTT
  - One of the most used algorithms is CUBIC

- Congestion control does not depend on the delays of the wireless medium
- Congestion control is applied upon detection of packet loss
  - Again, packet loss may not be a sign of congestion

Classification of Transport protocols

- How to make the transport protocol suited to wireless and ad-hoc networks?

- 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*

https://cseweb.ucsd.edu/classes/wi01/cse222/papers/brakmo-vegas-jsac95.pdf

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

https://ieeexplore.ieee.org/document/6596860

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

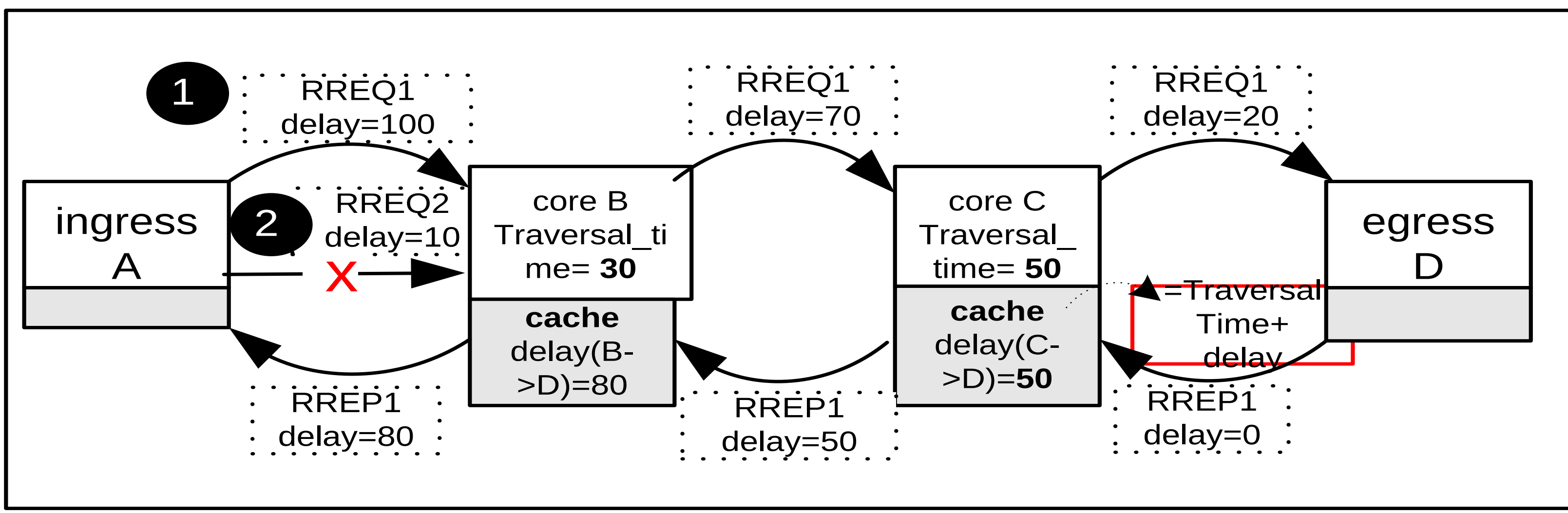
- Routing is an essential component for QoS
- It can inform a source node of the bandwidth and QoS availability of a destination node and of the path to the destination node
- Add QoS requirements in routing metrics
  - Difficult to route maintenance
  - Overhead of QoS routing
  - Reserved resources 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

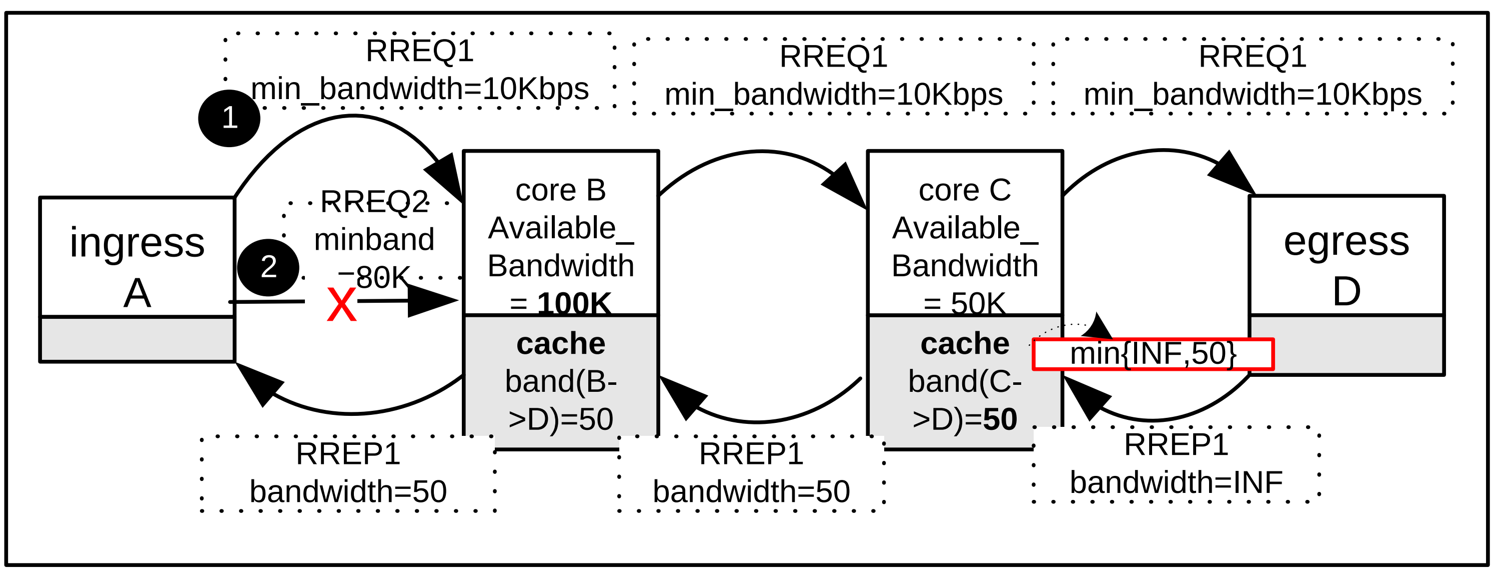
QoS for AODV- Delay

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



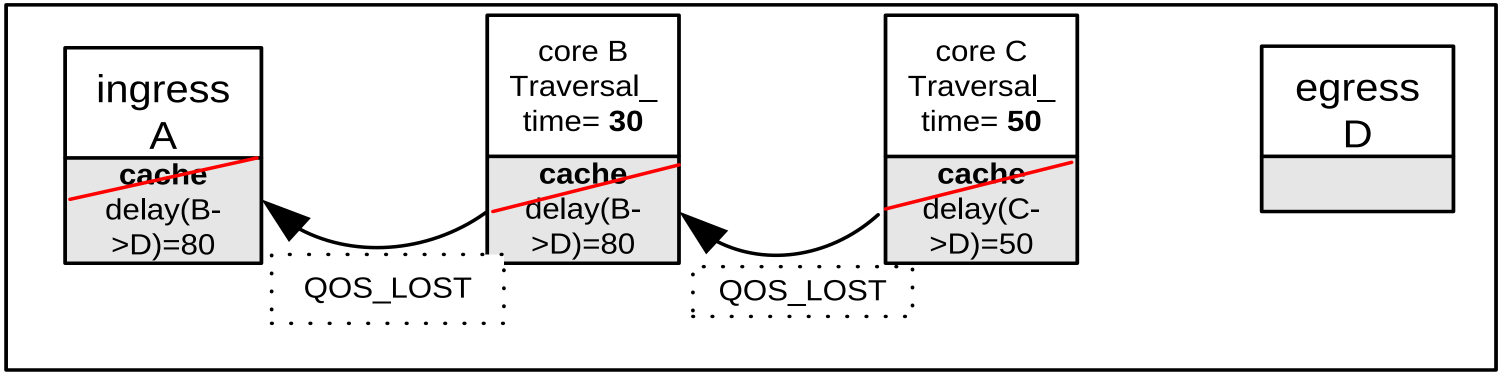
QoS for AODV- Bandwidth

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



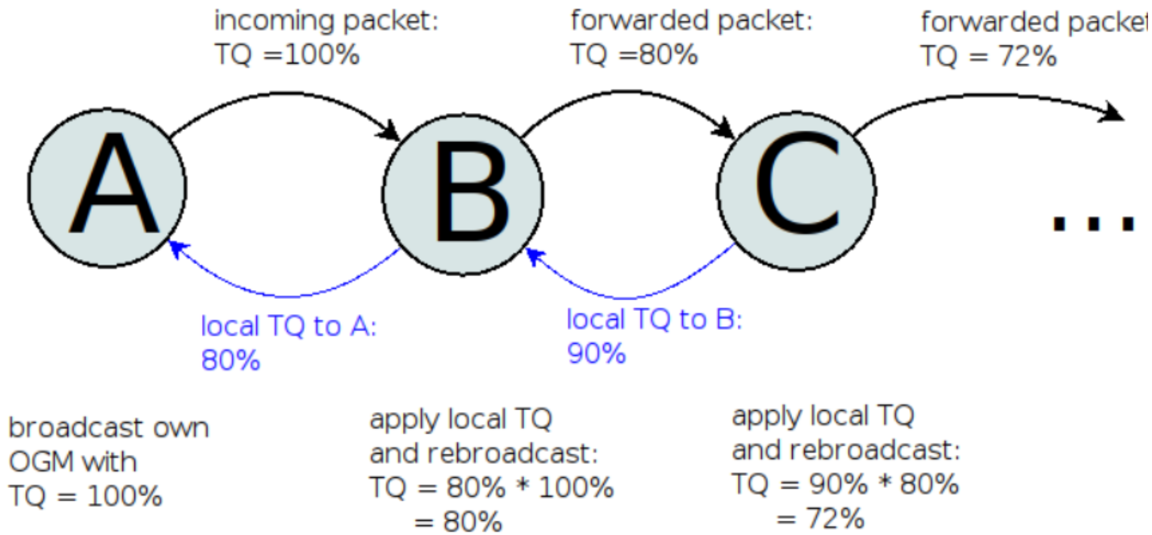
QoS for AODV- Loosing QoS

- 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?



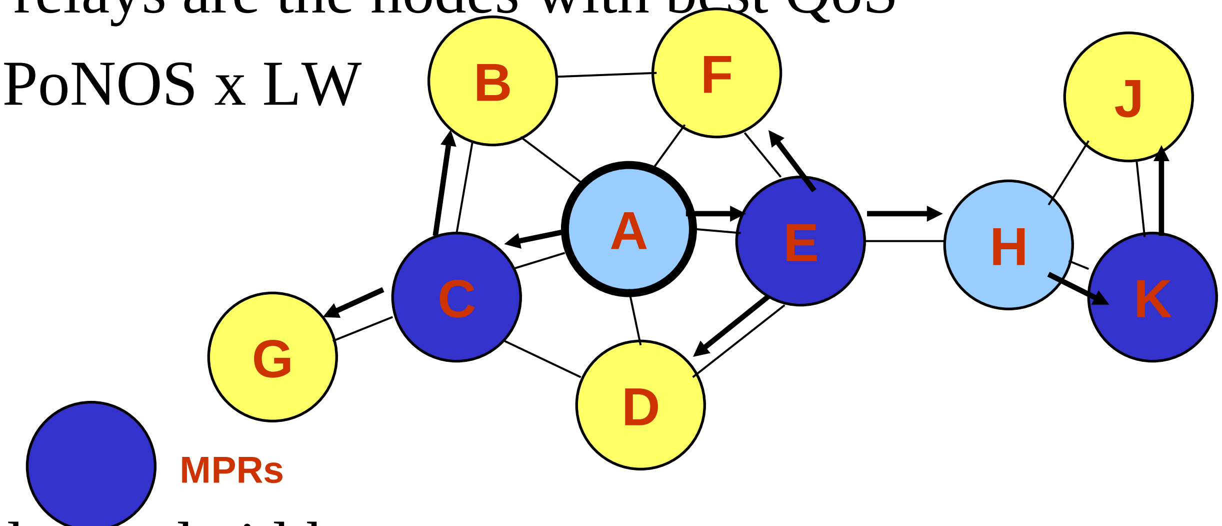
Transmission Quality (Batman v.3)

- 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.

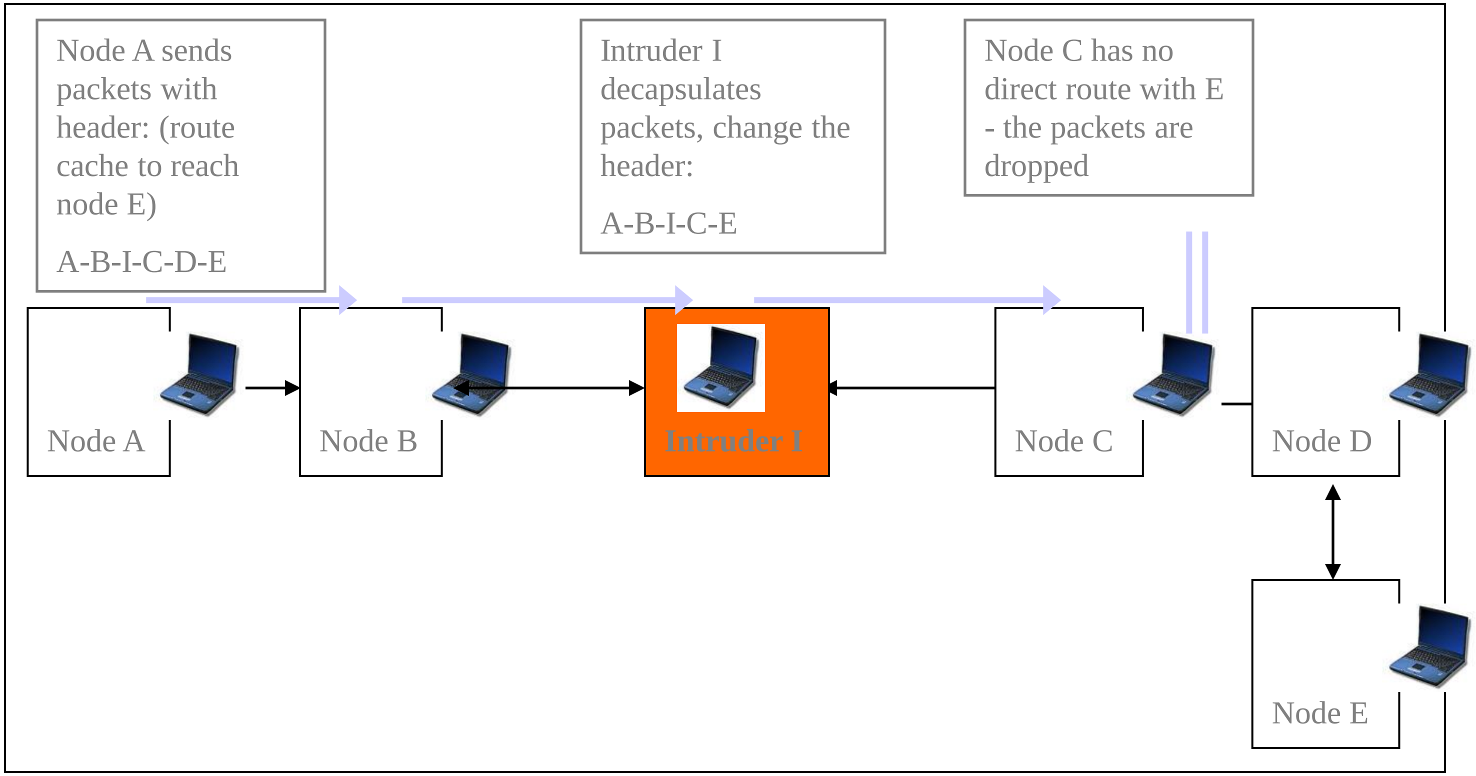
Security

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

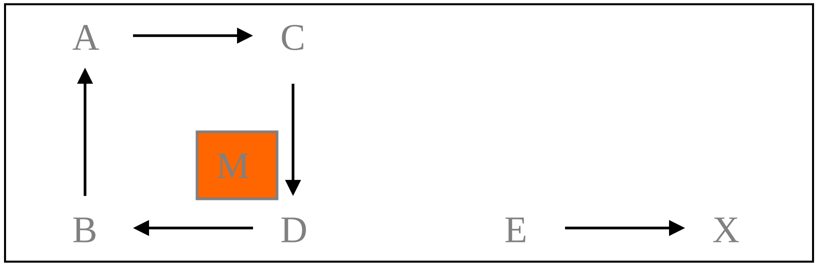
Possible routing attacks: attacks using modification

- 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

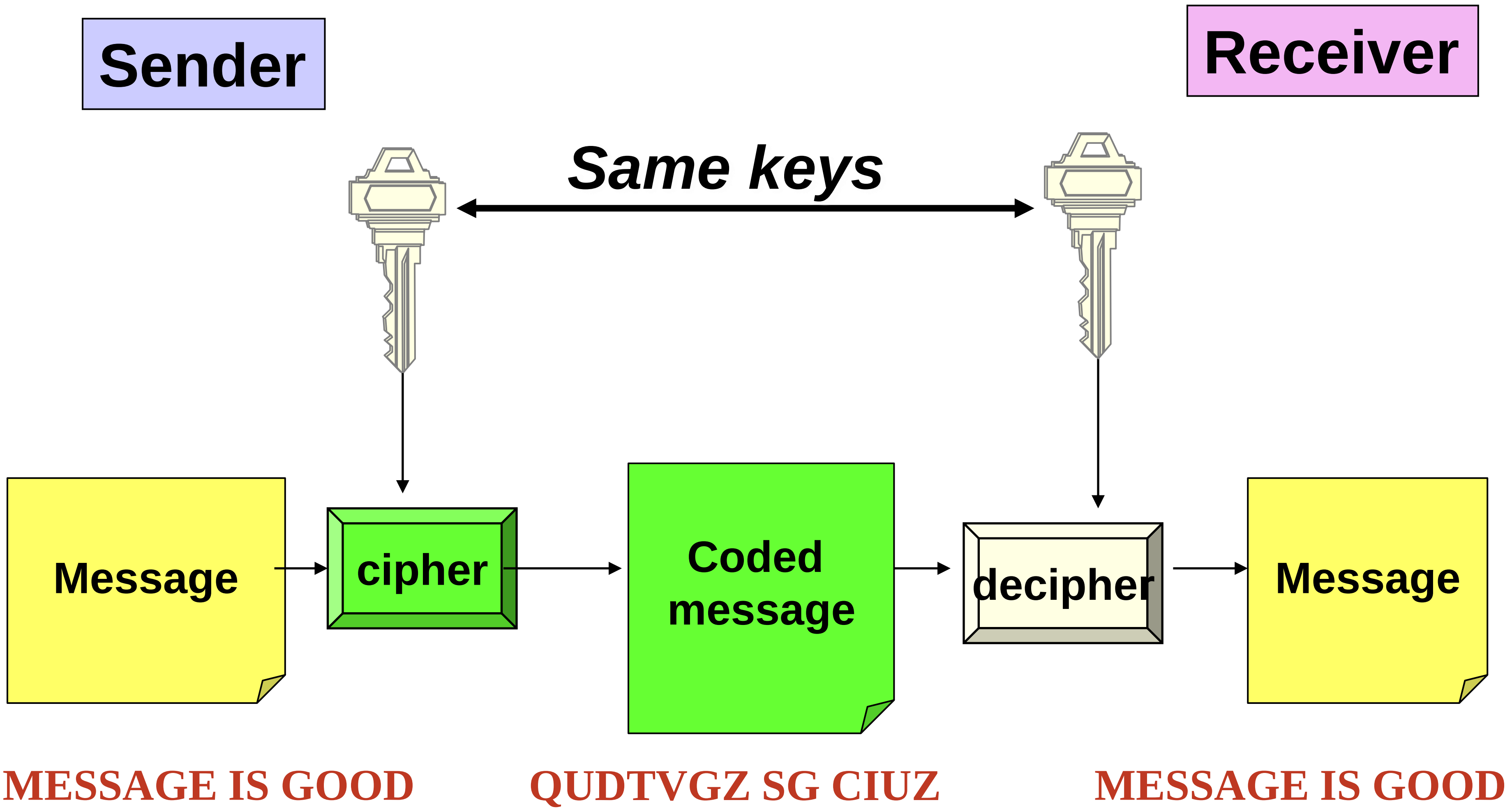
- Generates traffic to disturb the good operation of an ad-hoc network
  - Falsifying route error messages
    - Isolate nodes
  - Corrupting routing state
    - Hacker can easily broadcast a message with a spoofed IP address such that the other nodes add this new route to reach a special node S
    - The malicious node will receive the packets intended to S
  - Routing table overflow attack
    - Hacker can send in the network a lot of routes 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

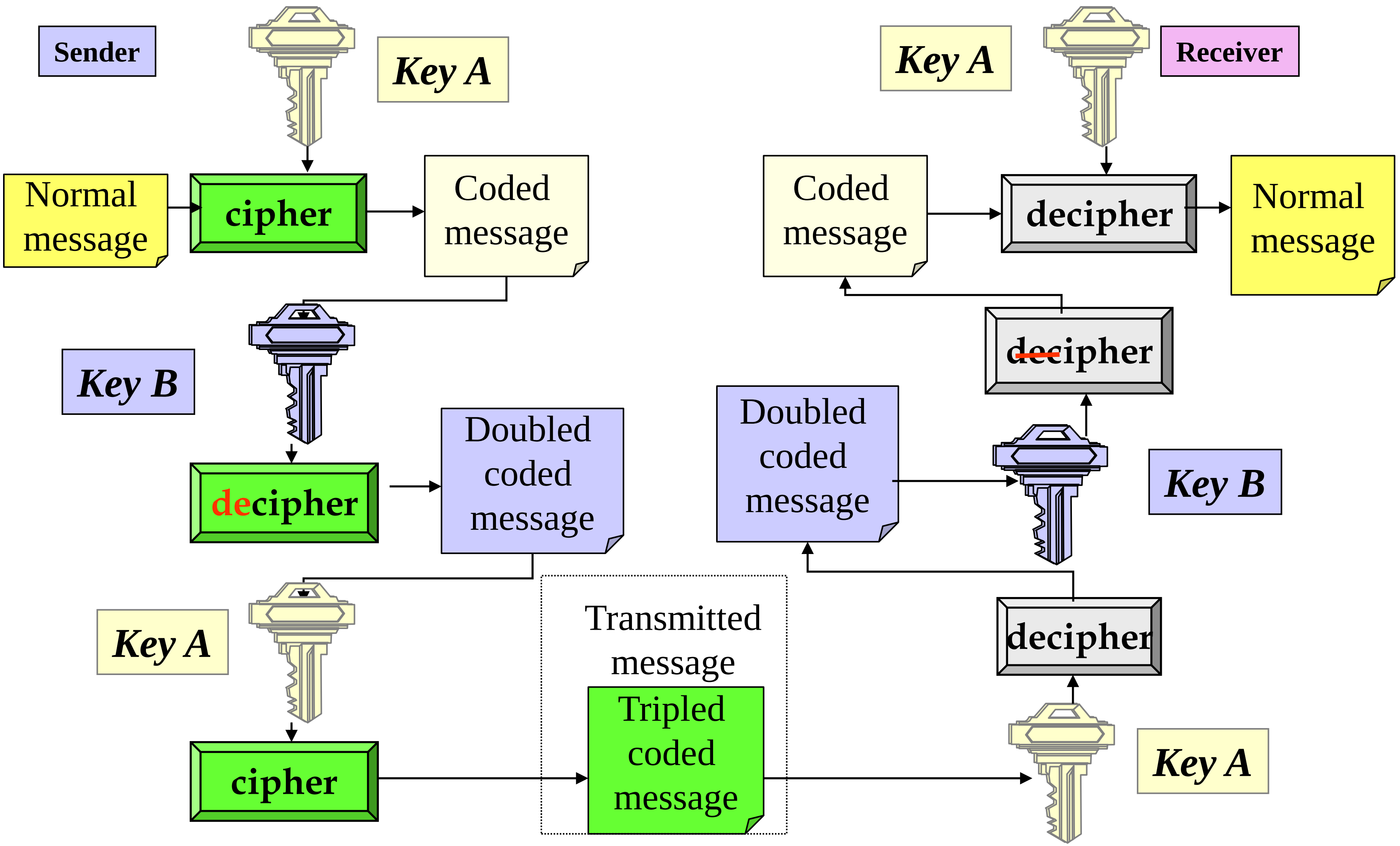
Symmetric cipher

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

Symmetric cipher



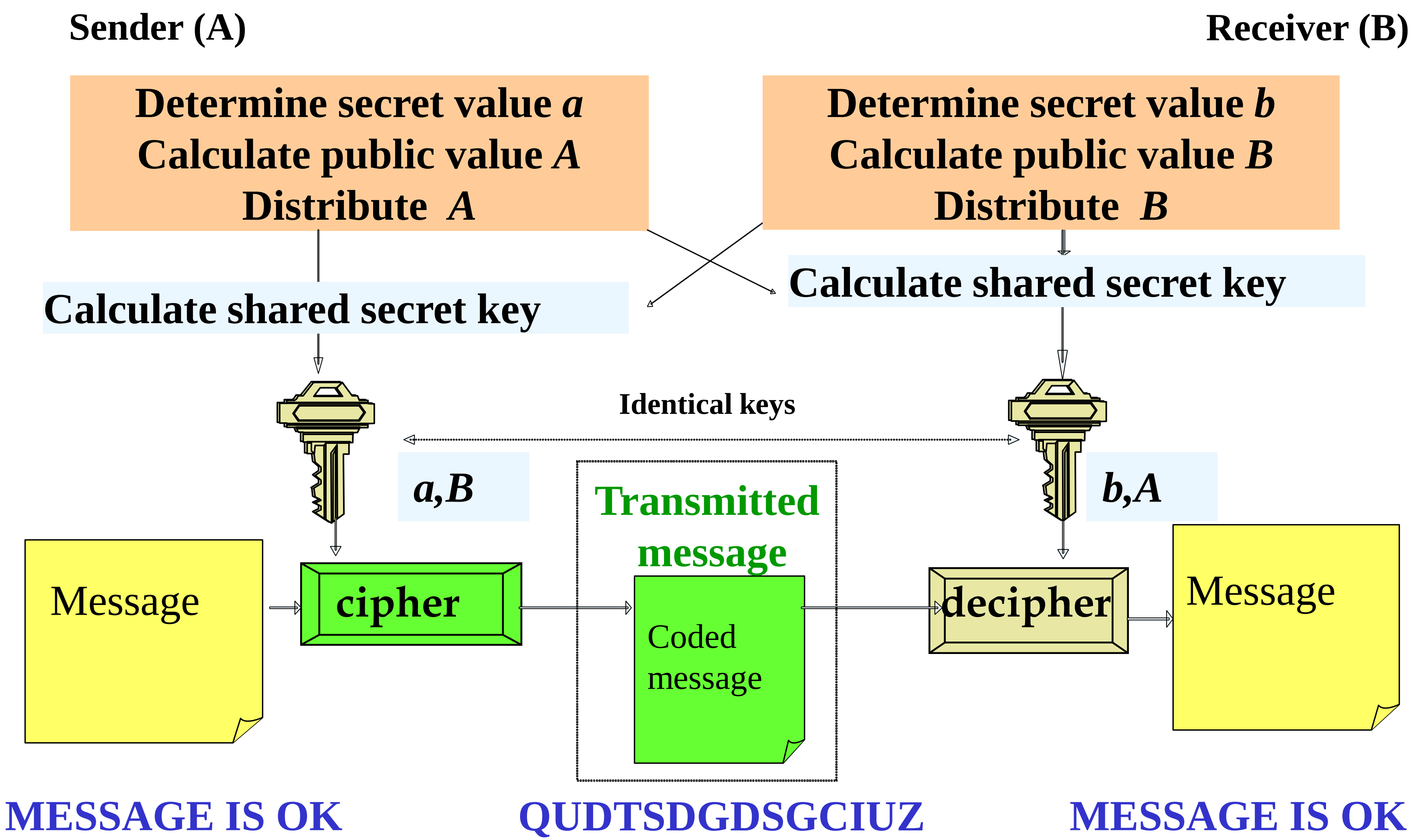
Triple symmetric mechanisms (e.g. “3-DES”)



Asymmetric cipher

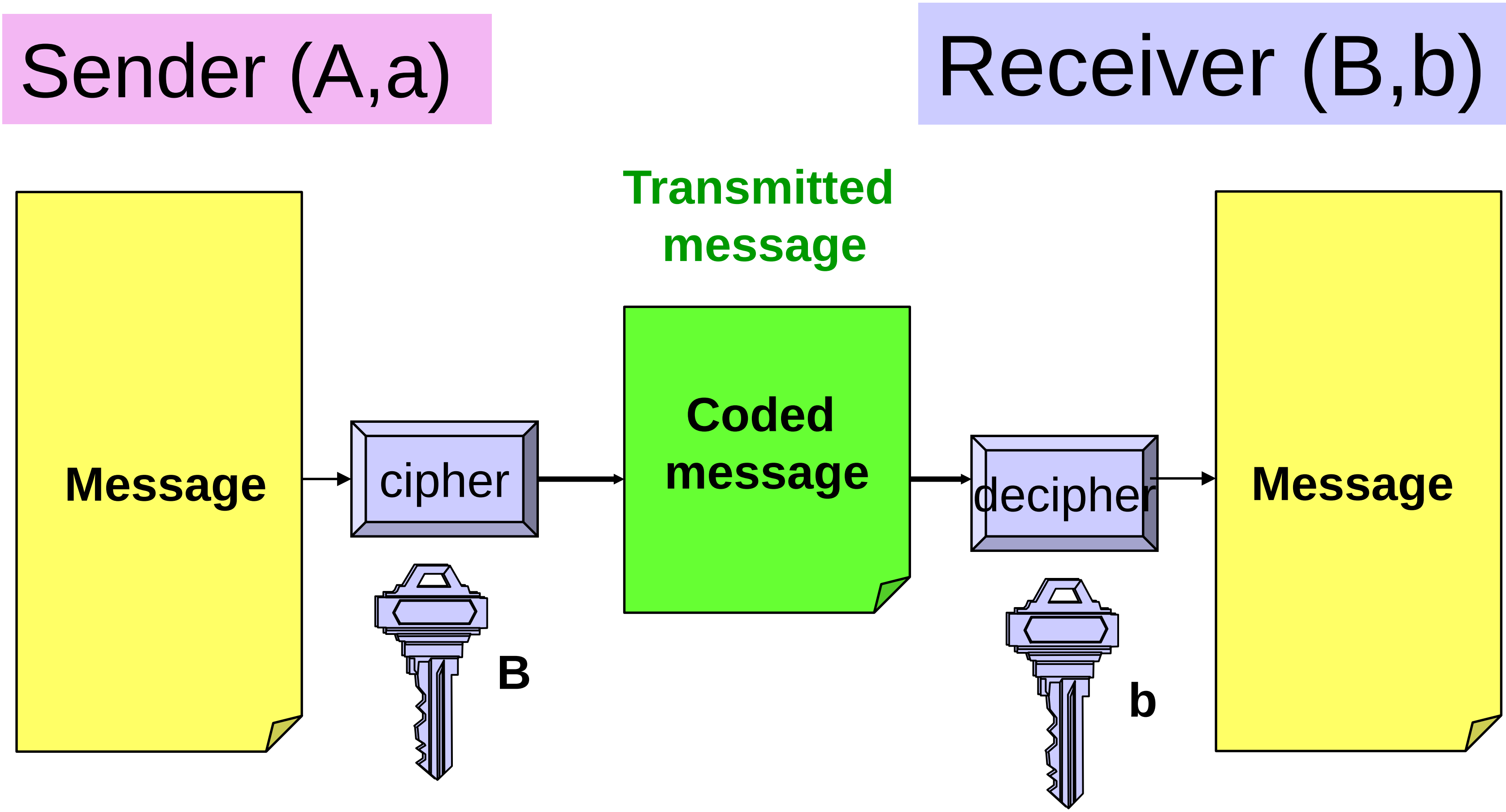
- Also known as PKE - public key encryption
- Advantages
  - It is not needed to share secret keys *à priori*
  - It is scalable and versatile
- Disadvantages
  - Generally computationally intensive
  - It may require a certificate authority
  - Private keys have to be confidential

Diffie-Hellman



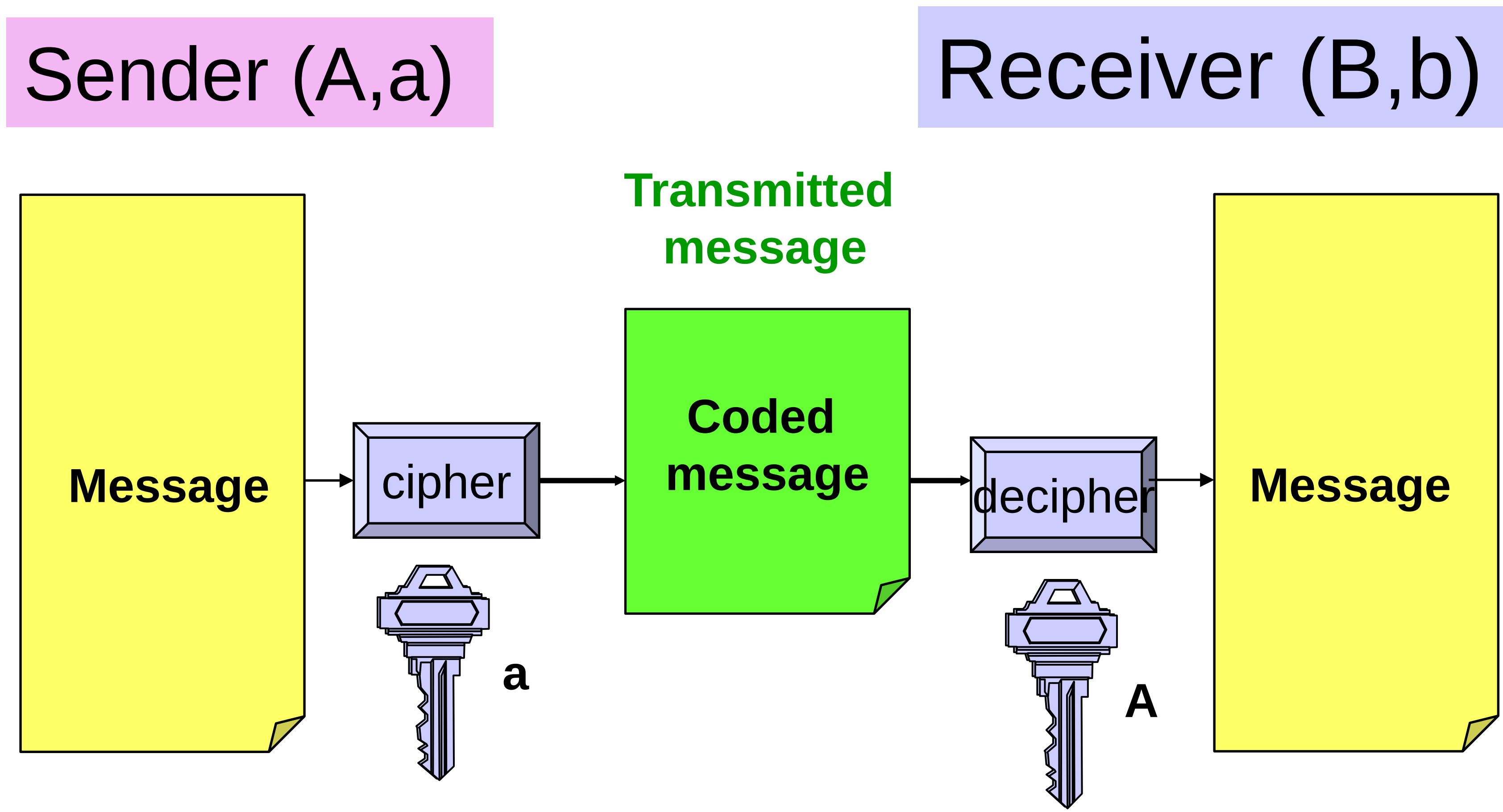
https://www.techtarget.com/searchsecurity/definition/Diffie-Hellman-key-exchange

Public-private pair for confidentiality



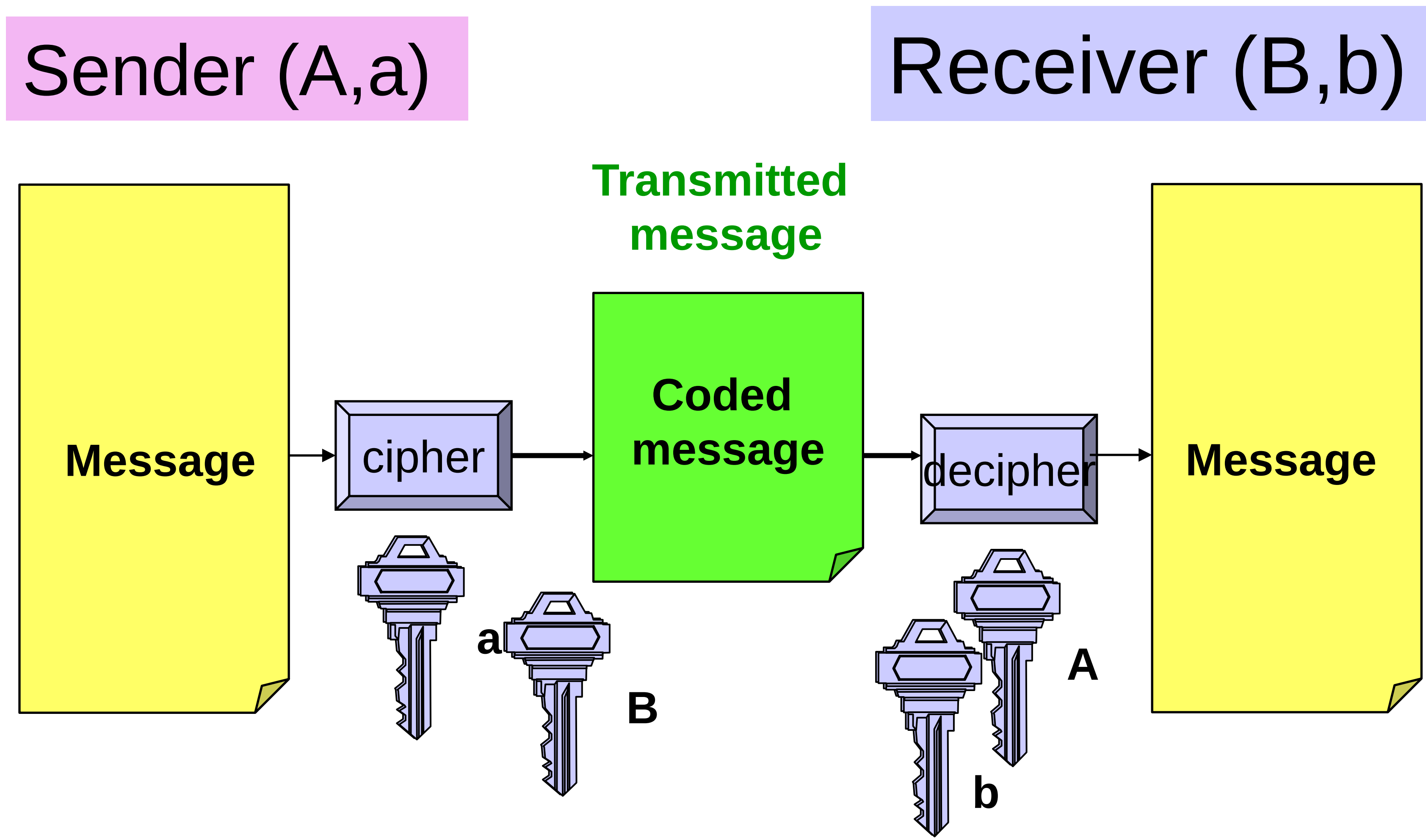
Guarantees confidentiality without sender guarantee

Public-private pair for authentication



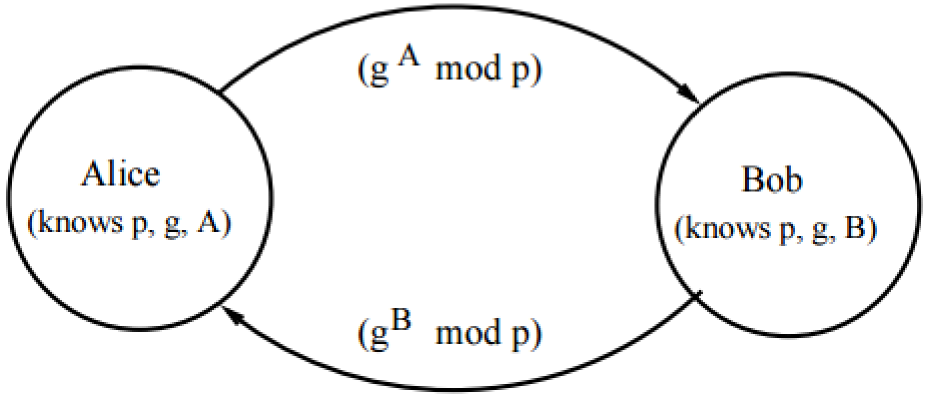
Authenticates the sender, since only the sender will have the secret key a, corresponding to the public key A

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 ($\mathbf{g}^{\mathbf{a}}$mod**p**).
- Bob chooses the secret number **b**, and sends to Alice ($\mathbf{g}^{\mathbf{b}}$mod**p**).
- Alice calculates (($\mathbf{g}^{\mathbf{b}}$mod**p**) $^{\mathbf{a}}$ mod **p**).
- Bob calculates (($\mathbf{g}^{\mathbf{a}}$mod**p**) $^{\mathbf{b}}$ mod **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 $5^6 \bmod 23 = 8$.
- Bob chooses $b = 15$ and sends $5^{15} \bmod 23 = 19$.
- Alice calculates $19^6 \bmod 23 = 2$.
- Bob calculates $8^{15} \bmod 23 = 2$.

- 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**}

https://www.comparitech.com/blog/information-security/rsa-encryption/

Example RSA

1. Selects prime numbers: $\boldsymbol{p}$=17 & $\boldsymbol{q}$=11
2. Calculates $\boldsymbol{n} = pq$ =17×11=187
3. Calculates $\phi(n)=(p-1)(q-1)$=16×10=160
4. Selects e *:* gcd(**e**,160)=1; chooses *e*=7
5. Determines d*:* $\boldsymbol{d}e$=1 mod 160 and d < 160: value is d=23 because 23×7=161= 160+1
6. Publishes public key KU= {**e**,**N**}={7,187}
7. Maintains secret key KR={**d**,**p**,**q**} = {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 $0 \le M < N$
- To decipher C :
  - Uses its private key $KR=\{d,p,q\}$
  - Calculates: $M=C^d \bmod N$
- Message M has to be lower than 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

1213107243921127189732367153161244042847242763370141092563454931230196
4373042085619324197365322416866541017057361365214171711713797974299334
871062829803541

q

1202752425547874888595622079373451212873338780368207543365389998395517
9850988797899869146900809131611153346817050832096022160146366346391812
470987105415233

n

1459067680075833232301869393490706352924018723753571643995818710198734
3879900535893836957140267014980212181808629246742282815702292207674690
6543401224889672472407926969987100581290103199317858753663710862357656
5105078837142971156373427889114635351027120327651665184117268598379886
72111837205085526346618740053

$\phi(n)$

1459067680075833232301869393490706352924018723753571643995818710198734
3879900535893836957140267014980212181808629246742282815702292207674690
6543401224889648313811232279966317301397777852365301547848273478871297
2220585874571528916064592697181119268971163555070802643999529549644116
811947516513938184296683521280

e - the public key

65537

d - the private key

8948942500927444436822854592177309391966958606588425744549785445648767483962981839093494197326287961679797060891728367987549933157416111385408881327548811058824719307758252727843790650401568062342355006724004246665654232383502922215493623289472138866445818789127946123407807725702626644091036502372545139713

Encryption/Decryption

Encryption: $19766202164023008896244827187751 50^e \bmod n$

35052111338673026690212423937053328511880760811579981620642802346685810623109850235943049080973386241113784040794704193978215378499765413083646438784740952306932534945195080183861574225226218879827232453912820596886440377536082465681750074417459151485407445862511023472235560823053497791518928820272257787786

Decryption:

$350521113386730266902124239370533285118807608115799816206428023466858106231098502359430490809733862411137840407947041939782153784997654130836464387847409523069325349451950801838615742252262188798272324539128205968864403775360824656817500744174591514854074458625110234722355608230534977915189288202722577877 86^d \bmod n$

19766202164023008896244827187751 50 (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 (SOPKM)

- The scope of key management

Centralized certificate authority — Fully self-organized

- 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

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1195151

SOPKM: Public Keys and Public-Key Certificates

- If a user u believes that a given public key K_v belongs to a given user v, then u can issue a public-key certificate in which K_v is bound to v by the signature of u.
- Certificates are issued with a limited validity period T_v
- Each certificate contains its issuing and expiration times.

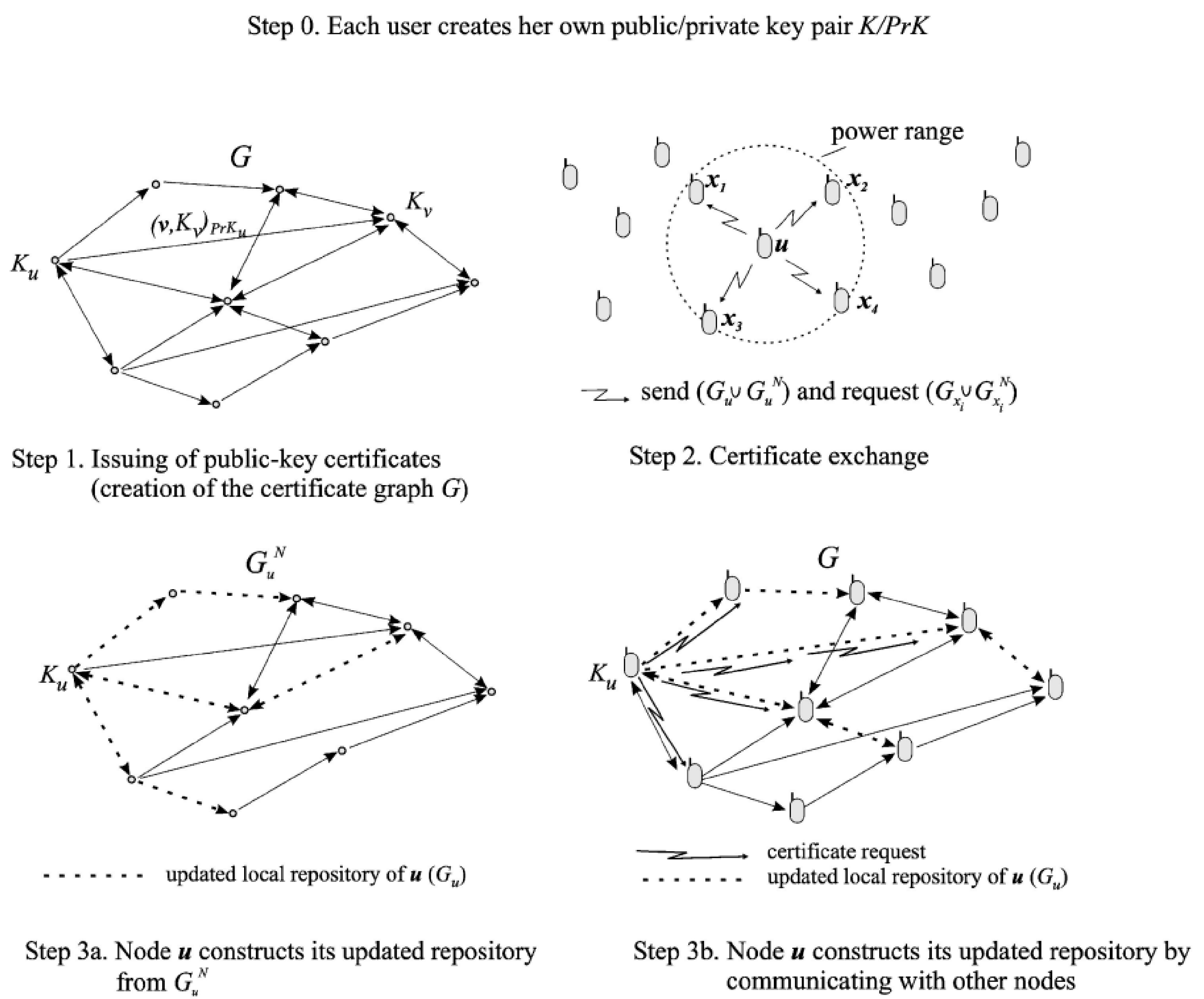
- u gets the public key K_v of v through a side channel
- u creates certificate for v to bind the K_v and v
- u stores the certificate of v in its local repository G_u and sends the certificate to v
- Each time a user u issues a certificate that binds another user v to her public-key K_v, u sends the certificate to v
  - *In this way, each certificate is stored at least twice: by its issuer and by the user to whom it is issued.*

SOPKM: Update repositories of certificate graphs

Certificate of u to v, using u private key pr $K_u(v, K_v)$

G_u Updated global certificate of u

G_u^N non-updated global certificate of u

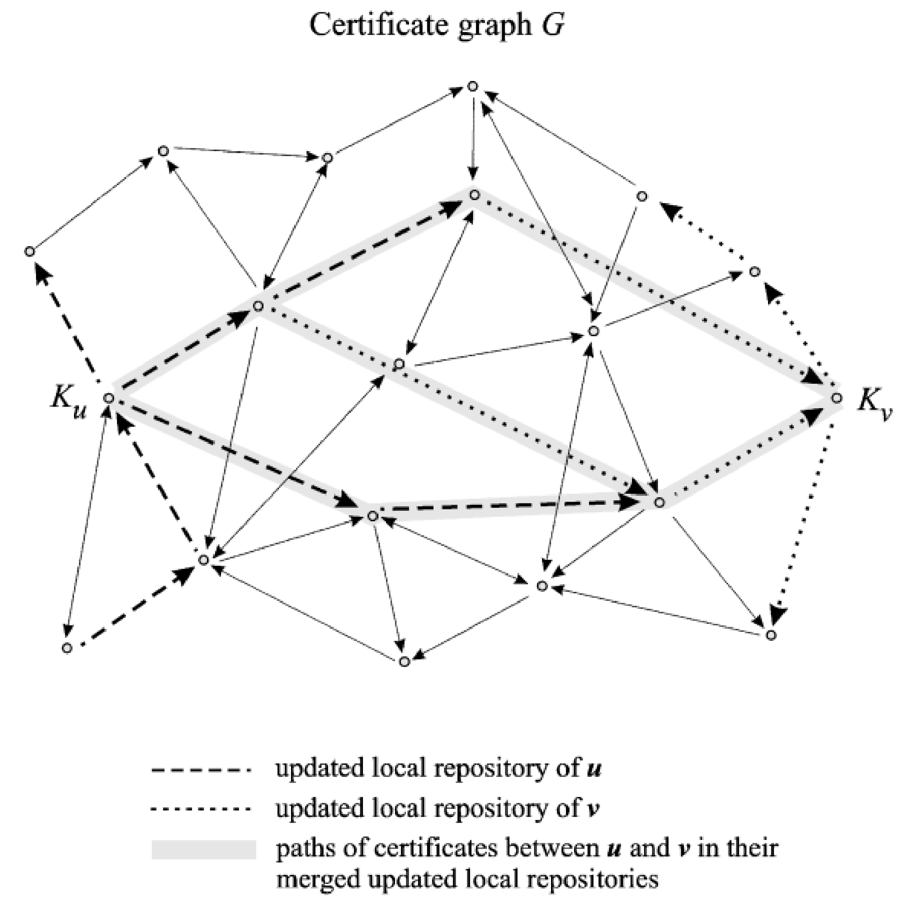


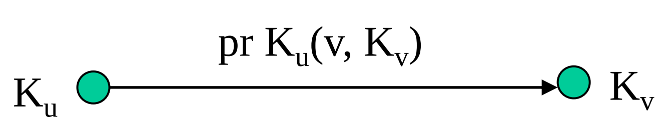
SOPKM: Certificate Exchange

- Each node *u* multicasts its subgraphs G_u and G_u^N to its physical neighbors (only 1 hop away)
- In this message, *u* does not send actual certificates, but only appropriate unique identifiers (e.g., their hash values)
- The neighbors of node *u* that receive the message from *u* reply with the hash values of the certificates in their updated and non-updated repositories.
- Node *u* then crosschecks the received values with the certificates that it holds and requests from its neighbors only the certificates that it does not hold.

SOPKM: Global Connectivity Graph

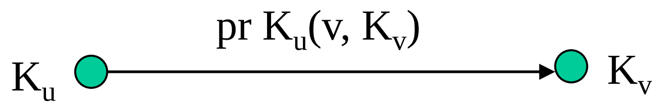
- Global certificate graph
- Connectivity of certificate graph
  - Small world phenomenon
    - Due to social relationships
    - Mobility increases the connectivity
      - Mobile nodes/users exchange their public keys whenever they meet
- Vertices: public keys of some nodes
- Edges: public keys certificates issued by users



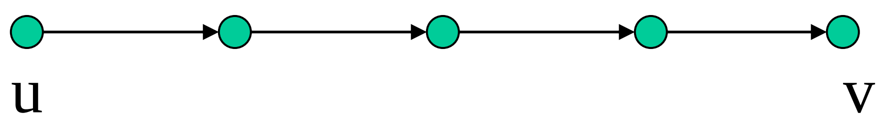


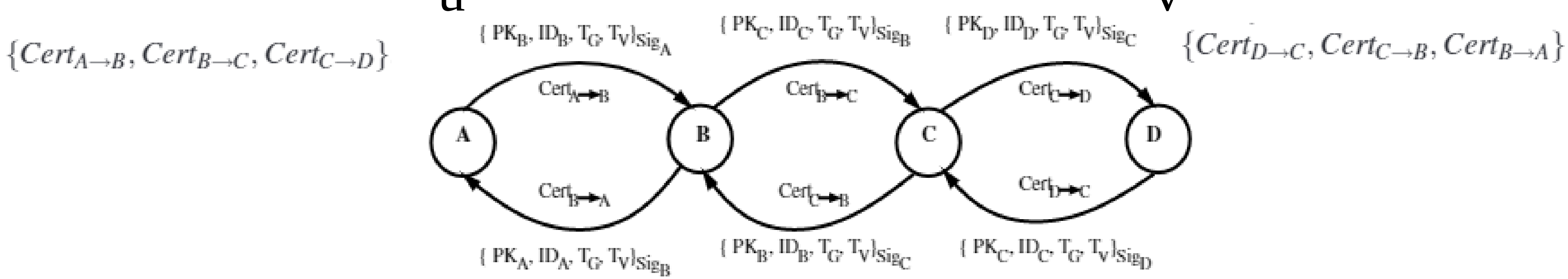
SOPKM: Certificate graph

- Vertices: public keys of the 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





SOPKM: Certificate 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

SOPKM: Malicious Users

- The certificate exchange mechanism allows nodes to gather virtually all certificates from *G*
  - Nodes cross-check user-key bindings in certificates that they hold and detect any inconsistencies (i.e., conflicting certificates).
  - If they contain inconsistent user-key bindings (i.e., if both certificates contain the same username but different public keys)
  - If they contain the same public-key, but are bound to different usernames.
- Several certificate exchanges to solve conflicts, if they are needed

Self-securing ad-hoc wireless networks (SSAWN)

- Usually, an entity is trusted only if it is verified by a central authority, which cannot be the case in wireless and ad-hoc networks
- Goal of a self-securing network
  - Achieve high security assurance
  - High success ratio
  - Efficient communication
- Localized trust model, an entity is trusted if any *k* trusted entities claim so within a certain time period
  - *k* entities typically among the entity's one-hop neighbors
    - Cares most the trustworthiness of its immediate neighbors in practice - a node will communicate with the rest of the world via its one-hop neighbors.
  - Once a node is trusted by its local community, it is globally accepted as a trusted node.
  - Otherwise, a locally distrusted entity is regarded as untrustworthy in the entire network.

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1021731

SSAWN: Shared secrets

- 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)
- SK is used to sign certificates for all nodes in the network.
- A certificate signed by SK can be verified by the well-known public key P K.
- 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)

SSAWN: Basic Operation

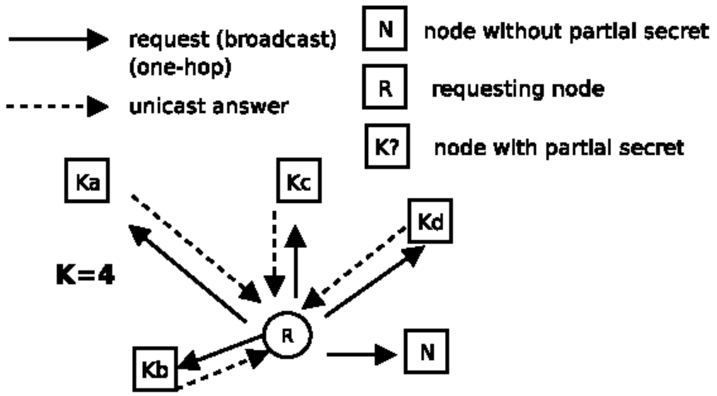
- Basic operation
  - Distributed PKI
    - System private key is split to server nodes
    - Quorum of *k* servers produce certificate updating
    - Structure of certificate

ID_i	T_{valid}	K_i	*flag*	*ver*	*sign.*	*issuer*	*Algo.*

  - Operates in phases
    - Server group formation/maintenance
    - Certificate updating/revocation
    - Shared key updating/renewing
- SK is not visible, known or recoverable by any network node
- Each node carries a certificate signed with SK.
- PK is assumed to be well-known for certificate verification.
- Nodes without valid certificates are denied from access to any network resources such as routing and packet forwarding

SSAWN: Shared secrets

- Partial secret key as a 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



SSAWN: Shared secrets

- Upon the receipt of v_i's certification request, a node checks its records.
  - If its record shows v_i as a well-behaving legitimate node, it returns a "partial" certificate by applying its share of SK.
  - Otherwise, the request is dropped.
- By collecting *k* partial certificates, v_i combines them together to generate the full new certificate as if it were from a CA server
  - Upon receiving *k* partial certificates from coalition, node v_i multiplies them together to recover its full certificate (lagrange interpolation – polynomial that passes through the several points, https://mathworld.wolfram.com/LagrangeInterpolatingPolynomial.html)
- A misbehaving or broken node will be unable to renew its certificate.
- A valid certificate represents the trust from a coalition of *k* nodes
  - Nodes with valid certificates are globally trusted.
  - Each node contributes to the overall trust management and maintenance by monitoring and certifying its neighboring nodes