



# 11. WSN Security

## Security in sensor networks, key exchange, verification

Wireless sensor networks

Martin Úbl  
ublm@kiv.zcu.cz

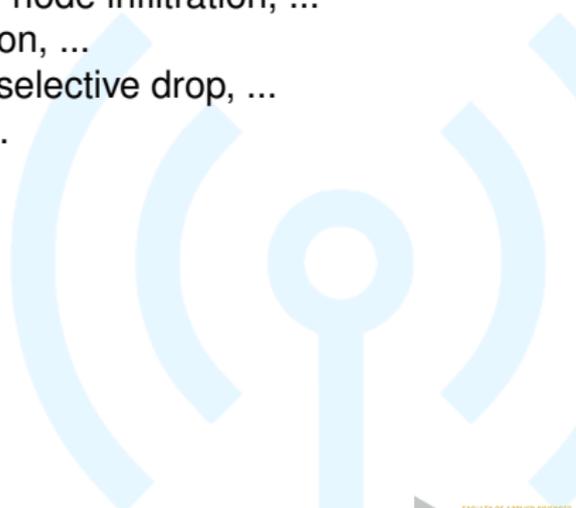
2023/24

# Security

- ❑ WSN's are networks like any other
- ❑ security must be taken into consideration
- ❑ nodes must detect and/or prevent any attempt to compromise the operation
- ❑ the methods differ from the ones used in ordinary networks

# Security

- ❑ we must consider security basically on all layers:
  - ❑ physical – jamming, noise generation, ...
  - ❑ data link – false neighborhood, node infiltration, ...
  - ❑ network – sink attack, redirection, ...
  - ❑ transport – congestion attack, selective drop, ...
  - ❑ application – data poisoning, ...



# Security

- ❑ assumptions:
  - ❑ low-power operation, limited battery life
  - ❑ computational and memory resources very limited
  - ❑ often deployed to open space – reachable by humans, ...
- ❑ each attack targets basically every possible limitation
- ❑ we are often willing to "sacrifice" something, if security is of main interest

# Security

- ❑ Problems:
  - ❑ key generation
  - ❑ encryption / decryption
  - ❑ key distribution
  - ❑ key exchange (shared key)
  - ❑ signature and signature verification
  - ❑ hashing
  - ❑ privacy
  - ❑ safe routing
  - ❑ physical security



# Security

- ❑ basic requirements:
  - ❑ authentication
  - ❑ privacy
  - ❑ integrity
  - ❑ reliability
  - ❑ availability
  - ❑ recent data



# Security

- ❑ what else to consider:
  - ❑ large node count, no individual monitoring – one node might get stolen
  - ❑ one compromised node may compromise the whole network
  - ❑ eavesdropping on communications
  - ❑ if someone really wants to compromise our network, he will eventually succeed
    - ❑ the same holds true for virtually every network
  - ❑ to compromise an encryption, one usually needs a large amount of data
    - ❑ WSN transfer just a few bytes per payload – not much

# Security

## Authentication

- ❑ *authentication*
  - ❑ verifying the identity of the peer node (end node, ...)
  - ❑ helps with identifying injected traffic
  - ❑ we usually use Message Authentication Codes (MAC)
  - ❑ in conjunction with hashing and encryption
  - ❑ some digital signature algorithms may be expensive

# Security

## Privacy

- *privacy*
  - no unauthorized node can read the data
  - i.e., only authorized nodes can interpret the data
  - we use encryption to ensure privacy
    - symmetric, asymmetric?
  - we encrypt whole packets or parts of the packet
    - encrypting whole packets encrypts the overheads, thus reducing the probability of compromising the protocol

# Security

## Integrity

- ❑ *integrity*
  - ❑ messages are not altered (either intentionally or unintentionally)
  - ❑ a very dangerous would be, if the attacker altered clock synchronization, localization, key exchange, routing information, ...
  - ❑ altering of aggregate data
  - ❑ we usually use MAC



# Security

## Availability

### □ *availability*

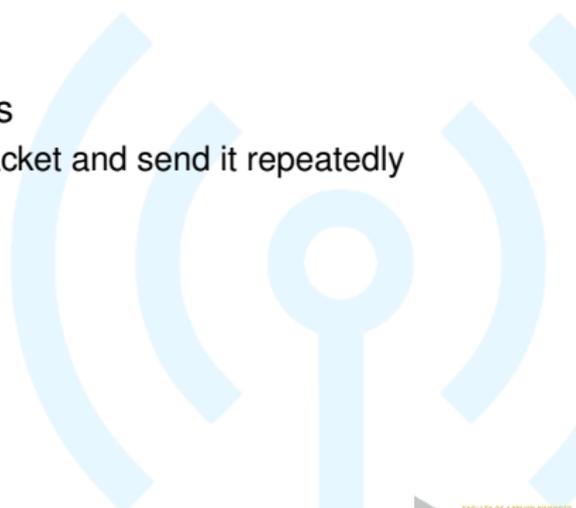
- we must ensure operational state of the WSN
- at least during mission time
- availability of node / network
- this also includes DoS attack protection
- in general, we aim to have high availability from the nature of WSN



# Security

## Recent data

- *recent data*
  - we aim to have recent data in all nodes / on the edges
  - early event detection
  - reliable transfers
  - we need to avoid replay attacks
    - if the attacker capture the packet and send it repeatedly



# Security

## Cryptography

- ❑ cryptography in WSN
- ❑ shares the same base principles as in "big world"
  - ❑ asymmetric cryptography
    - ❑ slow
    - ❑ complex
    - ❑ safe
  - ❑ symmetric cryptography
    - ❑ faster
    - ❑ needs shared key (distribution problems)
    - ❑ potentially less safe
  - ❑ message authentication codes
    - ❑ often simple
    - ❑ requires shared secret
  - ❑ digital signatures
    - ❑ more complex, slow
    - ❑ does not require shared secret (pre-shared public key suffices)

# Security

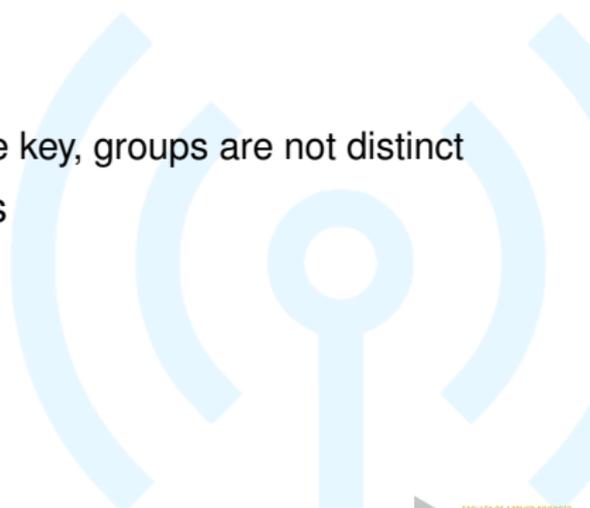
## Cryptography

- ❑ embedded MCUs often have a cryptography coprocessor
  - ❑ *AES coprocessor, RSA coprocessor, Security coprocessor, Crypto core, ...*
- ❑ it is very convenient to use it
- ❑ increases security – avoids known bugs
- ❑ reduces energy consumption – specialized instructions or direct hardware support
- ❑ we don't need to implement the algorithm on our own
- ❑ often much faster

# Security

## Key management

- ❑ let us imagine a scenario, when we need to have encryption keys for all nodes
- ❑ there are several cases:
  - ❑ every node has the same key
  - ❑ every node has its own key
  - ❑ a group of nodes has the same key, groups are not distinct
- ❑ every case has its pros and cons



# Security

## Key management

- ❑ case 1: symmetric cryptography, every node has the same key
- ❑ basically nonsense
- ❑ the key may leak very quickly
- ❑ e.g., somebody captures a single node and extracts the key from memory
  - ❑ then, he could impersonate master node

# Security

## Key management

- ❑ case 2: asymmetric cryptography, every node has the same key
- ❑ specifically – one key pair for master node (data sink, ...), one key pair for all nodes
- ❑ slightly more secure, but shares the same problem as in case 1
- ❑ one small difference – cannot impersonate master node with regular node private key

# Security

## Key management

- ❑ case 3: symmetric cryptography, every node has its own key
- ❑ much safer
  - ❑ or is it?
- ❑ compromising a single node allows for impersonation of just a single node
  - ❑ or does it?
- ❑ remember, WSNs are often multi-hop
- ❑ therefore, nodes must communicate with each other
- ❑ → nodes must store *symmetrical* keys of its neighbors
- ❑ thus, it is not much safer, **attacker could impersonate a lots of nodes**
- ❑ additionally, we don't have much space to store lots of keys
- ❑ this may be even worse than cases 1 and 2

# Security

## Key management

- ❑ case 4: asymmetric cryptography, every node has its own key
  - ❑ even more safe
  - ❑ compromising a single node allows for impersonation of just a single node
    - ❑ this time for real
  - ❑ a node must store its own private key
  - ❑ a node must store a table of public keys of other peer nodes
  - ❑ a key of a reasonable length might have at least 1 kB
    - ❑ we often do not have more than a few kB of flash memory
  - ❑ though this is safer in theory, we cannot achieve it in practice

# Security

## Key management

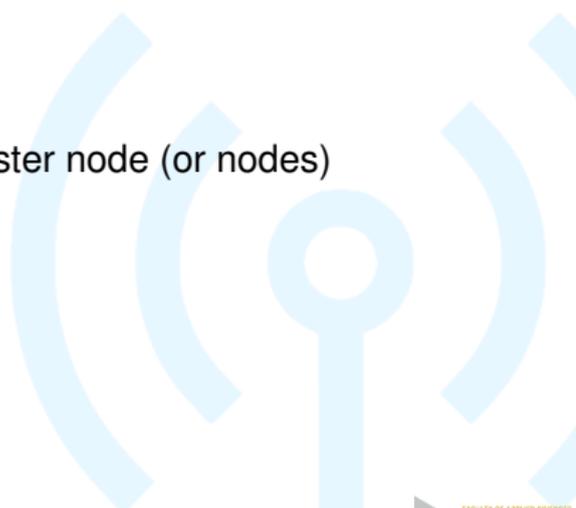
- we need another scheme
- something reasonably safe
- something, that respects memory constraints of embedded devices
- key distribution schemes*



# Security

## Key management

- key distribution
  - pre-shared
  - centralized
  - decentralized
- distribution methods
  - based on master key
  - based on cooperation with master node (or nodes)
  - based on third-party trust
  - fully decentralized



# Security

## Master key

- ❑ *master key*
- ❑ pre-shared master key
- ❑ node key is created from a random number and distributed as encrypted with the master key
- ❑ problem: compromised master key = compromised network
- ❑ potential resolution: delete the master key upon receiving node key
- ❑ limited scalability

# Security

## Key distribution

- ❑ probabilistic storage
- ❑ we do not store all keys, but just a subset of keys
- ❑ we randomly choose a few keys to store
- ❑ we communicate with just those nodes
- ❑ upon selection, construct a safe routing table
- ❑ eventually – we reach the master node
- ❑ disadvantage: what if we choose very distant nodes?
  - ❑ energy consumption goes way up

# Security

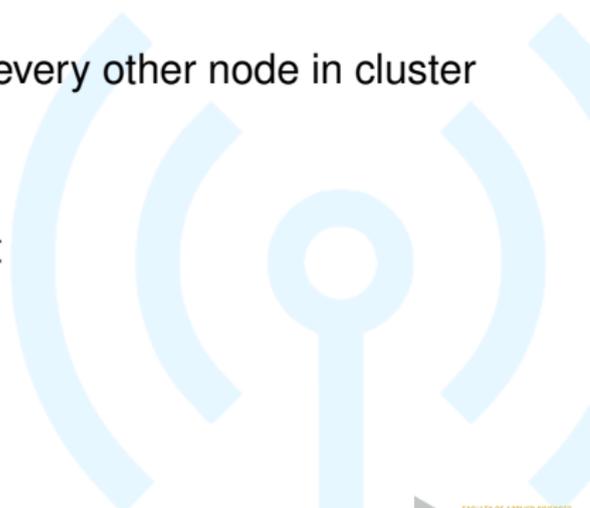
## Key distribution

- cluster-based trust
- nodes are clustered before propagating keys
- cluster heads have a public key of upper level node (parent)
- every node in cluster stores cluster head public key
- when clusters are small enough, it could be very efficient

# Security

## Key distribution

- probabilistic cluster-based storage
- nodes are clustered before propagating keys
- every cluster is guaranteed to have a node, that is able to communicate with upper level
- every node in cluster can reach every other node in cluster (not directly)
- allows for larger clusters
- increases delays, not so efficient
- a bit safer



# Security

## Implementation

- on what layer to implement encryption?
- physical – somehow possible, if we consider e.g., frequency hopping as a form of security
- data link – possible, practical, but imposes unwanted overhead; hop-by-hop security
- network – end-to-end encryption
- transport – "service-to-service" encryption – not very useful
- application – data-level security – does not protect routing and data link information

# Security

## Implementation

- on what layer to implement integrity protection and authentication?
- physical – no chance
- data link – possible, hop-by-hop authentication
  - potentially energy consuming
  - better to use a forward error correction here (faster, more benefits)
- network – possible, end-to-end authentication
  - good balance between security and performance
- transport – "service-to-service" integrity – not very useful
- application – data-level integrity and authentication – same as above, does not protect L2/L3 layers

# Security

## Final remarks

- we will talk more about security in the next lecture
- we will consider attacks on all layers
- attempt to design a solution to protect the WSN against all of them

