

PA197 Secure Network Design



Cryptography aspects in Wireless Sensor Networks

Lukáš Němec lukas.nemec@mail.muni.cz, Petr Švenda
Faculty of Informatics, Masaryk University

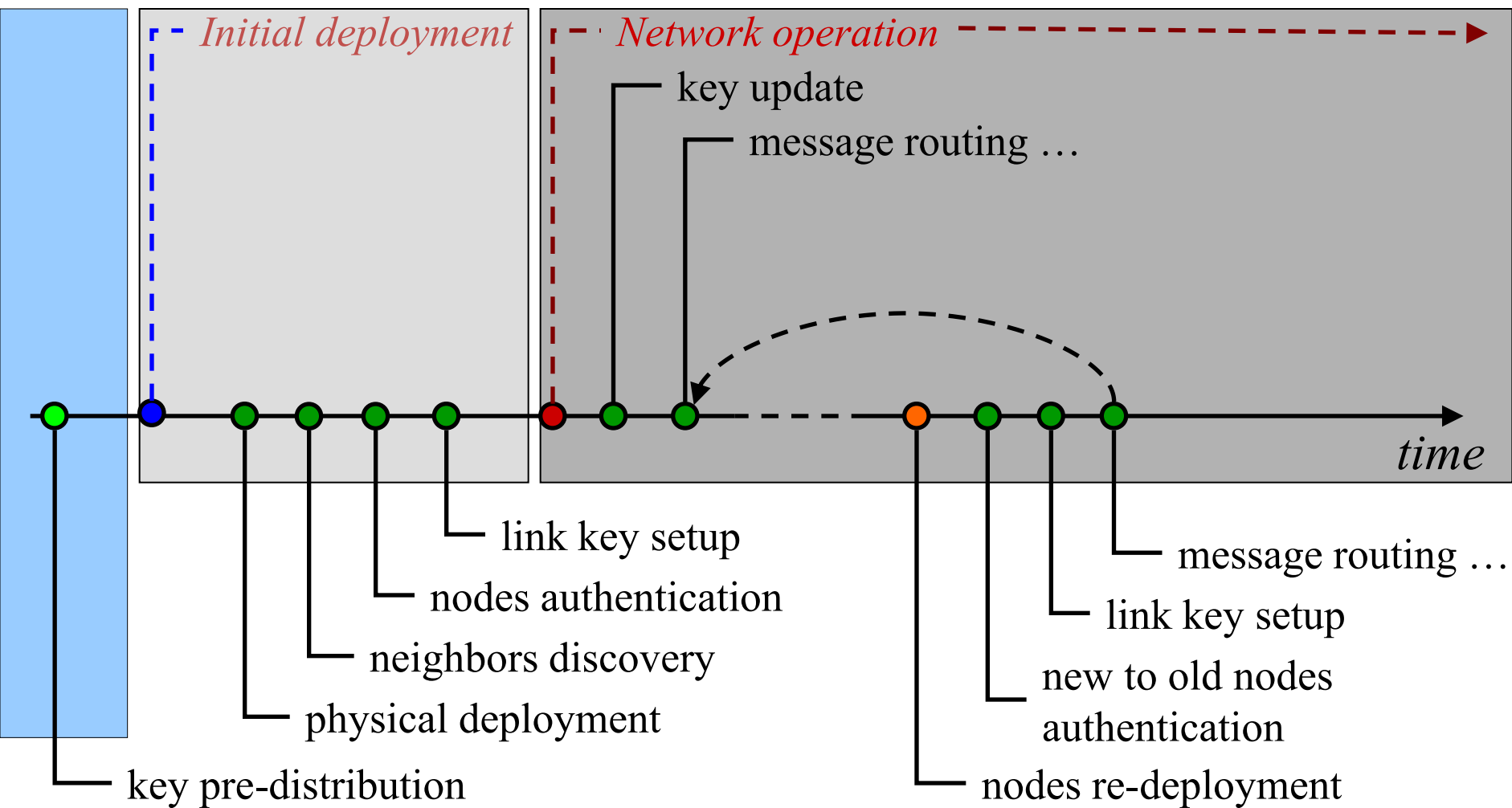
CRCS

Centre for Research on
Cryptography and Security

Lecture overview

- Cryptography and key management in WSNs
 - Approaches and typical issues
- Partial compromise and what can be done
 - Dealing with partially compromised network
- Case study: WSNProtectLayer

Network lifetime



Wireless Sensor Networks – Crypto

CRYPTOGRAPHIC ASPECTS

Do we have need for on-node crypto?

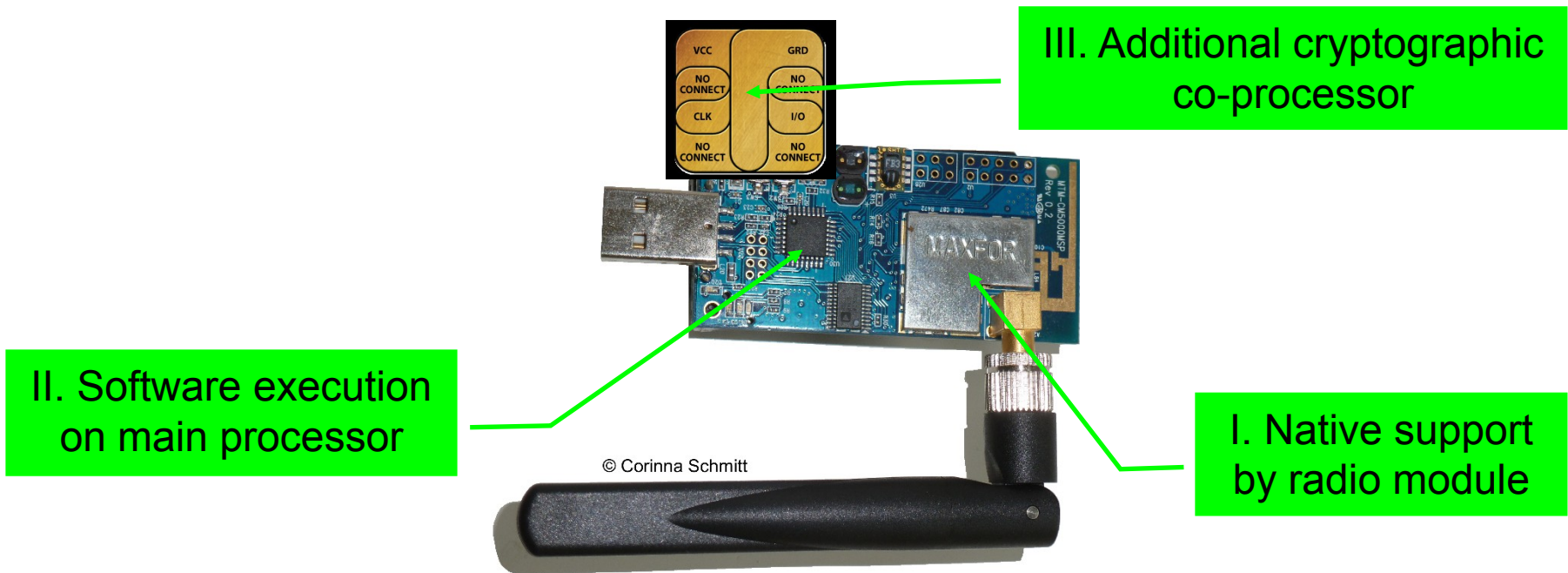
- Data for base-station (end-to-end)
- Data for neighbors (hop-by-hop encryption)
- Nodes authentication
- Authenticated broadcast
- Group/cluster-keys (aggregation)
- Traffic analysis resistance (phantom routing...)
- No-keys, symmetric crypto, asymmetric crypto
- Random number generation (IV, padding, keys...)

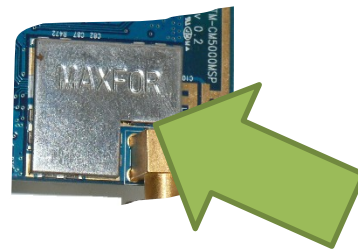
Recall: WSN specifics

- Limited computation power and memory
- Limited energy
 - Consumed by communication, computation, storage...
- Limited connectivity
- No direct central synchronization
 - Low-range radio
 - No or loosely synchronized clocks
- Limited or no tamper resistance

Native vs. software-only cryptography

- I. Native support inside radio module
- II. Software execution on main processor
- III. Additional cryptographic co-processor





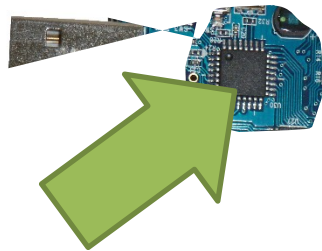
I. CRYPTO IN RADIO MODULE

Native cryptographic support by radio

- Cryptographic functionality provided by radio module
 - Supported algorithms depend on used standard, only very few
 - Usually easy to use and transparent to developer/user
 - Energy efficient (ASIC)
- Usually focus only on link-level security
 - Encryption, integrity (MAC), node authentication, key establishment
- Performance matched to radio's transmission rate
- Allows for better parallelization => lower latency
 - Main processor not occupied with cryptographic operation
- Customized crypto protocols usually not possible

Native cryptographic support - examples

- IEEE 802.15.4 (ZigBee, AES-128b)
 - AES-CBC-MAC-32/64 (no encryption, 4/8B MAC)
 - AES-CTR (CTR mode for encryption, no MAC)
 - AES-CCM-32/64 (encryption + MAC)
- Bluetooth LE/Smart (AES-128b, ECDH P-256)
 - AES-CCM (encryption + MAC)
 - ECDH (key establishment)
- ...



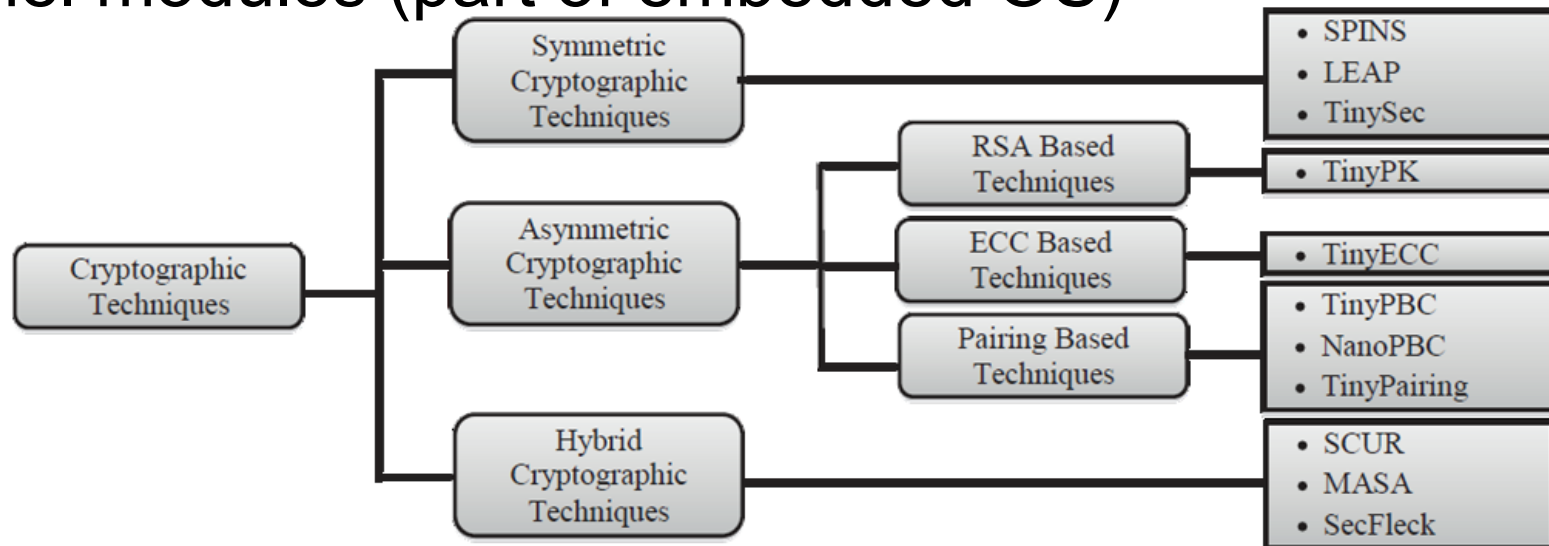
II. CRYPTO ON MAIN PROCESSOR

Crypto on main processor

- Cryptographic functionality executed on main processor
 - Performance highly depends on main processor
 - Usually less energy efficient and possibly slower than other options
- High flexibility: customized algorithms and protocols
 - Anything that can be compiled, fit and executed on MCU
 - Important parameters: code size (EEPROM), state (RAM), speed
- Introduces additional latency
 - Main processor occupied with crypto operation, serialization
- Possibility to update implementation in the field
 - Over-the-air (OTA) updates
- Keys can be extracted after node capture
 - no tamper resistance

Available implementations

1. Standalone algorithms (e.g., AES)
2. General-purpose libraries (mostly C)
3. Platform specific libraries (TinySec...)
4. Kernel modules (part of embedded OS)



Security Frameworks for Wireless Sensor Networks-Review

| Framework | Encryption | Cipher | Freshness (CTR) | Key Agreement | Code Requirement | Authentication | Cost (time/energy) | Support |
|-------------|---------------------|-------------------------------|-----------------|---------------------------------|--|---------------------------------------|---|--|
| SPIN | CTR mode | RC5 (Block) | Yes | Master Key & Delayed Disclosure | 2674B | CBC-MAC | 7.24 ms | SmartDust |
| LEAP | RC5 | RC5 (Block) | No | Pre-deployed (Master Variable) | ROM: 17.9KB RAM: no. of neighbours | CBC-MAC | Variable (No. of neighbours) | Mica2 |
| TinySec | CBC mode (Optional) | Cipher independent | No | Any | RAM: 728B program space: 7146B | CBC-MAC | RC5(C): 0.90ms Skipjack(C): 0.38ms RC5(C, assembly): 0.26ms | Mica, Mica2, & Mica2Dot |
| TinyPK | RSA | - | No | PK-RSA | 13387B (512 bit key) | CA-signed Diffie-Hellman public value | 3.8 s | Mica1, Mica2 |
| TinyECC | ECIES | - | No | ECDH | 20818B (micaz) | ECDSA | 20266.47ms / 486.4 mJ (micaz) | Mica2/MicaZ, TelosB/Tmote Sky, BSNV3, & Imote2 |
| TinyPBC | PBC | - | No | ID-NIKDS | Stack: 2,867B RAM: 368B ROM: 47,948B | ID-NIKDS | 5.45s (pairing computation) | Mica2 & MicaZ |
| NanoPBC | - | - | - | - | - | - | - | MicaZ |
| TinyPairing | PBC | - | No | - | RAM: 392B ROM: 21,742B | - | 21.95 s | MICA family, Telos, eyesIFX, Intel's Imote, |
| SCUR | Rabbit | Rabbit (Stream) (128bit) | Yes | Pre-Deployed key | - | - | - | - |
| MASA | - | - | No | - | - | - | - | - |
| SecFleck | RSA | RSA (Block) (2048 bit) & XTEA | No | - | RAM: 52B Program space: 1.082B | RSA | RSA (s/w): 219,730 μ s / 7,030.0 μ J RSA (h/w): 27 μ s / 5.4 μ J XTEA (s/w): 18 μ s / 0.6 μ J | Fleck sensor node |

Modes for encryption / integrity

- CBC used often in software libraries (simple)
 - Need for initialization vector update and synchronization
- CTR mode
 - possibility for precomputation => lower latency when packet arrives
 - No message length extension
 - (used also in Bluetooth LE / IEEE 802.15.4 ZigBee)
- CBC-MAC - same underlying code reused

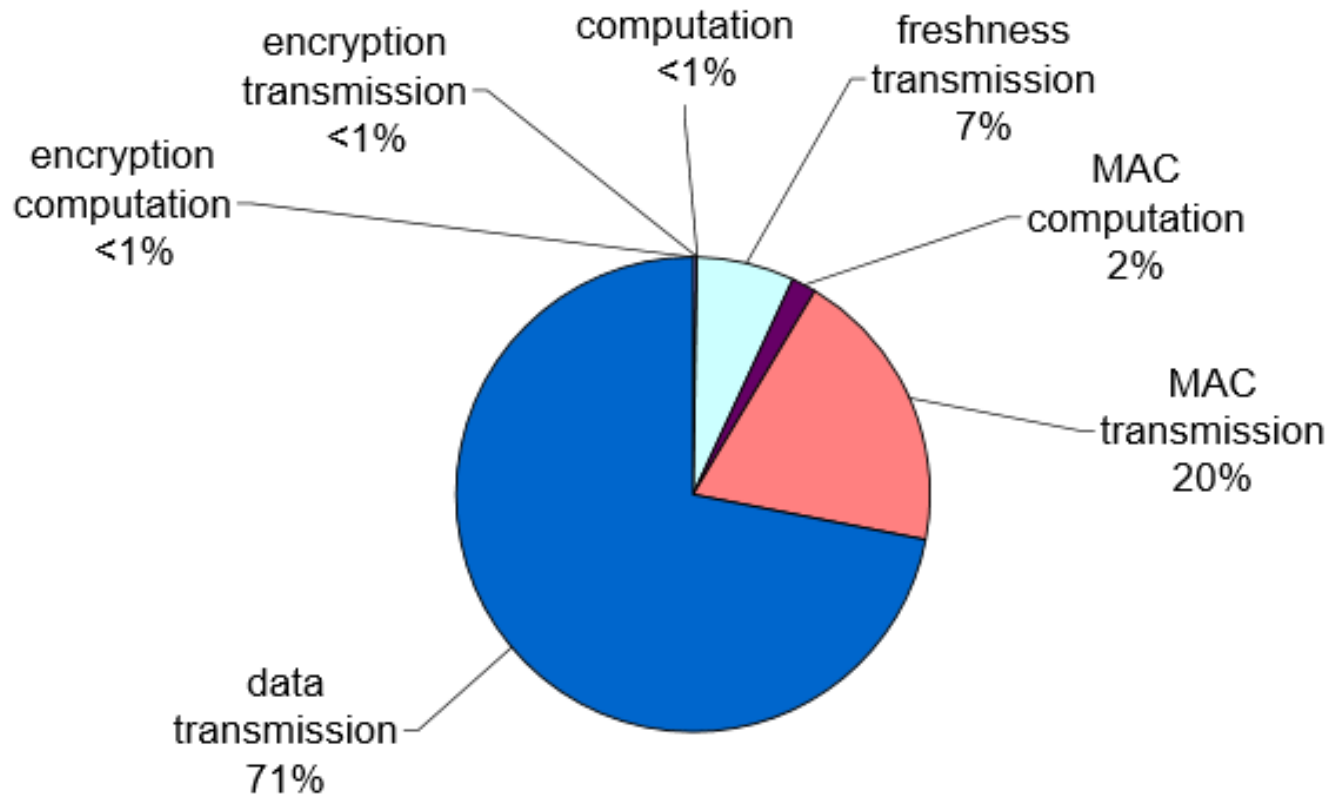
Initialization vector management

1. IV is send with every packet
 - Shorter than normally (e.g., 2 bytes only), ~10% overhead
 - Relatively low number of messages (65k) before key update
 - Advantage in high packet loss environments
 - Example: TinySec, ZigBee
2. IV is kept synchronized (counter), no IV send
 - Resynchronization on packet loss required
 - Example: SPINS
3. Only part of IV send (last few bits)
 - Balance between overhead and expected number of lost packets
 - Example: MiniSec-U

SPINS/SNEP (Perrig et al., 2002)

- Suite of lightweight protocols
 - Based on symmetric cryptography only, RC5 (stream c.)
- SNEP: Sensor Network Encryption Protocol
 - Semantic security, Data authentication
 - Replay protection – synchronized counters
 - Freshness – weak (counter), strong (challenge)
 - Low communication overhead
- De-facto benchmark for protocols proposed later

SPINS – energy consumption



<http://users.ece.cmu.edu/~adrian/projects/mc2001/mc2001.pdf>

Asymmetric crypto – energy consumption

- Significantly different ratio w.r.t. symmetric crypto
 - Most energy consumed by computation of operation (MCU)
 - Transmission accounts only to about 1% of energy use
 - Even when significantly longer signature is transmitted
 - 128B RSA signature vs. 4-8B MAC
- Overall impact on network lifetime is still very small
 - Relevant only to networks with high number of signed messages
- More important factors are code size, state and increased probability of collision during transmission
- https://www.ics.uci.edu/~steffenp/files/SASN_piotrowski.pdf

Authenticated Broadcast

- Authenticated message to be delivered to “all” nodes
- Solution1: Asymmetric crypto
 - Potentially high computation and transmission overhead
- Solution2: Single network-wide key for MAC verification
 - Single compromised node => attacker can forge BS’s messages
- Solution3: Unique key between every node and BS
 - Compromised node => only messages to this node can be forged
 - But separate message (or at least MAC) for every node needs to be computed and delivered (significant overhead)
- Can we use symmetric crypto and have only single key?

μTesla: Authenticated Broadcast

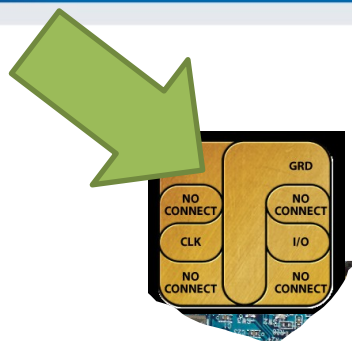
1. Message broadcasted from base station with MAC
 - Node stores received message, but cannot verify yet
2. Base station later broadcasts key used for MAC (“epoch”)
 - Once broadcasted, nodes can verify messages from given epoch
 - New messages from previous epoch are not accepted any more
 - As MAC key for that epoch is now public
3. Message authentication keys form hash key chain
 - No need to store keys for older epochs
 - Validity of MAC keys can be verified against pre-distributed root

Hash chains (as used in μ Tesla)

- $\text{root} = H^1(H^2(\dots H^X(\text{seed})\dots))$
- Knowledge of root will not allow to compute any H^i
 - Inversion of hash function H is hard
- H^i can be quickly verified against H^{i-1}
 - Unlimited length of chain (if root is not required)
 - Length X chosen in advance (if root is pre-distributed)
- Knowledge of *seed* allows to compute any chain value
 - Used by base station for MAC key computation
- *root* used for verification of μ Tesla MAC keys
 - By deployed nodes

μ Tesla properties

- Very low overhead (MAC/message + key/epoch)
- Requires loosely synchronized clock ("epochs")
- Robust against packet loss
- Overhead independent from number of nodes



Tamper Resistant Hardware and Asymmetric crypto on WSN node

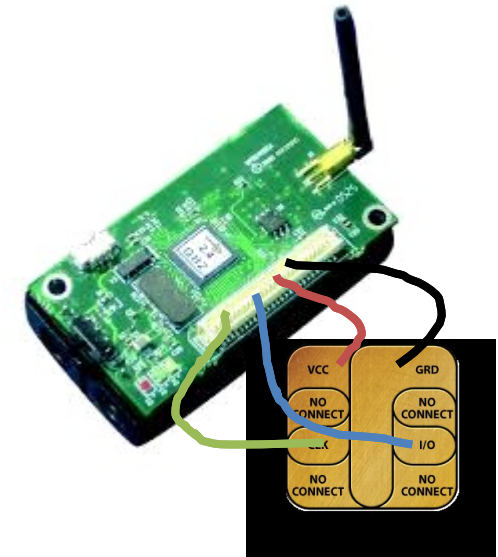
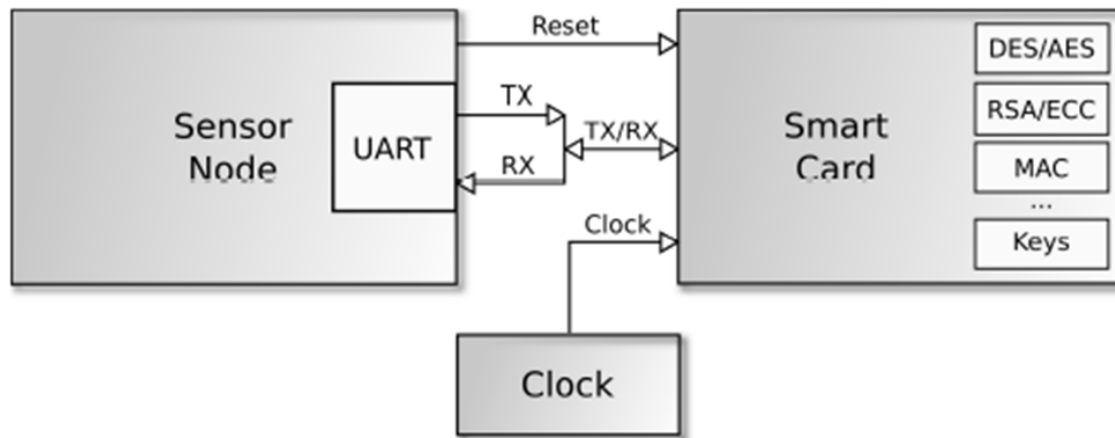
III. CRYPTO CO-PROCESSOR

Cryptographic co-processor

- Additional dedicated co-processor for crypto ops
 1. Only cryptographic speedup (no tamper protection)
 2. Also tamper protection of cryptographic secrets
- Possibility to parallelize (MCU/Crypto/Radio)
- Small to medium flexibility (fixed set of algorithms)
- Energy efficient
- E.g., cryptographic smart card provides:
 - Strong tamper resistance, RSA-1024/2048, ECC...
 - Strong protection also for keys for symmetric crypto
 - Relatively cheap (\$2, Feitian A40 Infineon SLE78)

Smart card to sensor node connection

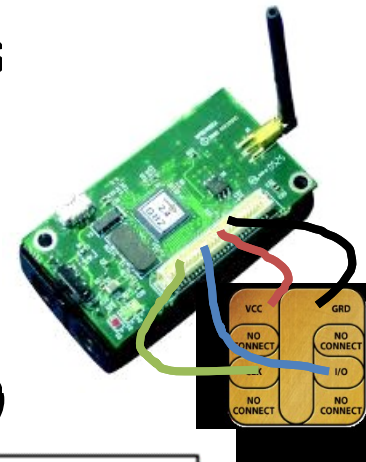
- Direct connection via serial interface (UART)
 - Communication speed 9600 baud, APDU commands
- Keys and crypto operation executed only on-card



Hanáček, Nagy, Pecho: *Power Consumption of Hardware Cryptography Platform for Wireless Sensor*, IEEE CS, 2009, s. 6, ISBN 978-0-7695-3914-0

Performance with cryptographic smartcard

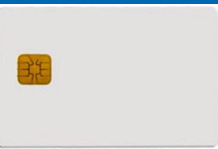
- Total time = $T(\text{dataIn}) + T(\text{operation}) + T(\text{dataOut})$
- Experiment: MICAz (ATmega128L), GemXpress
- Performance for RSA-1024b
 - 30x faster (750ms), 27x more energy efficient (27mW)
- Performance for RSA-2048b
 - 88x faster (1900ms), 70x more energy efficient (79mW)



| Algorithm and realization | | | Key length (bit) | T (s) | W (mWs) |
|--------------------------------|-----------|-----|------------------|--------|---------|
| Software solution ² | Signature | RSA | 1024 | 22.03 | 726.99 |
| Software solution | Signature | RSA | 2048 | 166.85 | 5506.05 |
| Hardware solution | Signature | RSA | 1024 | 0.75 | 27.15 |
| Hardware solution | Signature | RSA | 2048 | 1.89 | 79.09 |

Performance with newer cards

- Even faster with current cards and faster UART
 - 9600 baud → 128000 baud => 12x faster data I/O
- \$2 Feitian A40 smart card (Infineon SLE78)
 - 25ms per single RSA-1024b operation
 - 150ms per single RSA-2048b operation
- Expected performance
 - below 50ms (440x faster) for RSA-1024
 - below 200ms (900x faster) for RSA-2048
- Even cheaper and efficient ASICs available...



Multiple keys / engines can be stored

- <https://www.fi.muni.cz/~xsvenda/jcalgtest>

| Type of object | NXP CJ2A081 | NXP CJ2D081 80K | NXP JCOP21 v2.4.2R3 145KB |
|----------------------|-------------|-----------------|---------------------------|
| AESKey 128 | 877 | 729 | 678 |
| AESKey 256 | 658 | 607 | 565 |
| DESKey 196 | 748 | 607 | 565 |
| Cipher AES | 79 | 74 | 74 |
| Cipher DES | 147 | 136 | 136 |
| RSA CRT PRIVATE 1024 | 72 | 93 | 86 |
| RSA PRIVATE 1024 | 203 | 152 | 141 |
| RSA CRT PRIVATE 2048 | 61 | 51 | 47 |
| RSA PRIVATE 2048 | 108 | 82 | 77 |

Wireless Sensor Networks – Key Distribution

KEY DISTRIBUTION

Problem: wide range of assumptions

- Different works assume different types of WSNs
 - network architecture and topology
 - network nodes hardware and required lifetime
 - degree of (de)centralism, level of nodes mobility
 - communication medium used, quality of links
 - computational power, memory limitations, energy source
 - routing and data collection algorithms
 - assumptions about attacker capabilities
 - ...
- One security approach doesn't fit all scenarios

Level of keys pre-distribution (I.)

1. No pre-distribution

- all keys established after nodes are deployed
- e.g., Key Infection (exchange keys in plaintext)
- problem: usually assumes period of limited attacker

2. Fixed network wide “master” key(s)

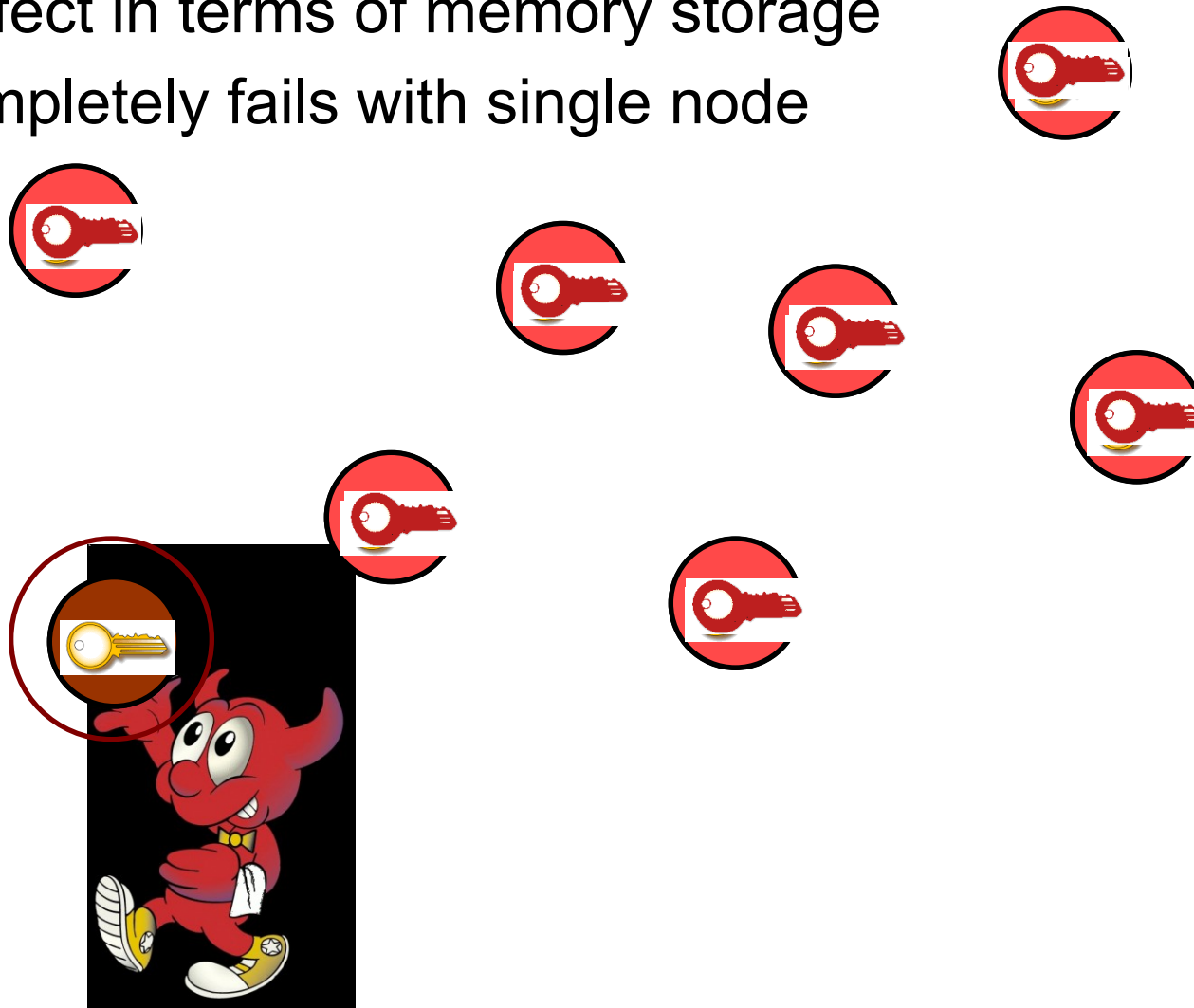
- pre-distributed keys allowing key establishment with all others
- problem: very low node capture resilience

2. Fixed network wide “master” key(s)

- Single master key shared by whole network
- All transmission encrypted/MAC by master key
 - What are possible attacks?
 - Reuse of key for long time, no origin authentication...
 - Compromise of master key (node capture)
- Link keys derived from master key
 - $\text{linkKey} = \text{KDF}(\text{nodeID1} \mid \text{nodeID2} \mid \text{random})$
- What attacks are possible?

Why “Master key” pre-distribution fails

- Perfect in terms of memory storage
- Completely fails with single node



Level of keys pre-distribution (II.)

1. No pre-distribution

- all keys established after nodes are deployed
- e.g., Key Infection (exchange keys in plaintext)
- problem: usually assumes period of limited attacker

2. Fixed network wide “master” key(s)

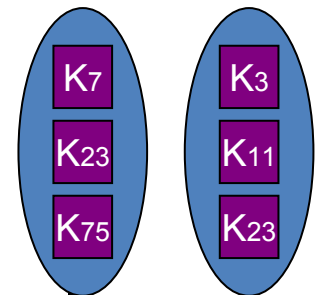
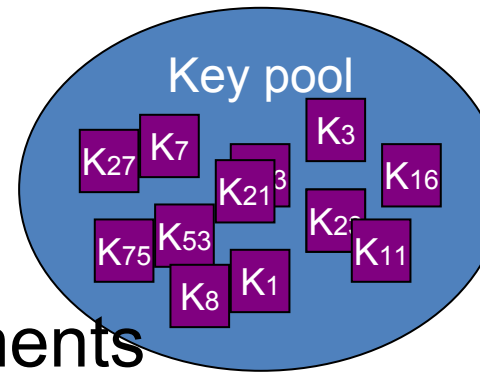
- pre-distributed keys allowing key establishment with all others
- problem: very low node capture resilience

3. Partial pre-distribution

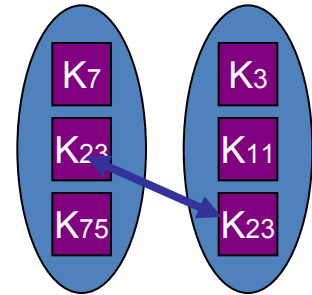
- not all nodes can establish key directly
- e.g., probabilistic pre-distribution [EG02]
- problem: node capture resiliency

Probabilistic key pre-distribution

- *Eschenauer & Gligor 2002*
- Elegant idea with low memory requirements
 - based on birthday paradox
 - large pool of cryptographic keys with unique IDs used
- For every node prior deployment:
 1. randomly select keys from large key pool
 2. return selected keys back to pool
 3. proceed with next node

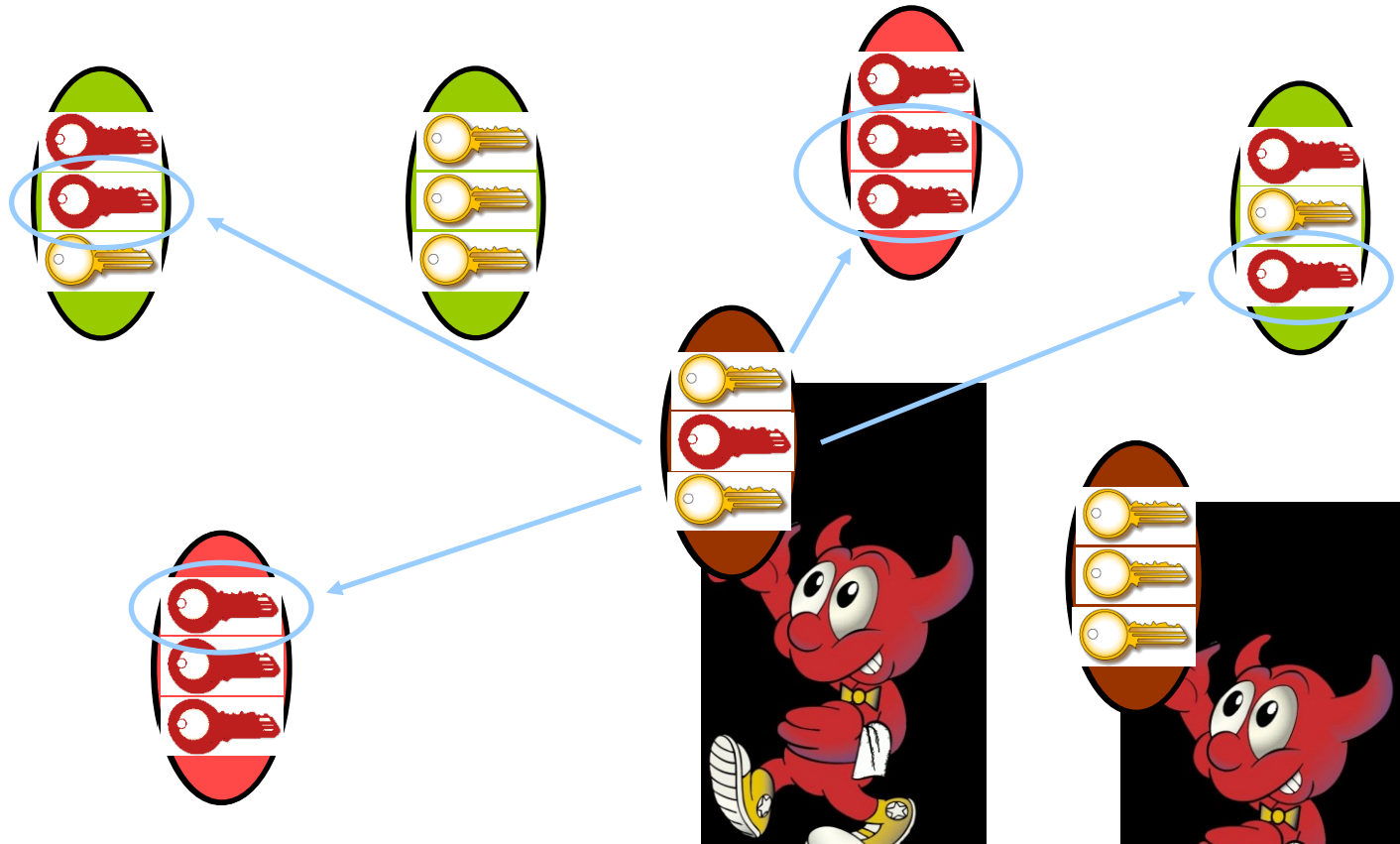


Probabilistic key pre-distribution (2)



- During neighbour discovery:
 1. neighbours establish radio communication
 2. nodes iterate over their keyrings for shared key(s)
 3. if shared (by chance) key(s) are found, secure link is established
 - e.g., 100 keys from 10000 \Rightarrow 60% probability at least one key shared
- Not all nodes can establish secure link
 - but sufficient connectivity probability can be set
- Node capture resilience (NCR) is a problem

How probabilistic pre-distribution fails



- Keys from uncaptured nodes compromised as well

Level of keys pre-distribution (III.)

4. Pairwise keys (node2BS, node2node)
 - all nodes can establish keys if necessary
 - Every node to BS, every node to every node

Pairwise keys – every node to BS

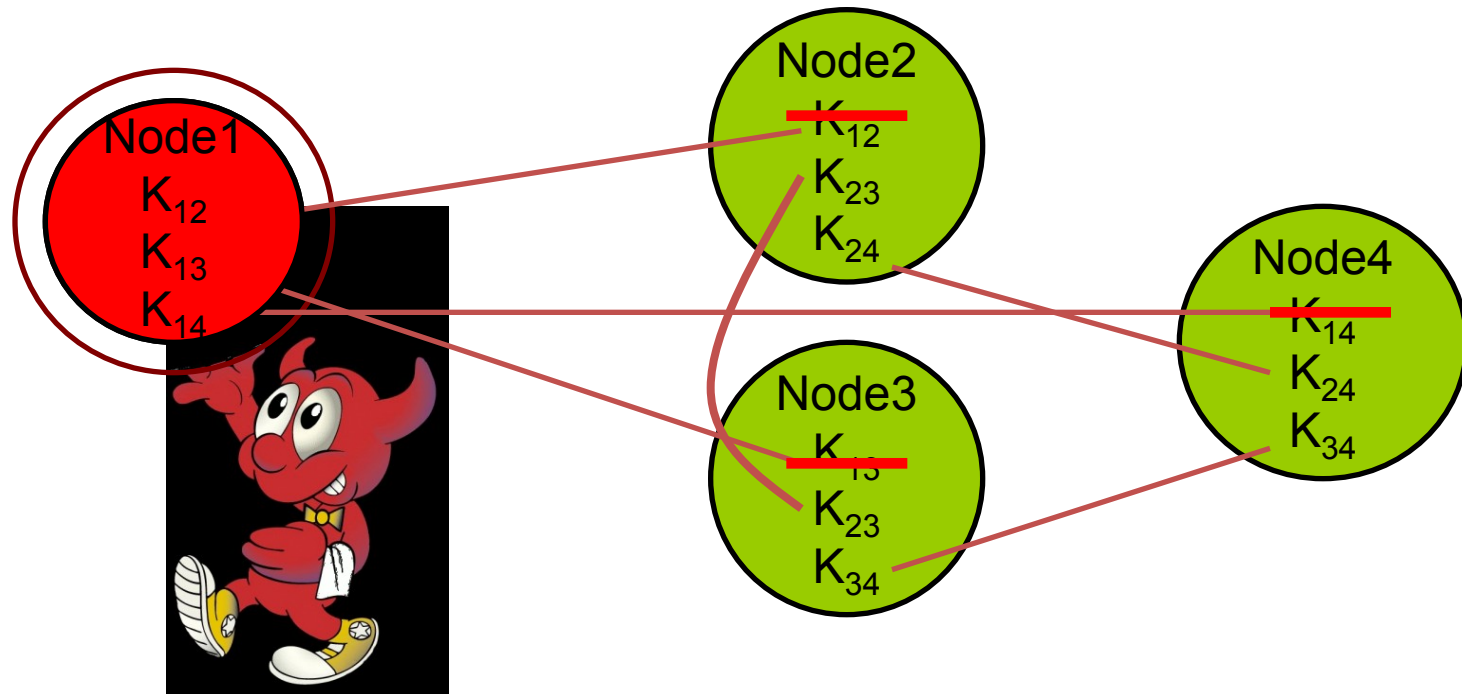
- Predistributed unique key(s) between BS and every node
 - BS holds database of all keys, node holds just single key to BS
- End-to-end encryption/MAC
 - Intermediate nodes just forward towards BS
 - Low latency, memory and computation overhead (no processing on intermediate nodes)
- Possibility for periodic key update
 - $\text{newKey} = \text{KDF}(\text{oldKey}, \text{"Period}_i\text{"})$, erase previous *oldKey*
 - Better than $\text{newKey} = \text{KDF}(\text{masterKey}, \text{"Period}_i\text{"})$ – why?
- Disadvantages of scheme?
 - No data aggregation, insertion of corrupted packets...

Pairwise keys – every node to every node

- Predistributed unique key(s) between every two nodes
 - Every node holds keys to all other potential neighbours
 - (1MB flash storage => 65k of 16B keys)
 - Proper key is found and used when needed
- Unused keys may be erased after neighbour discovery
 - When unused keys will not be necessary
 - No need for a priori knowledge of network layout
- Keys to not yet deployed nodes can be also included
 - Later redeployment of fresh nodes
 - Authentication between old and new nodes possible
- Node capture resiliency
 - no keys except for compromised node are revealed

How “Pairwise keys” pre-distribution fails?

- Only links to captured node are compromised
- Key from captured node can be used everywhere



Level of keys pre-distribution (2)

4. Pairwise keys (node2BS, node2node)

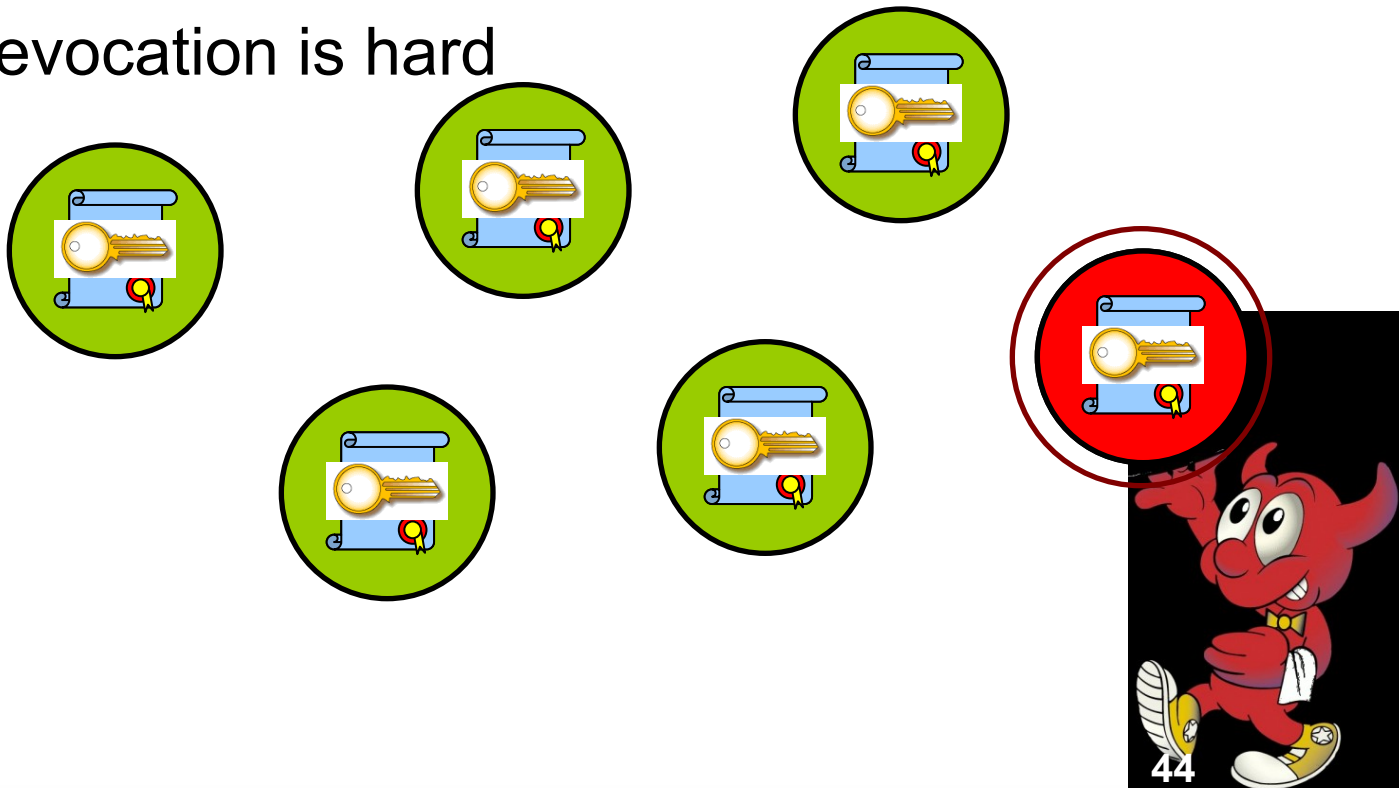
- all nodes can establish keys if necessary

5. Asymmetric cryptography

- all nodes can establish keys if necessary
- e.g., ECC, pairing-based crypto
- shown to be feasible (2.5 sec verification, 20KB ROM)
- problem: revocation of compromised keys/nodes

Why asymmetric cryptography may fail?

- Only links to captured node are compromised
- High computational/transmission overhead ($> 128B$)
- Private key from captured node can be used everywhere
- Revocation is hard



Level of keys pre-distribution (2)

4. Pairwise keys

- all nodes can establish keys if necessary

5. Asymmetric cryptography

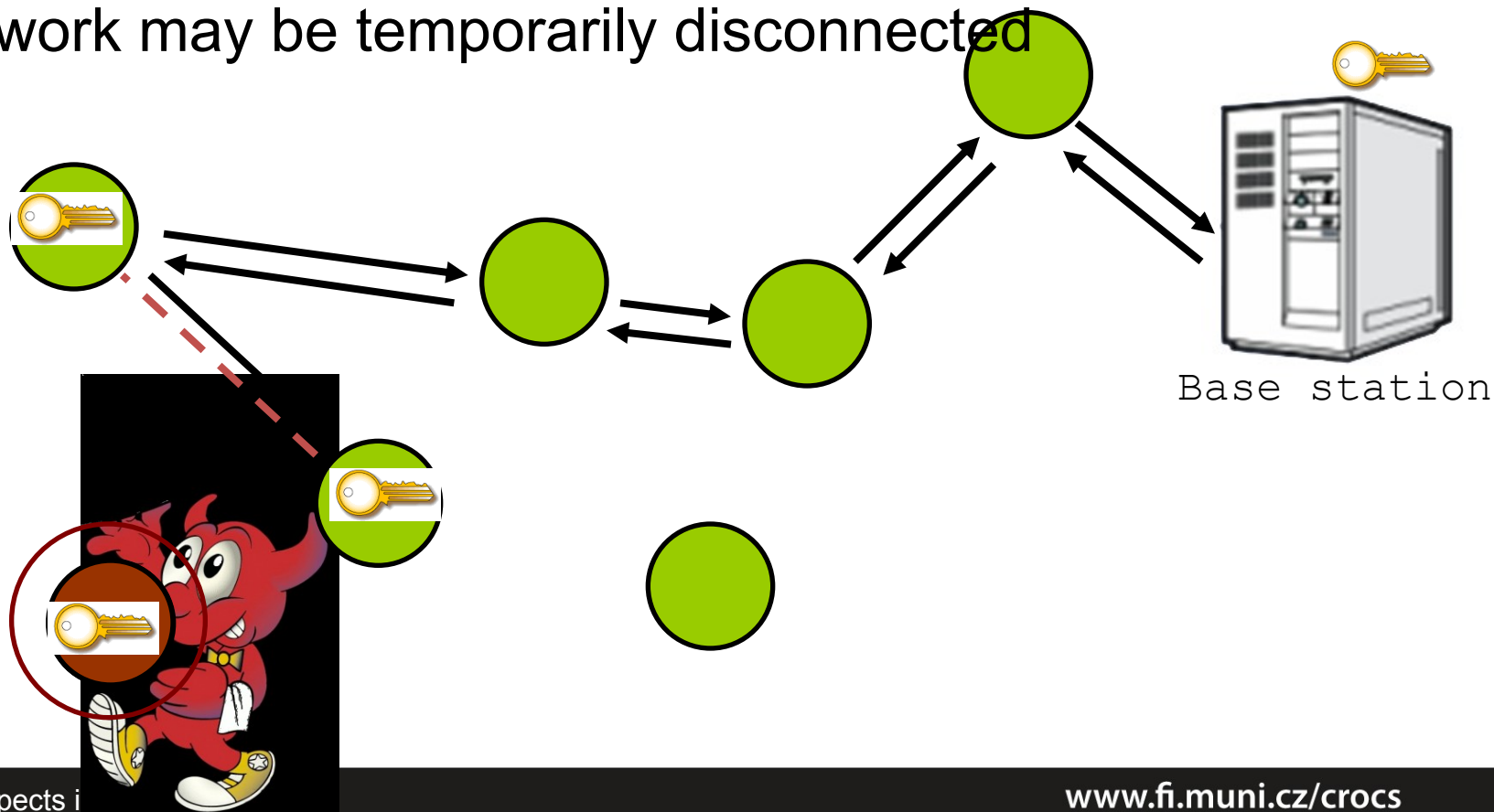
- all nodes can establish keys if necessary
- e.g., ECC, pairing-based crypto
- shown to be feasible (2.5 sec verification, 20KB ROM)
- problem: revocation of compromised keys/nodes

6. Central key distribution (via Base Station)

- BS acts as trusted third party, centralized solution (SPINS)
- problem: multi-hop communication to BS

How TTP distribution fails?

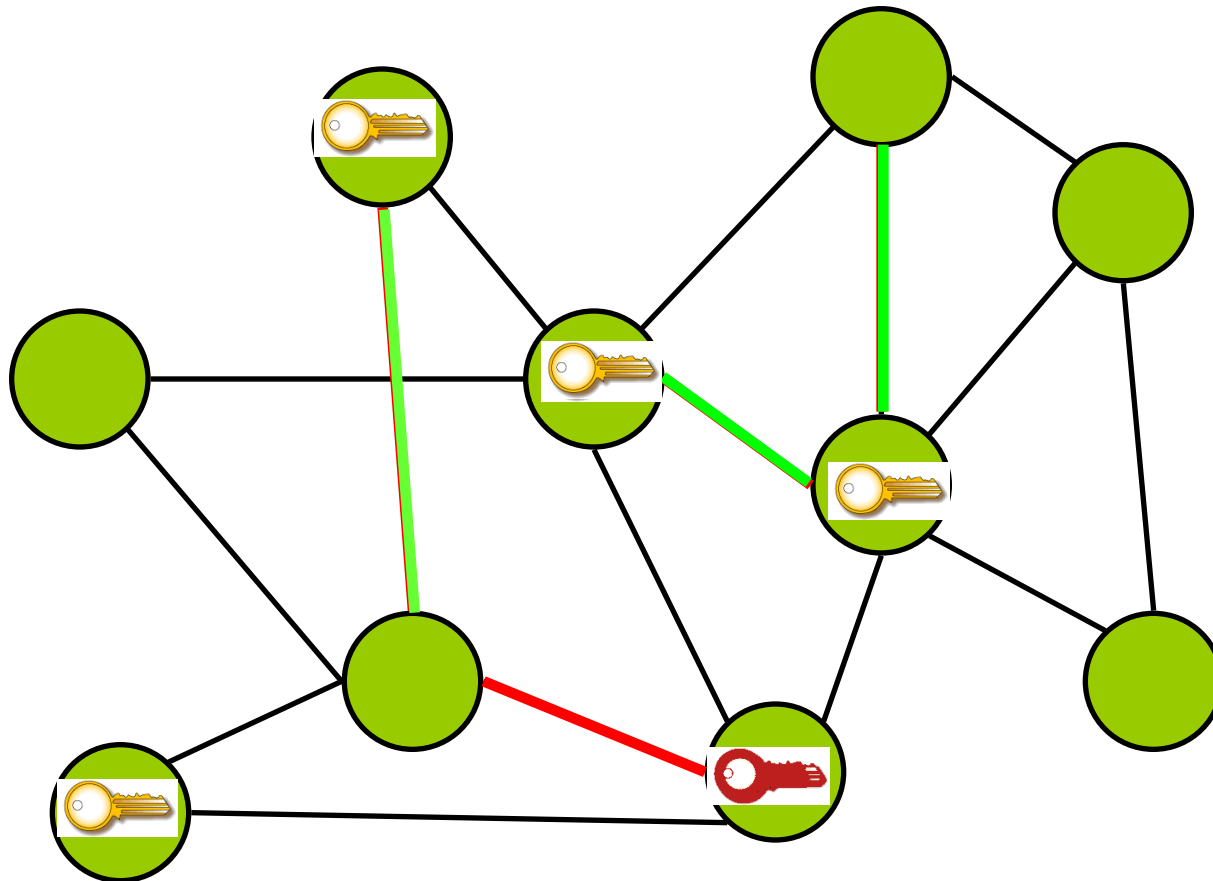
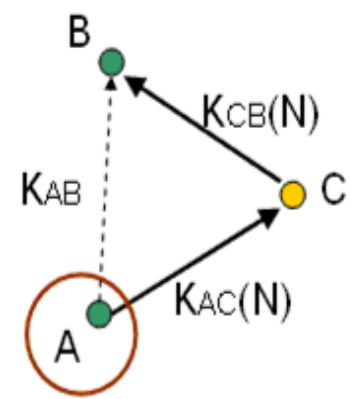
- Every key is established via base station (good control)
- Communication is multi-hop and energy expensive
- Network may be temporarily disconnected



All approaches vulnerable to some extent.
What should we do with partial compromise?

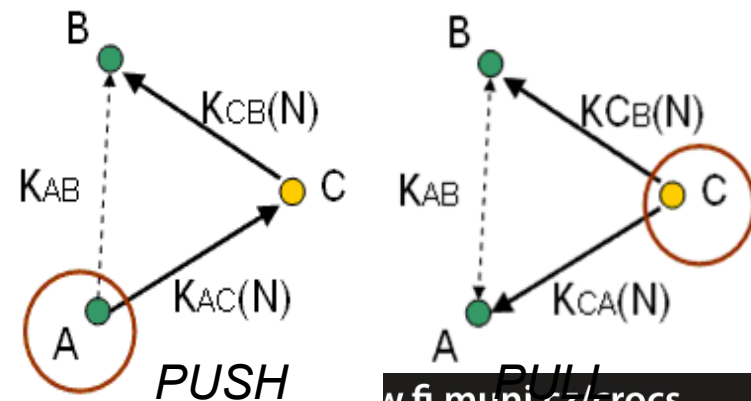
PARTIAL COMPROMISE

Secrecy amplification protocols



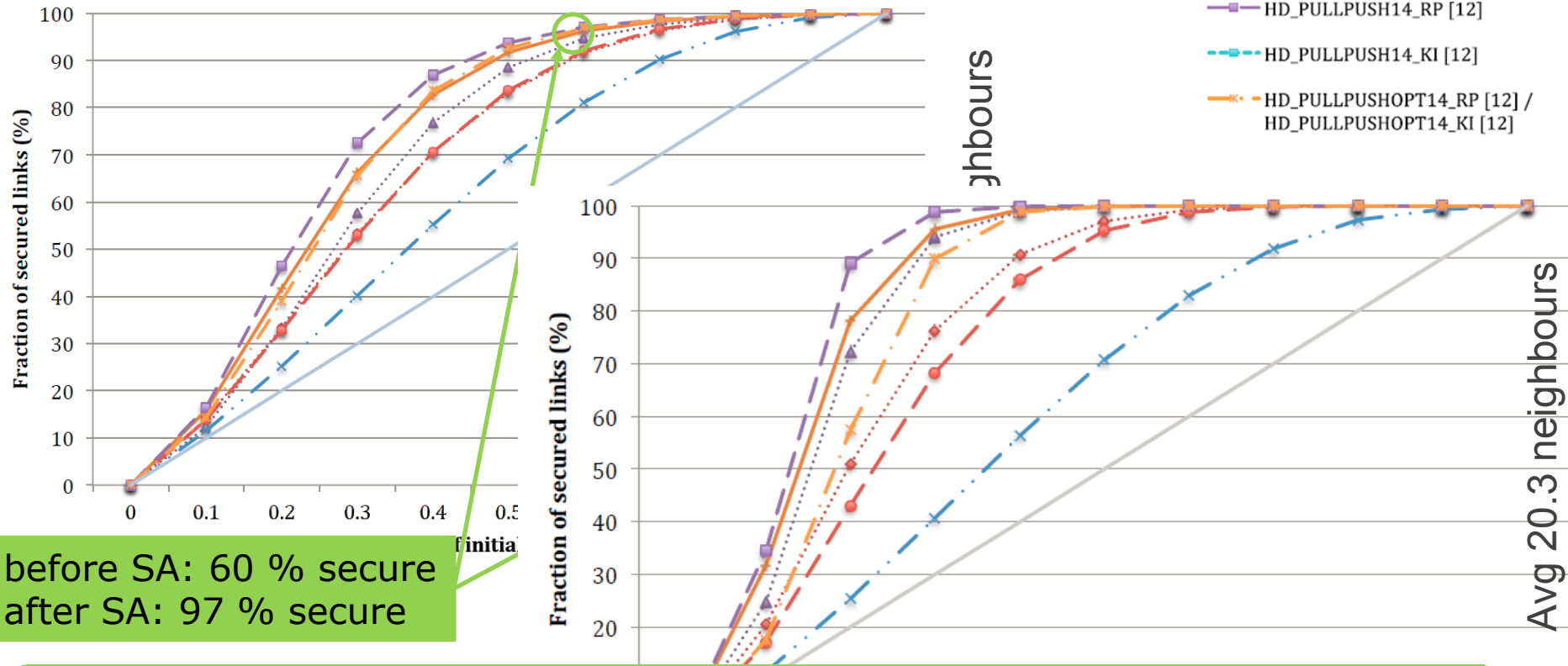
Secrecy amplification protocols

- Additional protocol executed atop of distributed keys
 - network partially compromised after some attack
 - some link keys known to attacker (eavesdropped, captured)
- Secrecy amplification is able to secure previously compromised link(s)
 - transport of fresh link key over secure path
 - success depends on compromise pattern
- Protocol can be executed even when information about compromise is not available
 - old and new key is combined



- NO_3PUSH04 [1] / NO_3PULL05 [4]
- NO_4PUSH04 [1] / NO_4PULL05 [4]
- NO_EA09 [15]
- GO_EA09 [15]
- GO_EA12_RP [14]
- GO_EA12_KI [14]
- HD_PULLPUSH14_RP [12]
- HD_PULLPUSH14_KI [12]
- HD_PULLPUSHOPT14_RP [12] / HD_PULLPUSHOPT14_KI [12]

Comparison: total success rate



before SA: 60 % secure
after SA: 97 % secure

Depending on network density, up to 30 % → 95 %

Main advantages of secrecy amplification

1. Is preventive measure (no detection/reaction)
 2. Can work in (partially) compromised environment
 3. Work with different underlying (pre)distributions
 4. Are introducing secrets (keys) usable only locally
 5. Can be (automatically) parameterized/optimized
 6. Can run continuously – attacker must maintain its presence
- Survey: <http://www.crcs.cz/papers/wistp2015>

Practical implementation – results

- Scenario: 10 neighbours on average
- Hybrid secrecy amplification protocol
- TinyOS 2.1.2 implementation
 - < 500B RAM (peak usage, reusable later), ~3KB code
 - Seconds to minutes to reliably map radio propagation
 - highly depends on surrounding noise, etc.
 - ~1 KB of payload is transmitted during whole secrecy amplification phase (by every node)
 - 1 second worth local computation
 - 1-10 seconds to transmit all amplification messages

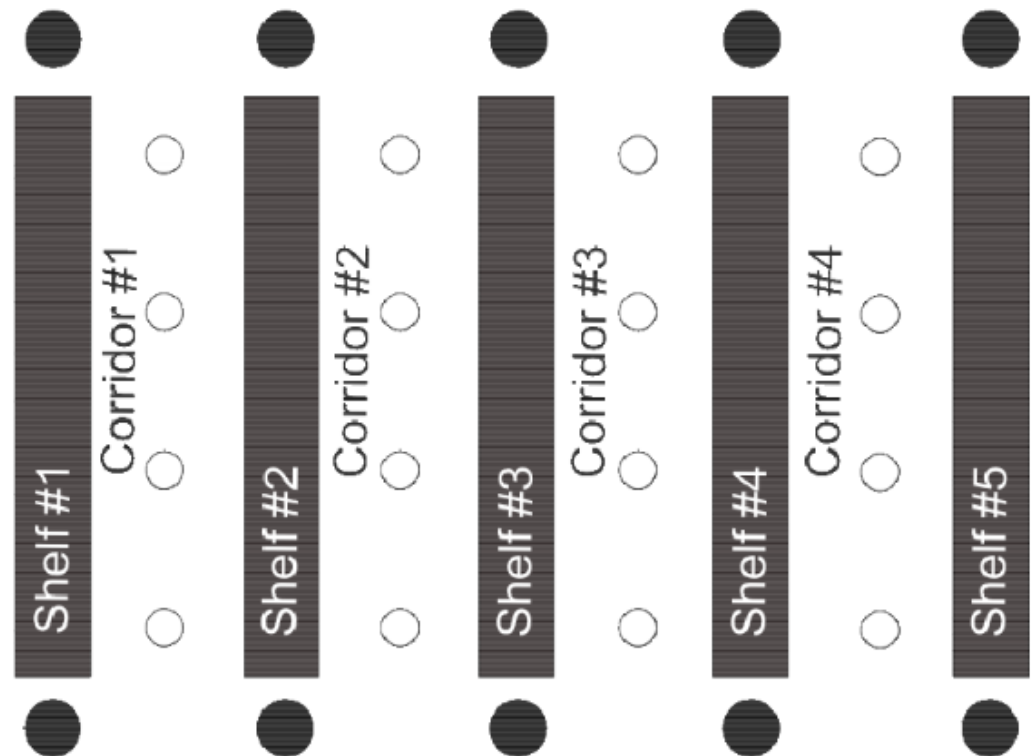


<https://github.com/crocs-muni/WSNProtectLayer>

CASE STUDY: WSNPROTECTLAYER

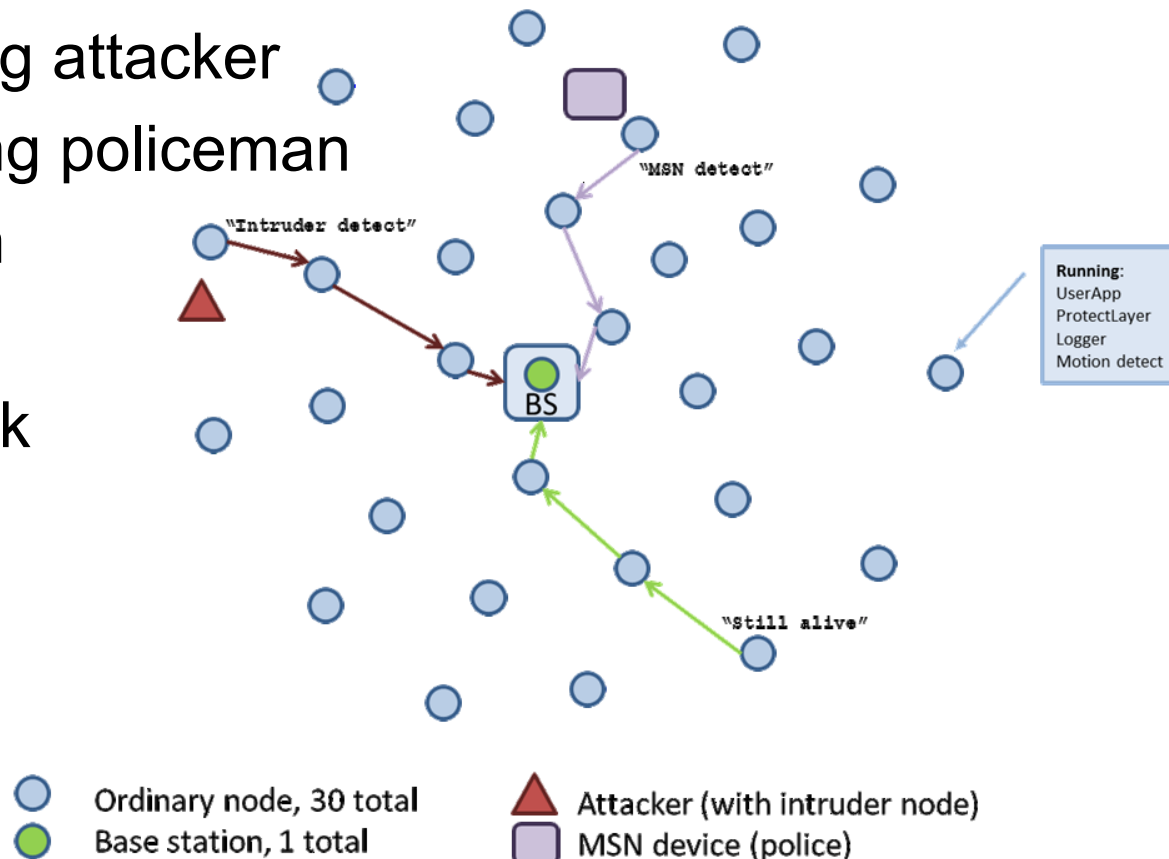
Scenario 1 - Warehouse

- Monitored devices with RFID-based radio tags
- Tracking of person movement
- Static routes
- Long-living network



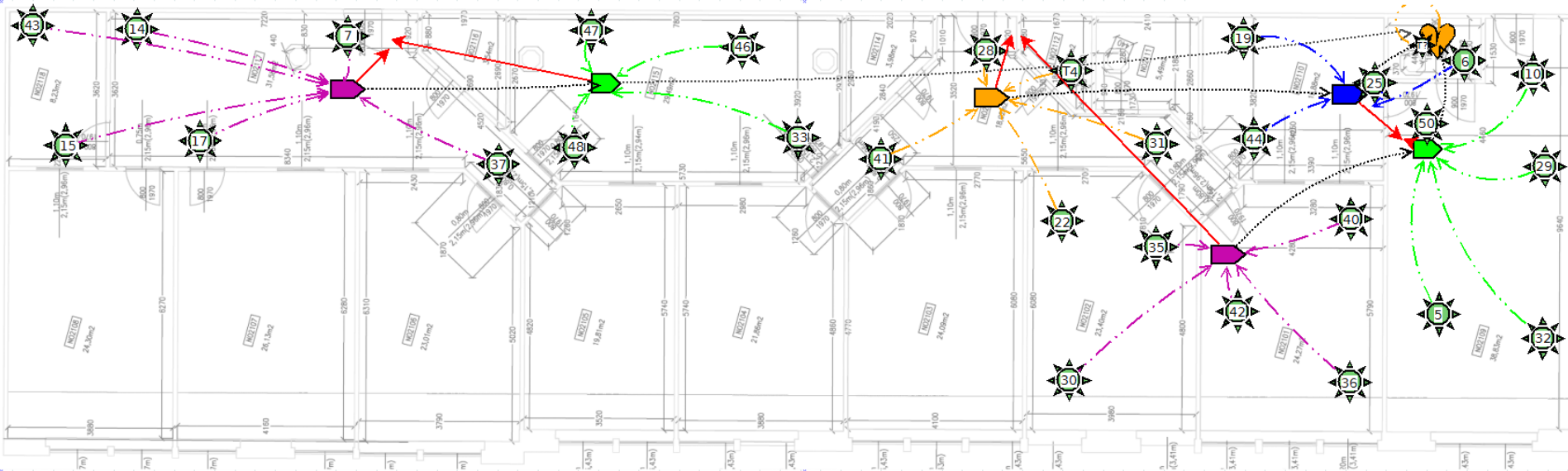
Scenario 2 – Police unit

- Defense of central point (base station)
- Detection of moving attacker
- Reporting of moving policeman
- Jamming detection
- Dynamic routes
- Short-living network



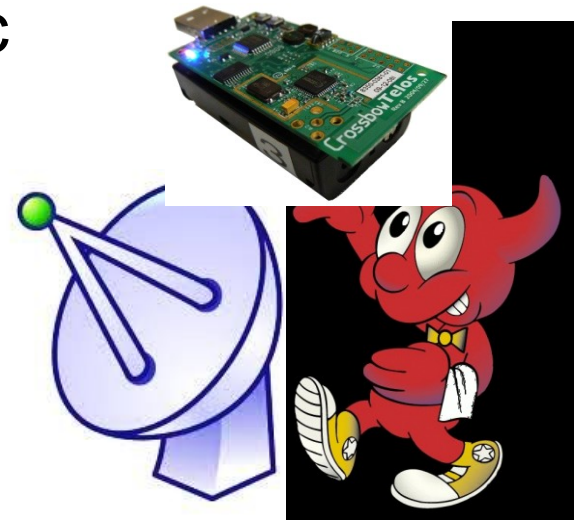
Scenario 3 – Building monitoring

- Tracking of selected person movement
- Multiple levels of privacy protection
- Static routes
- Long-living network



Attacker models assumed

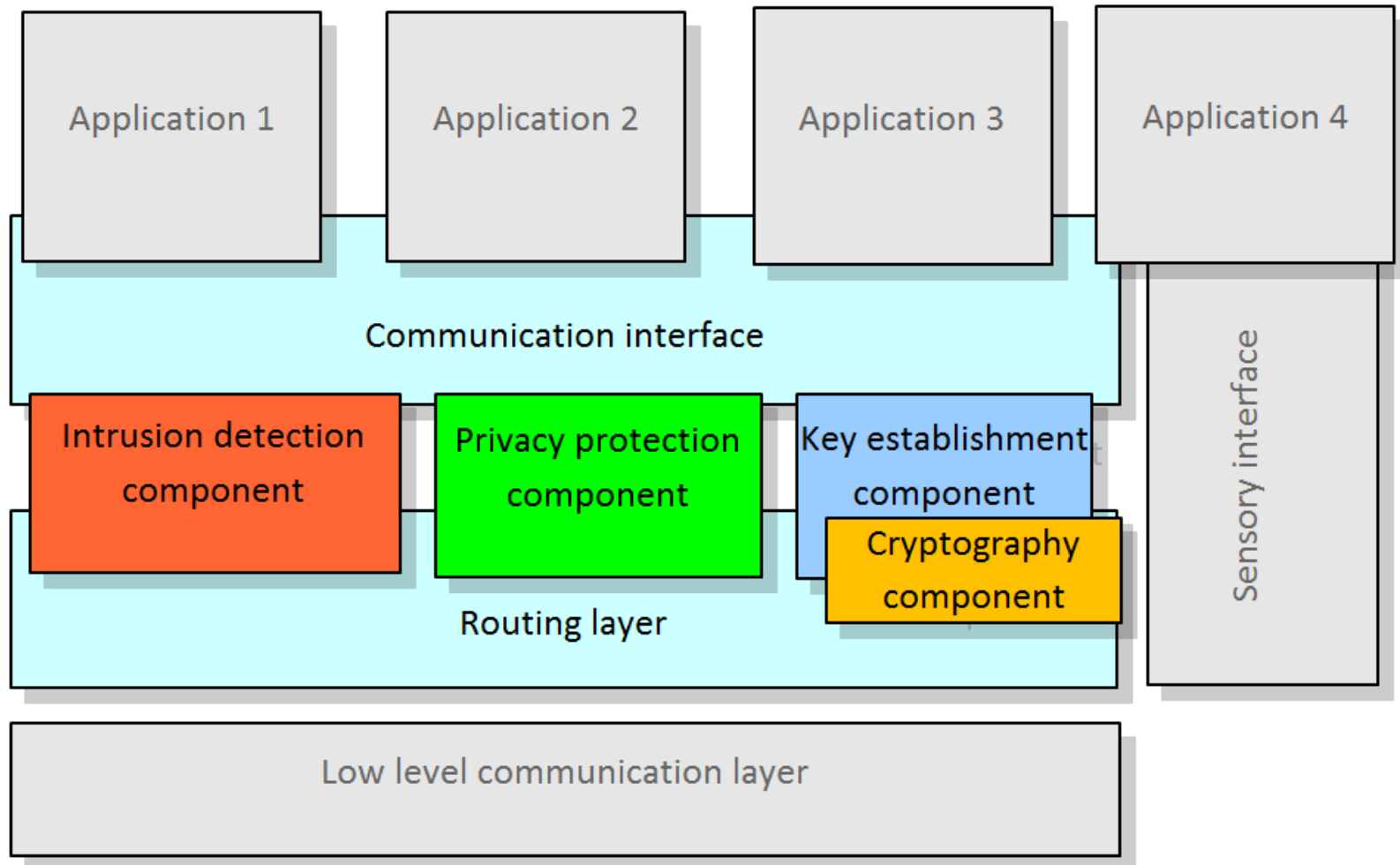
- Local / global passive eavesdropping
 - Packet capture, traffic analysis
- Active attacker manipulating traffic
 - Packet dropping, injection, jamming
- Active attacker capturing nodes
 - And extracting cryptographic keys



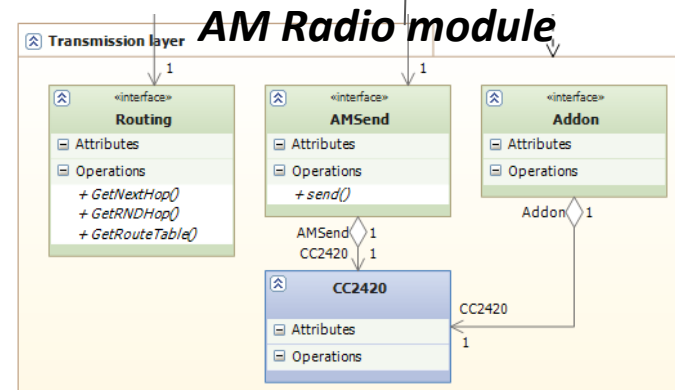
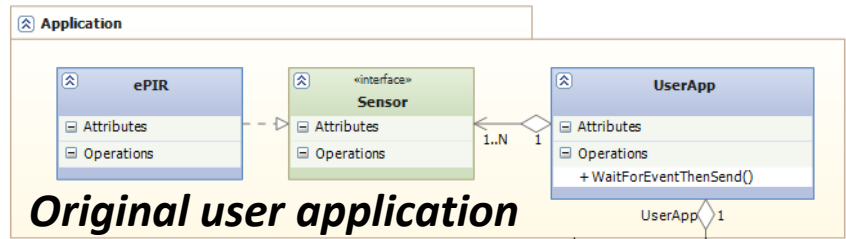
Core architecture components

- **Intrusion detection component**
 - Distributed packet dropper and jamming detection
 - Local neighbour reputation metric
 - Base station notified when misbehaving node is detected
- **Privacy protection component**
 - 4 levels of protection, controlled by authenticated broadcast
 - Open communication
 - Message integrity and authentication
 - Packet encryption
 - Traffic analysis-resistant phantom routing
- **Key management component**
 - Cryptographic key distribution and establishment (node, base stations)
 - Cryptographic services for other components

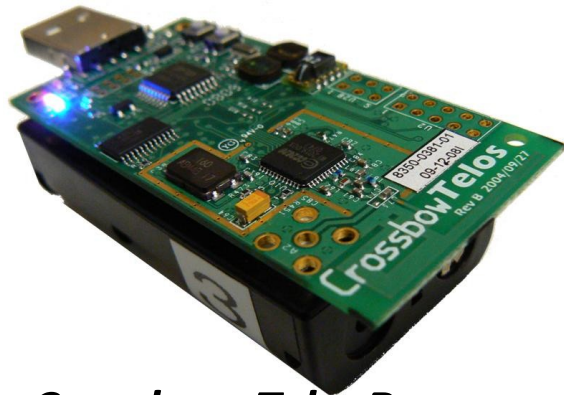




Architecture



Hardware used, testbed

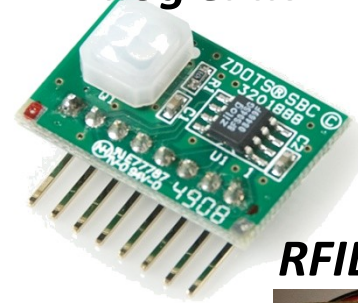


Crossbow TelosB

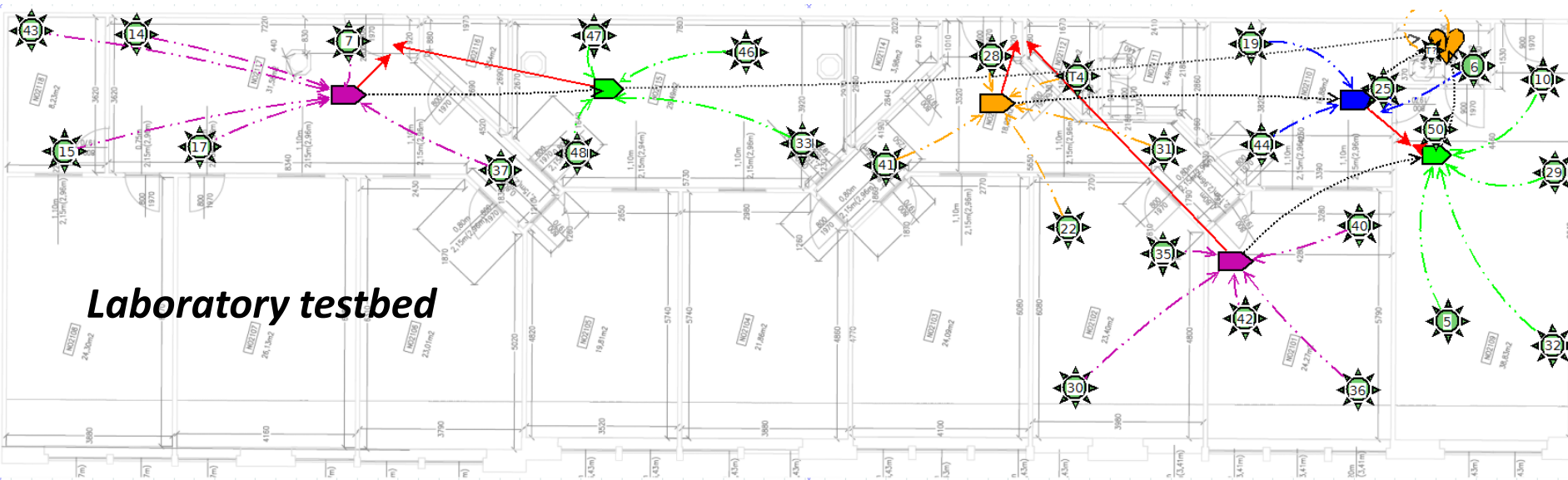


Crossbow MICAz

Zilog ePIR



RFID reader 125kHz



Laboratory testbed

```

configuration BlinkToRadioAppC {
}
implementation {
  components MainC;
  components LedsC;
  components BlinkToRadioC as App;
  components new TimerMilliC() as Timer0;
  components new TimerMilliC() as InitTimer;

```

---> Original Components

```

components ActiveMessageC;
components new AMSenderC(AM_BLINKTORADIO);
components new AMReceiverC(AM_BLINKTORADIO);

```

---> Replaced by new ProtectLayerC

```

// Basic components wiring
App.Boot -> MainC;
App.Leds -> LedsC;
App.Timer0 -> Timer0;
App.InitTimer -> InitTimer;

```

---> Original wirings

```

App.Packet -> AMSenderC;
App.AMPacket -> AMSenderC;
App.AMControl -> ActiveMessageC;
App.AMSend -> AMSenderC;
App.Receive -> AMReceiverC;

```

---> Replaced by new one to ProtectLayerC

```

}

```

```

configuration BlinkToRadioAppC {
}
implementation {
  components MainC;
  components LedsC;
  components BlinkToRadioC as App;
  components new TimerMilliC() as Timer0;
  components new TimerMilliC() as InitTimer;

```

components ProtectLayerC;

```

// Basic components wiring
App.Boot -> MainC;
App.Leds -> LedsC;
App.Timer0 -> Timer0;
App.InitTimer -> InitTimer;

```

```

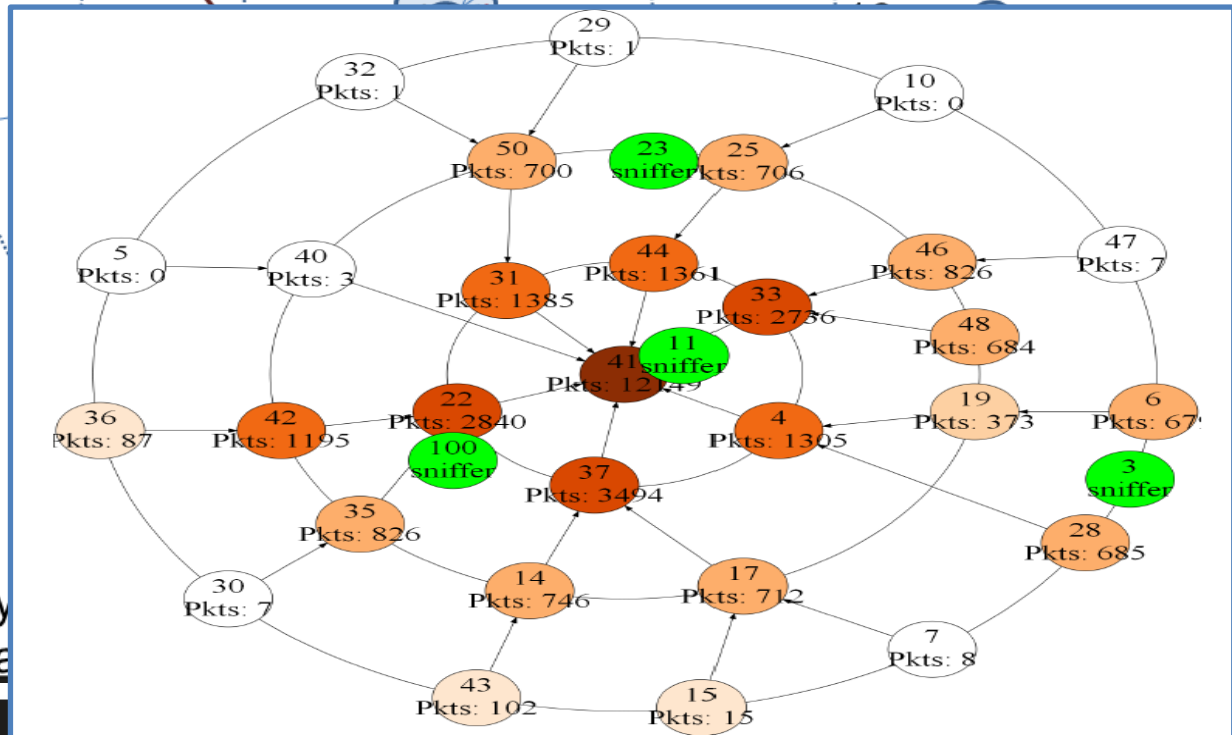
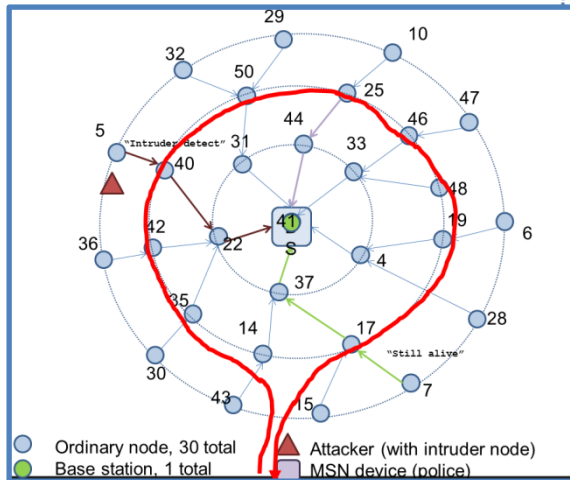
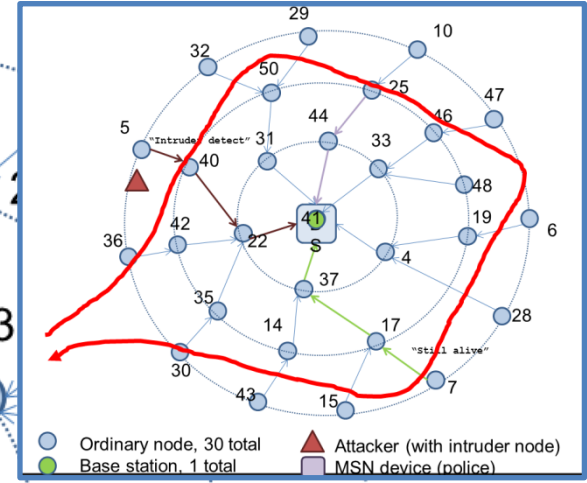
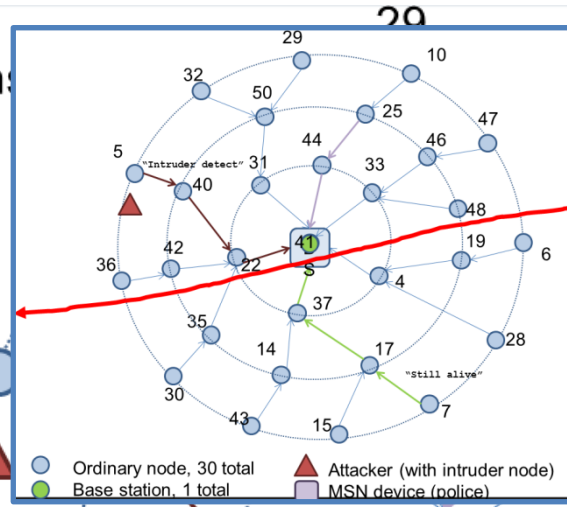
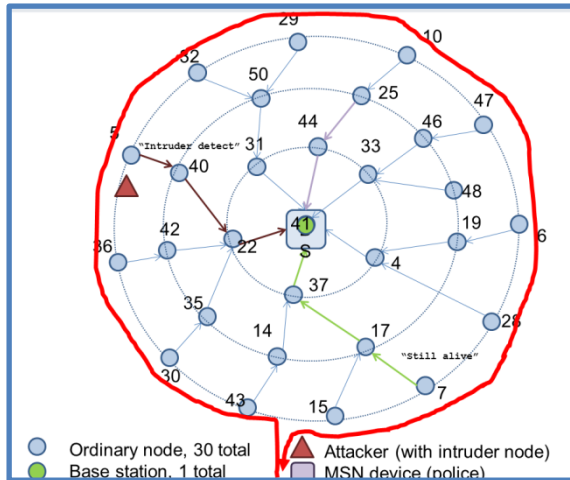
App.Packet -> ProtectLayerC.Packet;
App.AMControl -> ProtectLayerC.AMControl;
App.AMSend -> ProtectLayerC.AMSend;
App.Receive -> ProtectLayerC.Receive;

```

```

}

```



Try it!

- TinyOS 2.x-based (TelosB nodes used)
- *Václav MATYÁŠ, Petr ŠVENDA, Andriy STETSKO, Dušan KLINEC, Filip JURNEČKA a Martin STEHLÍK.*
WSNProtectLayer – security middleware for wireless sensor networks. Securing Cyber-Physical Systems. USA: CRC Press, 2015. s. 119-162, 44 s. CRC Press. ISBN 978-1-4987-0098-6.
- <https://github.com/crocs-muni/WSNProtectLayer>

Summary

- Common security protocols often cannot be used
 - Preference for symmetric crypto-only solutions
 - Low transmission overhead important due to energy
- Key distribution is (as usual) critical factor
- Partial compromise should be anticipated
 - And protocols designed to be able to cope with it
- Mandatory reading
 - A. Perrig et al: SPINS: Security Protocols for Sensor Networks
 - <https://users.ece.cmu.edu/~adrian/projects/mc2001/mc2001.pdf>