Security and Cooperation in Wireless Networks

http://secowinet.epfl.ch/

Security and Cooperation in Wireless Networks

a tutorial presented at Performance 2007, Cologne, Oct 2, 2007.

Outline

- New wireless networks and new challenges (25')
- Thwarting malicious behavior
  - introduction to cryptography and security techniques (30')
  - naming and addressing (20')
  - secure routing (30')
- Thwarting selfish behavior
  - introduction to game theory (30')
  - selfishness in packet forwarding (20')
  - border games in cellular networks (20')

Security and Cooperation in Wireless Networks

http://secowinet.epfl.ch/

New wireless networks and challenges

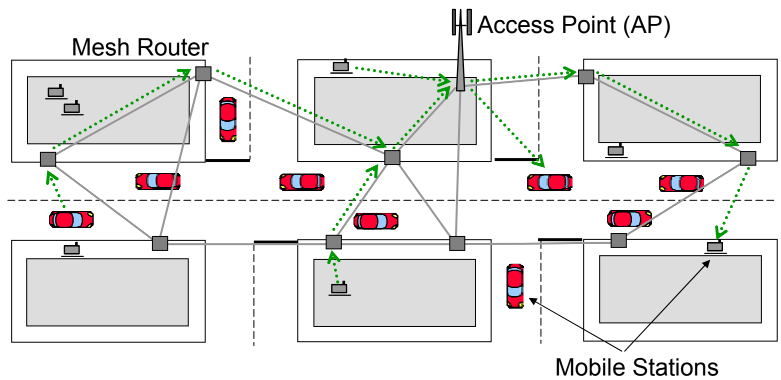
new wireless networks;
new challenges;
the issue of trust;

Upcoming wireless networks

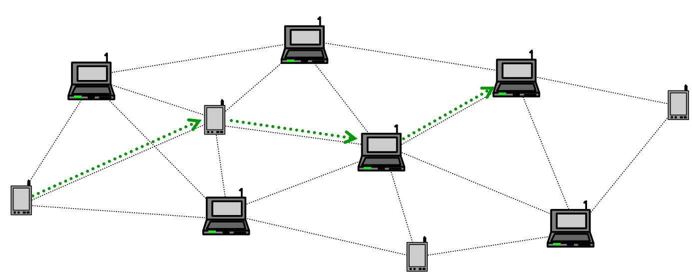
- everything beyond current wireless networks (3G and WiFi)
- examples:
  - wireless mesh networks (operator or community based)
  - infrastructureless ad hoc networks
  - vehicular communication systems
  - wireless sensor networks
  - RFID/NFC systems
  - personal area networks
  - body area networks
  - …

Wireless mesh networks

- mesh technology can be used to extend the coverage of wireless hot spots in a sizeable geographical area
  - Internet connectivity is provided to a larger population at a lower cost
- based on transit access points (mesh routers) and multi-hop wireless communications



Infrastructureless ad hoc networks



- infrastructureless operation = merging terminal and router functions
- nodes are potentially mobile
- application areas:
  - battlefield communications (and rescue operations)
  - free-of-charge personal communications
  - wireless embedded system (body area networks, networks of houshold appliances, vehicular ad hoc networks, ...)
- similar trend at the application layer is called peer-to-peer computing

Vehicular communications – motivation

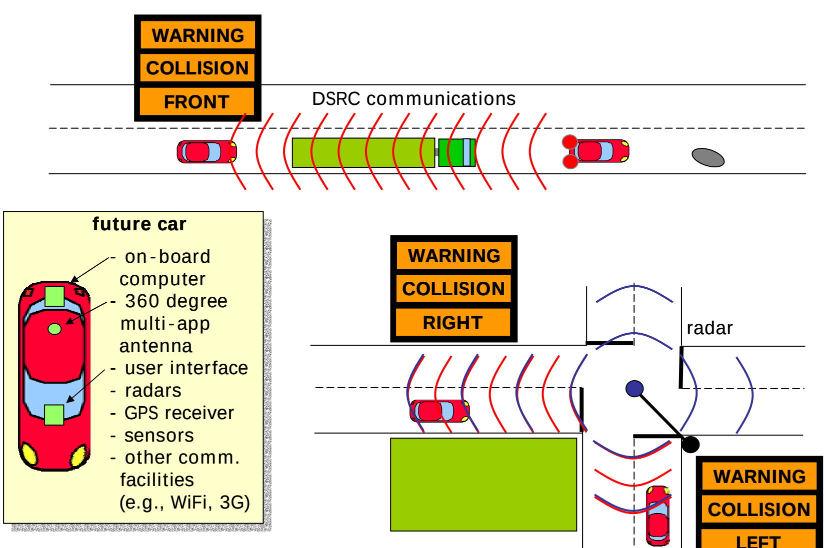
- side effects of road traffic

40000 people die and 1.5 million are injured every year in the EU

traffic jams generate a tremendous waste of time and fuel

- most of these problems could be solved by providing appropriate information to the driver or to the vehicle

Vehicular communications – examples (C2C and I2C)

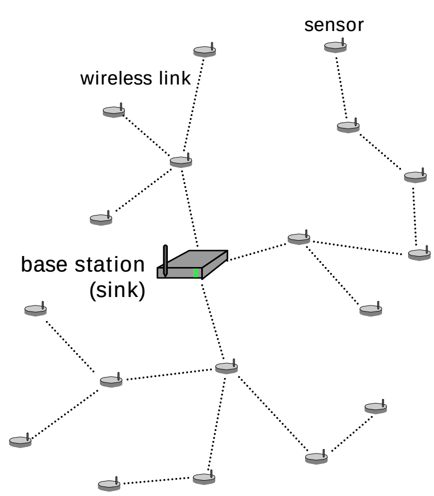


Envisioned VC applications for public safety

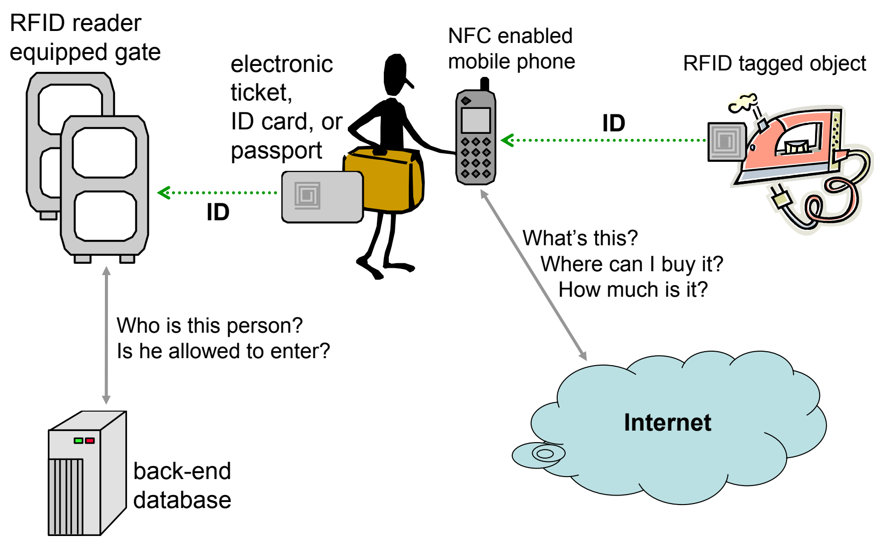
- APPROACHING EMERGENCY VEHICLE (WARNING) ASSISTANT (3)
- EMERGENCY VEHICLE SIGNAL PREEMPTION
- ROAD CONDITION WARNING
- LOW BRIDGE WARNING
- WORK ZONE WARNING
- IMMINENT COLLISION WARNING (D)
- CURVE SPEED ASSISTANCE [ROLLOVER WARNING] (1)
- INFRASTRUCTURE BASED – STOP LIGHT ASSISTANT (2)
- INTERSECTION COLLISION WARNING/AVOIDANCE (4)
- HIGHWAY/RAIL [RAILROAD] COLLISION AVOIDANCE (10)
- COOPERATIVE COLLISION WARNING [V-V] (5)
- GREEN LIGHT - OPTIMAL SPEED ADVISORY (8)
- COOPERATIVE VEHICLE SYSTEM – PLATOONING (9)
- COOPERATIVE ADAPTIVE CRUISE CONTROL [ACC] (11)
- VEHICLE BASED PROBE DATA COLLECTION (B)
- INFRASTRUCTURE BASED PROBE DATA COLLECTION
- INFRASTRUCTURE BASED TRAFFIC MANAGEMENT – [DATA COLLECTED from] PROBES (7)
- TOLL COLLECTION
- TRAFFIC INFORMATION (C)
- TRANSIT VEHICLE DATA TRANSFER (gate)
- TRANSIT VEHICLE SIGNAL PRIORITY
- EMERGENCY VEHICLE VIDEO RELAY
- MAINLINE SCREENING
- BORDER CLEARANCE
- ON-BOARD SAFETY DATA TRANSFER
- VEHICLE SAFETY INSPECTION
- DRIVER'S DAILY LOG

Wireless sensor networks

- environmental monitoring (for ecological and/or agricultural purposes)
- monitoring the state of structures (e.g., bridges, tunnels, …)
- remote patient monitoring (elderly and chronically ill people)
- industrial process automation
- building automation
- …
- military applications



RFID/NFC systems



Challenges for providing security

- multi-hop wireless communications
  - why?
    - reduce interference
    - reduce energy consumption
    - save on infrastructure deployment
  - consequences
    - terminals play the role of network nodes (routers)
    - where's the edge of the network?

- lack of physical protection
  - why?
    - unattended operation
    - no tamper resistance (it would cost a lot)
  - consequences
    - easy access to devices
    - nodes may be compromised

Hacking your Prius [CNET News.com]

More challenges (1/2)

- scale
  - thousands or millions of nodes (e.g., Smart Dust)
  - network is not necessarily hierarchically organized
  - or hierarchy is built on-the-fly
- mobility
  - dynamically changing topology
  - intermittent connectivity
  - transient relationships
- self-organization
  - infrastructureless operation
  - decentralization

More challenges (2/2)

- increased programmability of devices
  - easy to install new applications
  - basic operation of the device can be modified (e.g., software defined radio)
- resource constraints
  - tiny, embedded devices, running on batteries
  - no support for heavy cryptographic algorithms
  - energy consumption is an issue
- embedded systems
  - many nodes are not directly operated by humans
  - decisions must be made autonomously
- increased privacy risks
  - many wireless devices are carried by people or embedded in vehicles
  - easy tracking of whereabouts of individuals

Trust

- the trust model of current wireless networks is rather simple
  - subscriber – service provider model
  - subscribers trusts the service provider for providing the service, charging correctly, and not misusing transactional data
  - service providers usually do not trust subscribers, and use security measures to prevent or detect fraud
- in the upcoming wireless networks the trust model will be much more complex
  - entities play multiple roles (users can become service providers)
  - number of service providers will dramatically increase
  - user – service provider relationships will become transient
  - how to build up trust in such a volatile and dynamic environment?
- yet, trust is absolutely fundamental for the future of wireless networks
  - pervasiveness of these technologies means that all of us must rely on them in our everyday life!

Trust vs. security and cooperation

- trust preexists security
  - all security mechanisms require some level of trust in various components of the system
  - security mechanisms can help to *transfer* trust in one component to trust in another component, but they cannot create trust by themselves
- cooperation reinforces trust
  - trust is about the ability to *predict* the behavior of another party
  - cooperation (i.e., adherence to certain rules for the benefit of the entire system) makes predictions more reliable

Reasons to trust

- moral values
  - will be difficult to observe compliance with them
- experience about another party
  - relationships may not last long enough for this
- rule enforcement organizations
  - need to rely more on rule enforcement mechanisms
- **rule enforcement mechanisms**
  - prevent bad things from happening → security techniques
  - encourage desirable behavior → game theory and mechanism design

Malice and selfishness

- malice
  - willingness to do harm no matter what
- selfishness
  - overuse of common resources (network, radio spectrum, etc.) for one's own benefit

- traditionally, security is concerned only with malice
- but in the future, **malice and selfishness must be considered jointly** if we want to seriously protect wireless networks

Outline

- New wireless networks and new challenges (25')

- Thwarting malicious behavior
  - introduction to cryptography and security techniques (30')
  - naming and addressing (20')
  - secure routing (30')

- Thwarting selfish behavior
  - introduction to game theory (30')
  - selfishness in packet forwarding (20')
  - border games in cellular networks (20')

Security and Cooperation in Wireless Networks

http://secowinet.epfl.ch/

Introduction to cryptography and security techniques

symmetric and asymmetric key encryption;

hash functions;

MAC functions;

digital signatures;

key establishment protocols;

Introduction

- security is about how to prevent attacks, or -- if prevention is not possible -- how to detect attacks and recover from them
- an attack is a a *deliberate attempt* to compromise a system; it usually exploits weaknesses in the system's design, implementation, operation, or management
- attacks can be
  - passive
    - attempts to learn or make use of information from the system but does not affect system resources
    - examples: eavesdropping message contents, traffic analysis
    - difficult to detect, should be prevented
  - active
    - attempts to alter system resources or affect their operation
    - examples: masquerade (spoofing), replay, modification (substitution, insertion, destruction), denial of service
    - difficult to prevent, should be detected

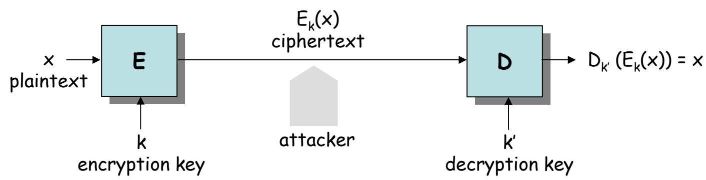
Main security services

- authentication
  - aims to detect masquerade
  - provides assurance that a communicating entity is the one that it claims to be
- access control
  - aims to prevent unauthorized access to resources
- confidentiality
  - aims to protect data from unauthorized disclosure
  - usually based on encryption
- integrity
  - aims to detect modification and replay
  - provides assurance that data received are exactly as sent by the sender
- non-repudiation
  - provides protection against denial by one entity involved in a communication of having participated in all or part of the communication
  - two basic types: non-repudiation of origin and non-repudiation of delivery

Some security mechanisms

- encryption
  - symmetric key, asymmetric (public) key
- digital signature
- access control schemes
  - access control lists, capabilities, security labels, ...
- data integrity mechanisms
  - message authentication codes, sequence numbering, time stamping, cryptographic chaining
- authentication protocols
  - passwords, cryptographic challenge-response protocols, biometrics
- traffic padding
- routing control
  - selection of physically secure routes

Operational model of encryption



- attacker's goal:
  - to systematically recover plaintext from ciphertext
  - to deduce the (decryption) key
- Kerckhoff's assumption:
  - attacker knows all details of E and D
  - attacker doesn't know the (decryption) key

Attack models

- ciphertext-only attack
  - the adversary can only observe ciphertexts produced by the same encryption key
- known-plaintext attack
  - the adversary can obtain corresponding plaintext-ciphertext pairs produced with the same encryption key
- (adaptive) chosen-plaintext attack
  - the adversary can choose plaintexts and obtain the corresponding ciphertexts
- (adaptive) chosen-ciphertext attack
  - the adversary can choose ciphertexts and obtain the corresponding plaintexts
- related-key attack
  - the adversary can obtain ciphertexts, or plaintext-ciphertext pairs that are produced with different encryption keys that are related in a known way to a specific encryption key

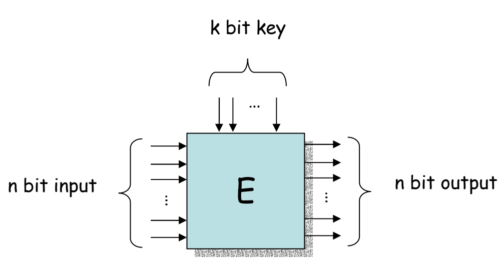
Asymmetric- vs. symmetric-key encryption

- symmetric-key encryption
  - it is easy to compute K' from K (and vice versa)
  - usually K' = K
  - two main types:
    - stream ciphers – operate on individual characters of the plaintext
    - block ciphers – process the plaintext in larger blocks of characters
- asymmetric-key encryption
  - it is hard (computationally infeasible) to compute K' from K
  - K can be made public (→ public-key cryptography)

Block ciphers

an n bit block cipher is a function $E: \{0, 1\}^n \times \{0, 1\}^k \rightarrow \{0, 1\}^n$, such that for each $K \in \{0, 1\}^k$, $E(., K) = E_K : \{0, 1\}^n \rightarrow \{0, 1\}^n$ is a **strong pseudorandom permutation**

(i.e., practically indistinguishable from a randomly chosen permutation even if the adversary is given oracle access to the inverse of the permutation)



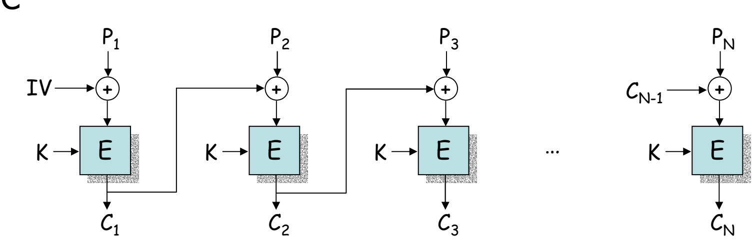
Examples: DES, AES

Block cipher modes of operation

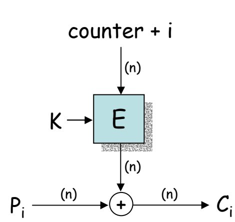
- ECB – Electronic Codebook
  - used to encipher a single plaintext block (e.g., a DES key)
- CBC – Cipher Block Chaining
  - repeated use of the encryption algorithm to encipher a message consisting of many blocks
- CFB – Cipher Feedback
  - used to encipher a stream of characters, dealing with each character as it comes
- OFB – Output Feedback
  - another method of stream encryption, used on noisy channels
- CTR – Counter
  - simplified OFB with certain advantages

Frequently used modes

- CBC



- CTR

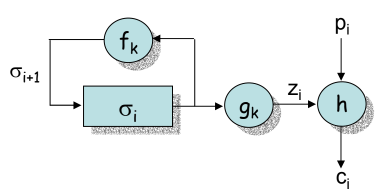


Stream ciphers

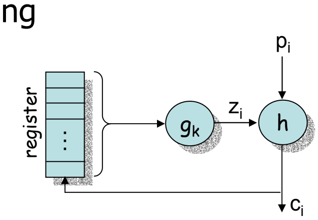
- while block ciphers simultaneously encrypt groups of characters, stream ciphers encrypt individual characters
  - may be better suited for real time applications
- stream ciphers are usually faster than block ciphers in hardware (but not necessarily in software)
- limited or no error propagation
  - may be advantageous when transmission errors are probable
- <u>note</u>: the distinction between stream ciphers and block ciphers is not definitive
  - stream ciphers can be built out of block ciphers using CFB, OFB, or CTR modes
  - a block cipher in ECB or CBC mode can be viewed as a stream cipher that operates on large characters

Types of stream ciphers

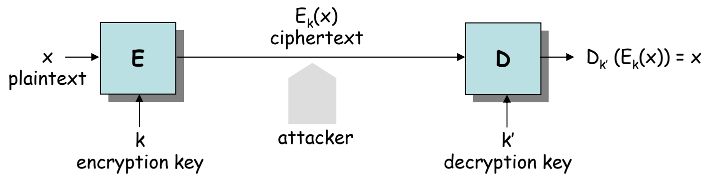
- synchronous



- self-synchronizing



Public-key cryptography



- asymmetric-key encryption
  - it is hard (computationally infeasible) to compute K' from K
- K can be made public (public-key cryptography)
  - no need for key setup before communication
- public-keys are not confidential but they must be authentic !
- the security of asymmetric-key encryption schemes is usually based on some well-known or widely believed hard problems

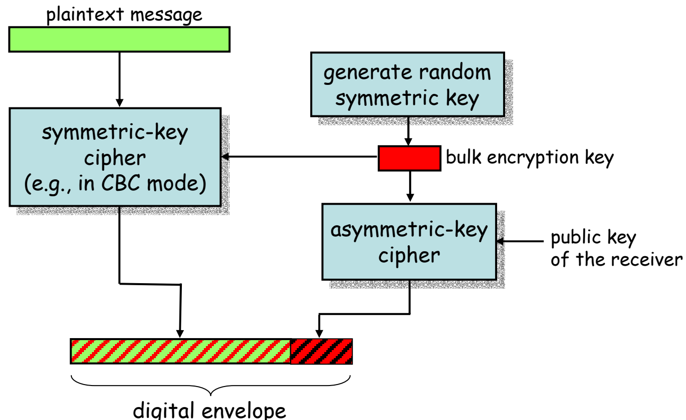
Examples of hard problems

- factoring problem (related cryptosystem: RSA)
  - given a positive integer n, find its prime factors
    - true complexity is unknown
    - it is believed that it does not belong to P
- discrete logarithm problem (related cryptosystem: ElGamal)
  - given a prime p, a generator g of Z_p^*, and an element y in Z_p^*, find the integer x, $0 \leq x \leq p-2$, such that $g^x \bmod p = y$
    - true complexity is unknown
    - it is believed that it does not belong to P

Digital enveloping

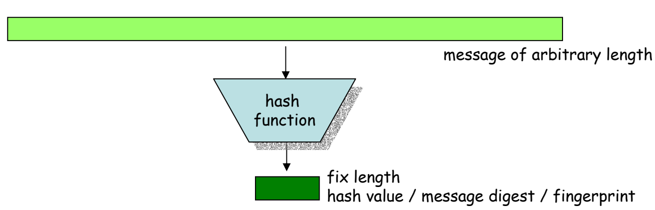
- most popular public-key encryption methods are several orders of magnitude slower than the best known symmetric key schemes

→ public-key encryption is used together with symmetric-key encryption; the technique is called digital enveloping



Hash functions

- a hash function maps bit strings of arbitrary finite length to bit strings of fixed length (n bits)
- many-to-one mapping → collisions are unavoidable
- however, finding collisions are difficult → the hash value of a message can serve as a compact representative image of the message (similar to fingerprints)

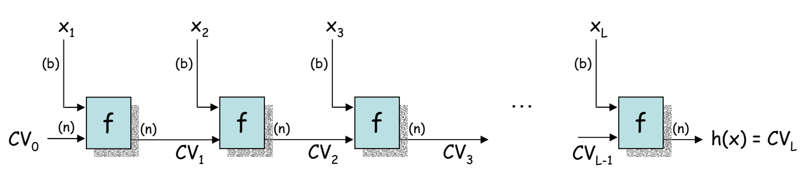


Desirable properties of hash functions

- ease of computation
  - given an input x, the hash value h(x) of x is easy to compute
- weak collision resistance (2nd preimage resistance)
  - given an input x, it is computationally infeasible to find a second input x' such that h(x') = h(x)
- strong collision resistance (collision resistance)
  - it is computationally infeasible to find any two distinct inputs x and x' such that h(x) = h(x')
- one-way hash function (preimage resistance)
  - given a hash value y (for which no preimage is known), it is computationally infeasible to find any input x s.t. h(x) = y

Iterated hash functions

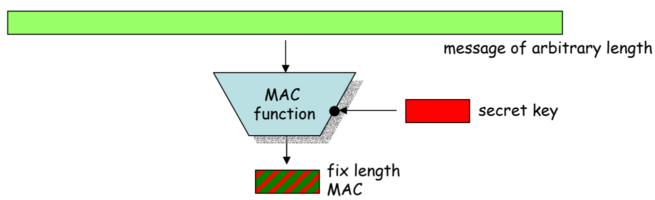
- input is divided into fixed length blocks $x_1, x_2, \ldots, x_L$
- last block is padded if necessary
  - Merkle-Damgard strengthening: padding contains the length of the message
- each input block is processed according to the following scheme



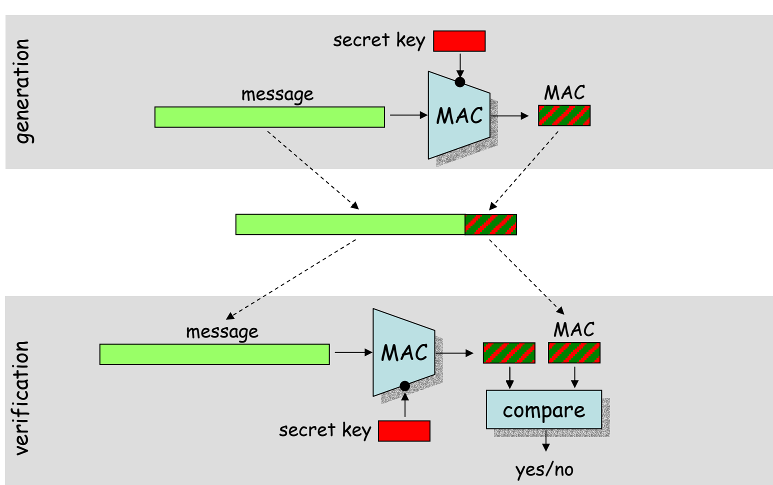
- f is called the compression function
  - can be based on a block cipher, or
  - can be a dedicated compression function
- examples: MD5, SHA1

Message authentication codes (MACs)

- MAC functions can be viewed as hash functions with two functionally distinct inputs: a message and a secret key
- they produce a fixed size output (say n bits) called the MAC
- practically it should be infeasible to produce a correct MAC for a message without the knowledge of the secret key
- MAC functions can be used to implement data integrity and message origin authentication services



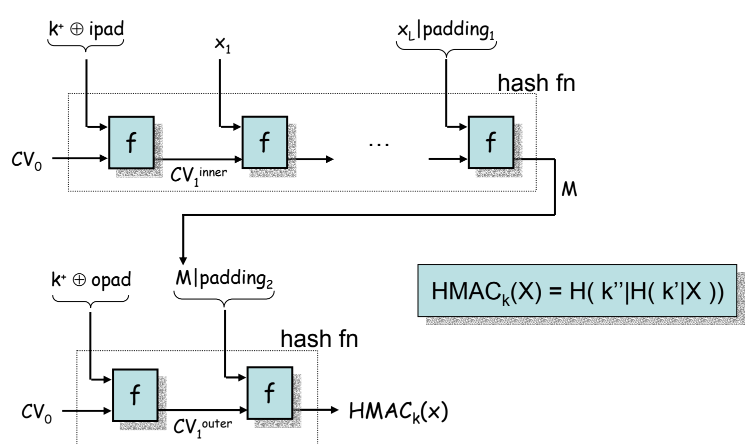
MAC generation and verification



Desirable properties of MAC functions

- ease of computation
  - given an input x and a secret key k, it is easy to compute $MAC_k(x)$
- key non-recovery
  - it is computationally infeasible to recover the secret key k, given one or more text-MAC pairs $(x_i, MAC_k(x_i))$ for that k
- computation resistance
  - given zero or more text-MAC pairs $(x_i, MAC_k(x_i))$, it is computationally infeasible to find a text-MAC pair $(x, MAC_k(x))$ for any new input $x \neq x_i$
  - computation resistance implies key non-recovery but the reverse is not true in general

HMAC

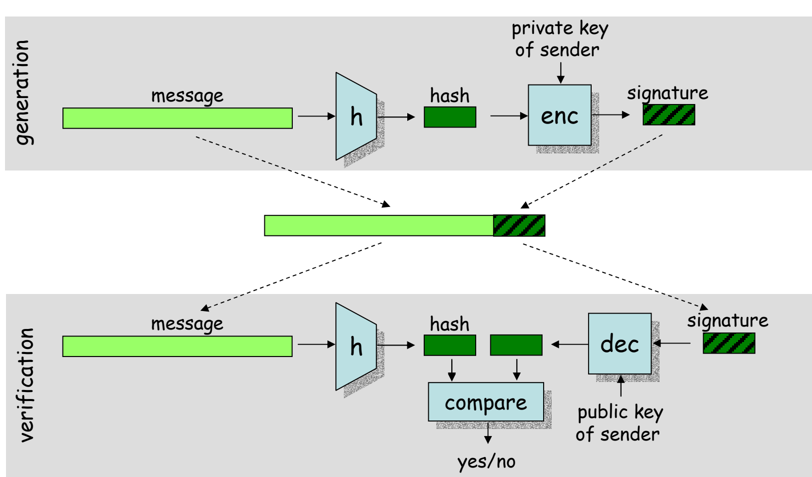


Digital signatures

- similar to MACs but
  - unforgeable by the receiver
  - verifiable by a third party
- used for message authentication and non-repudiation (of message origin)
- based on public-key cryptography
  - private key defines a signing transformation S_A
    - $S_A(m) = \sigma$
  - public key defines a verification transformation V_A
    - $V_A(m, \sigma)$ = true if $S_A(m) = \sigma$
    - $V_A(m, \sigma)$ = false otherwise

"Hash-and-sign" paradigm

- public/private key operations are slow
- hash the message first and apply public/private key operations to the hash value only



Examples of digital signature scheme

- RSA
  - essentially identical to the RSA encryption scheme
  - signature = decryption with private key
  - typical signature length is 1024 bits
- DSA (Digital Signature Algorithm)
  - based on the ElGamal signature scheme
  - typical signature length is 1024 bits
- ECDSA (Elliptic Curve DSA)
  - same as DSA but works over elliptic curves
  - reduced signature length (typically 320 bits)

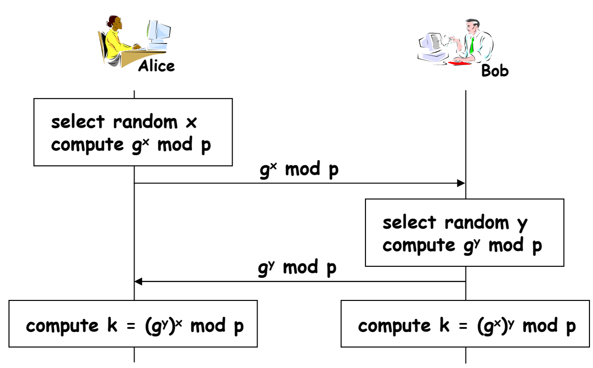
Key establishment protocols

- goal of key establishment protocols
  - to setup a shared secret between two (or more) parties
  - established shared secret is used as a *session key* to protect communication between the parties
- basic classification
  - key transport protocols
    - one party creates or otherwise obtains a secret value, and securely transfers it to the other party
  - key agreement protocols
    - a shared secret is derived by the parties as a function of information contributed by each, such that no party can predetermine the resulting value

Further services

- entity authentication
- implicit key authentication
  - one party is assured that no other party aside from a specifically identified second party (and possibly some trusted third parties) may gain access to the established session key
- key confirmation
  - one party is assured that a second (possibly unidentified) party actually possesses the session key
  - possession of a key can be demonstrated by
    - producing a one-way hash value of the key or
    - encryption of known data with the key
- key freshness
  - one party is assured that the key is new (never used before)

The Diffie-Hellman protocol



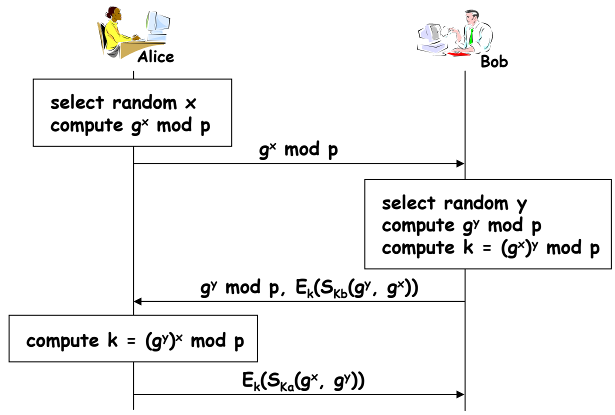
assumptions:

p is a large prime, g is a generator of Z_p^*,
both are publicly known system parameters

protocol characteristics:

key-agreement protocol
NO AUTHENTICATION
key freshness (randomly selected exponents)
no need for an (online) trusted third party

The Station-to-Station protocol



protocol characteristics:

mutual explicit key authentication (digital signatures, usage of the session key)

key freshness (random exponents)

off-line third party for issuing public key certificates is required

initial exchange of public keys between the parties may be required

Summary

- security is about how to prevent attacks, or – if prevention is not possible – how to detect attacks and recover from them
- an attack is a a *deliberate attempt* to compromise a system
- security is provided in form of security services that are implemented by using security mechanisms
- many security mechanisms are based on cryptography (e.g., encryption, digital signature, some data integrity mechanisms, some authentication schemes, etc.)

Naming and addressing

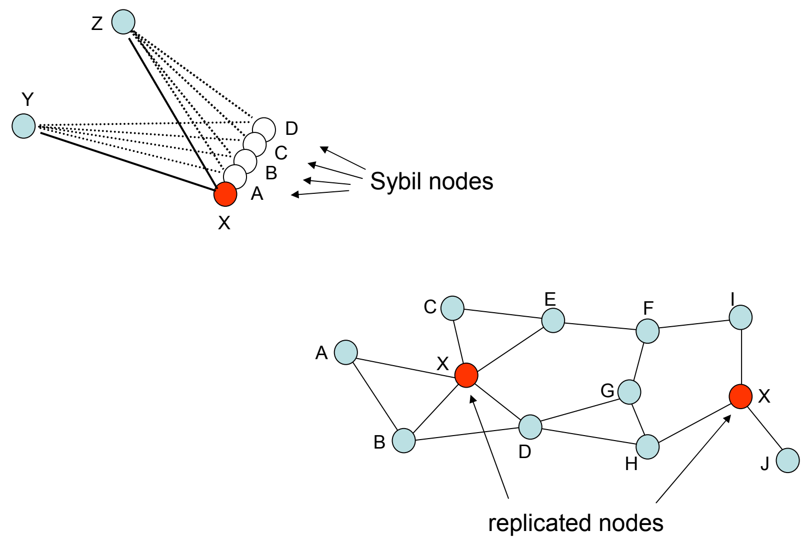
attacks against naming and addressing:
- address stealing
- Sybil attack
- node replication attack;

protection mechanisms:
- Cryptographically Generated Addresses
- witness based detection of node replication

Introduction

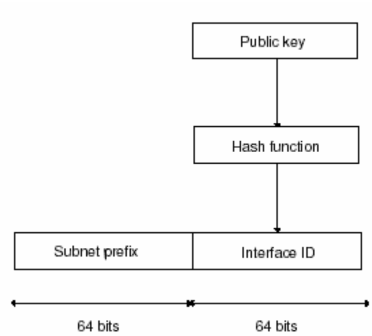
- naming and addressing are fundamental for networking
  - notably, routing protocols need addresses to route packets
  - services need names in order to be identifiable, discoverable, and useable
- attacks against naming and addressing
  - address stealing
    - adversary starts using an address already assigned to and used by a legitimate node
  - Sybil attack
    - a single adversarial node uses several invented addresses
    - makes legitimate nodes believe that there are many other nodes around
  - node replication attack
    - dual of the Sybil attack
    - the adversary introduces replicas of a single compromised node using the same address at different locations of the network

Illustration of the Sybil and node replication attacks



Cryptographically Generated Addresses (CGA)

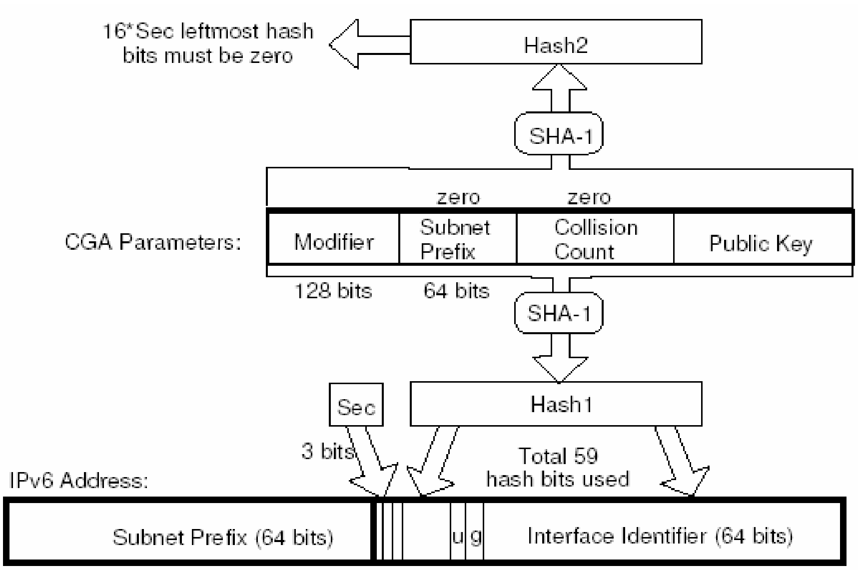
- aims at preventing address stealing
- general idea:
  - generate node address from a public key
  - corresponding private key is known only by the legitimate node
  - prove ownership of the address by proving knowledge of the private key
- example in case of IPv6:



A potential problem with CGA

- often only a limited number of bits of the address can be chosen arbitrarily (64 in our example)
- this number may be too small to guarantee second pre-image resistance
  - an adversary could pre-compute a large database of interface identifiers from public keys generated by himself, and use this database to find matches to victims' addresses
- a solution can be the technique called *hash extension*
  - increase the cost of address generation, and hence the cost of brute-force attacks, while keep constant the cost of address usage and verification

Hash extension



Protocol for CGA generation

1. Set the modifier field to a random 128-bit value.
2. Hash the concatenation of the modifier, 64+8 zero bits, and the encoded public key. The leftmost 112 bits of the result are Hash2.
3. Compare the 16*Sec leftmost bits of Hash2 with zero. If they are all zero (or if Sec=0), continue with Step (4). Otherwise, increment the modifier and go back to Step (2).
4. Set the collision count value to zero.
5. Hash the concatenation of the modifier, subnet prefix, collision count and encoded public key. The leftmost 64 bits of the result are Hash1.
6. Form an interface identifier by setting the two reserved bits in Hash1 both to 1 and the three leftmost bits to the value Sec.
7. Concatenate the subnet prefix and interface identifier to form a 128-bit IPv6 address.
8. If an address collision with another node within the same subnet is detected, increment the collision count and go back to step (5). However, after three collisions, stop and report the error.

Protocol for CGA verification

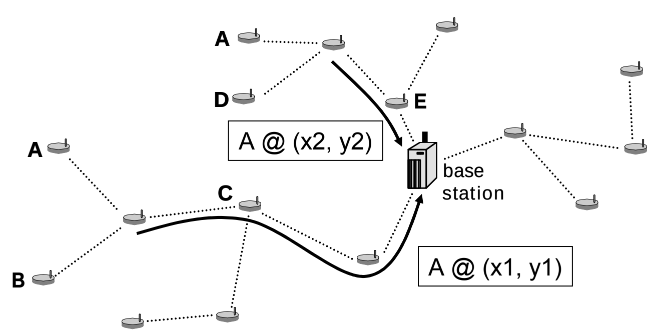
1. Check that the collision count value is 0, 1 or 2, and that the subnet prefix value is equal to the subnet prefix (i.e. leftmost 64 bits) of the address. The CGA verification fails if either check fails.
2. Hash the concatenation of the modifier, subnet prefix, collision count and the public key. The 64 leftmost bits of the result are Hash1.
3. Compare Hash1 with the interface identifier (i.e. the rightmost 64 bits) of the address. Differences in the two reserved bits and in the three leftmost bits are ignored. If the 64-bit values differ (other than in the five ignored bits), the CGA verification fails.
4. Read the security parameter Sec from the three leftmost bits of the interface identifier of the address.
5. Hash the concatenation of the modifier, 64+8 zero bits and the public key. The leftmost 112 bits of the result are Hash2.
6. Compare the 16*Sec leftmost bits of Hash2 with zero. If any one of these is nonzero, CGA verification fails. Otherwise, the verification succeeds.

Thwarting the Sybil attack

- note that CGAs do not prevent the Sybil attack
  - an adversary can still generate addresses for herself
- a solution based on a central and trusted authority
  - the central authority vouches for the one-to-one mapping between an address and a device
  - e.g., a server can respond to requests concerning the legitimacy of a given address
- other solutions take advantage of some physical aspects
  - e.g., identify the same device based on radio fingerprinting

Thwarting the node replication attack (1/2)

- a centralized solution
  - each node reports its neighbors' claimed locations to a central authority (e.g., the base station in sensor networks)
  - the central authority detects if the same address appears at two different locations
  - assumes location awareness of the nodes



Thwarting the node replication attack (2/2)

- a decentralized variant
  - neighbors' claimed location is forwarded to *witnesses*
  - witnesses are randomly selected nodes of the network
  - if a witness detects the same address appearing at two different locations then it broadcast this information and the replicated nodes are revoked

Analysis of the decentralized variant

- total number if nodes is n
- average number of neighbors is d
- each neighbor of A forwards A's location claim with probability p to g randomly selected witnesses
- average number of witnesses receiving A's location claim is p*d*g
- if there are L replicas of A, then for the probability of detection:

$$P_{det} > 1 - e^{-L(L-1)(pdg)2/2n}$$

- numerical example:

  n = 10000, d = 20, g = 100, p = 0.5
  $L = 2 \rightarrow P_{det} \sim 0.63$
  $L = 3 \rightarrow P_{det} \sim 0.95$

Conclusions

- there are various attacks against naming and addressing
  - address stealing
  - Sybil attack
  - node replication attack
- decentralization and lack of a central authority renders the defense against these attacks difficult
- proposed solutions (CGA, node replication detection using witnesses) provide only probabilistic guarantees
  - parameters should be chosen carefully

Secure routing

ad hoc network routing protocols;
attacks on routing;
countermeasures;
secured ad hoc network routing protocols;
the wormhole attack and its detection;

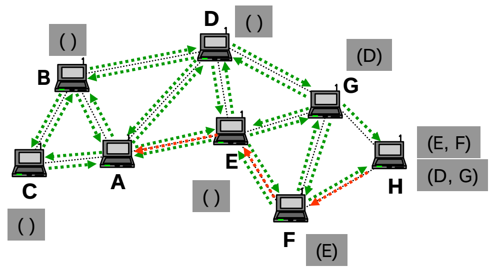
Ad hoc network routing protocols

- topology-based protocols
  - proactive
    - distance vector based (e.g., DSDV)
    - link-state (e.g., OLSR)
  - reactive (on-demand)
    - distance vector based (e.g., AODV)
    - source routing (e.g., DSR)
- position-based protocols
    - greedy forwarding (e.g., GPSR, GOAFR)
    - restricted directional flooding (e.g., DREAM, LAR)
- hybrid approaches

Example: Dynamic Source Routing (DSR)

- on-demand source routing protocol
- two components:
  - route discovery
    - used only when source S attempts to send a packet to destination D
    - based on flooding of Route Requests (RREQ) and returning Route Replies (RREP)
  - route maintenance
    - makes S able to detect route errors (e.g., if a link along that route no longer works)

DSR Route Discovery illustrated



A → *: [RREQ, id, A, H; ()]
B → *: [RREQ, id, A, H; (B)]
C → *: [RREQ, id, A, H; (C)]
D → *: [RREQ, id, A, H; (D)]
E → *: [RREQ, id, A, H; (E)]
F → *: [RREQ, id, A, H; (E, F)]
G → *: [RREQ, id, A, H; (D,G)]

H → A: [RREP, <source route>; (E, F)]

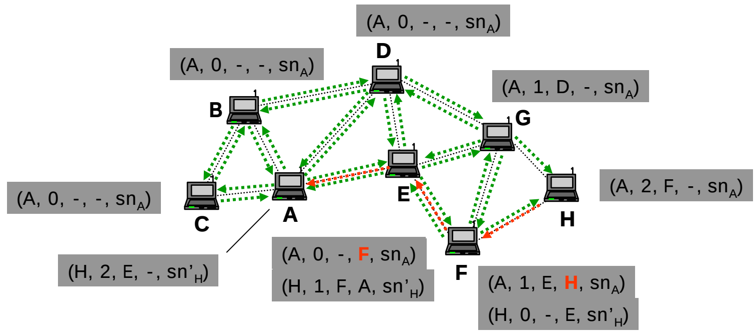
where <source route> is obtained

- from the route cache of H
- by reversing the route received in the RREQ
  - works only if all the links along the discovered route are bidirectional
  - IEEE 802.11 assumes that links are bidirectional
- by executing a route discovery from H to A
  - discovered route from A to H is piggy backed to avoid infite recursion

Example: Ad-hoc On-demand Distance Vector routing (AODV)

- on-demand distance vector routing
- uses sequence numbers to ensure loop-freedom and to detect out-of-date routing information
- operation is similar to that of DSR but the nodes maintain routing tables instead of route caches
- a routing table entry contains the following:
  - destination identifier
  - number of hops needed to reach the destination
  - identifier of the next hop towards the destination
  - list of precursor nodes (that may forward packets to the destination via this node)
  - destination sequence number

AODV Route Discovery illustrated

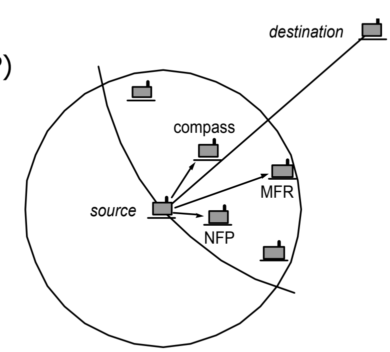


A → *: [RREQ, id, A, H, 0, sn_A, sn_H]
B → *: [RREQ, id, A, H, 1, sn_A, sn_H]
C → *: [RREQ, id, A, H, 1, sn_A, sn_H]
D → *: [RREQ, id, A, H, 1, sn_A, sn_H]
E → *: [RREQ, id, A, H, 1, sn_A, sn_H]
F → *: [RREQ, id, A, H, 2, sn_A, sn_H]
G → *: [RREQ, id, A, H, 2, sn_A, sn_H]

H → F: [RREP, A, H, 0, sn'_H]
F → E: [RREP, A, H, 1, sn'_H]
E → A: [RREP, A, H, 2, sn'_H]

Example: Position-based greedy forwarding

- assumptions
  - nodes are aware of their own positions and that of their neighbors
  - packet header contains the position of the destination
- packet is forwarded to a neighbor that is closer to the destination than the forwarding node
  - Most Forward within Radius (MFR)
  - Nearest with Forward Progress (NFP)
  - Compass forwarding
  - Random forwarding
- additional mechanisms are needed to cope with local minimums (dead-ends)



Attacks on routing protocols (1/2)

- general objectives of attacks
  - increase adversarial control over the communications between some nodes;
  - degrade the quality of the service provided by the network;
  - increase the resource consumption of some nodes (e.g., CPU, memory, or energy).
- adversary model
  - insider adversary
    - can corrupt legitimate nodes
  - the attacker is not all-powerful
    - it is not physically present everywhere
    - it launches attacks from regular devices

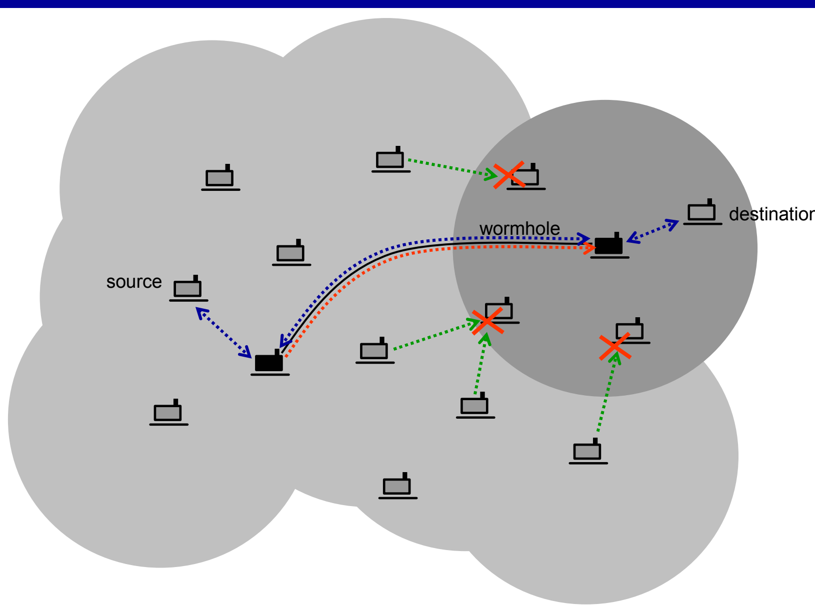
Attacks on routing protocols (2/2)

- attack mechanisms
  - eavesdropping, replaying, modifying, and deleting control packets
  - fabricating control packets containing fake routing information (forgery)
  - fabricating control packets under a fake identity (spoofing)
  - dropping data packets (attack against the forwarding function)
  - wormholes and tunneling
  - rushing
- types of attacks
  - route disruption
  - route diversion
  - creation of incorrect routing state
  - generation of extra control traffic
  - creation of a gray hole

Route disruption

- the adversary prevents a route from being discovered between two nodes that are otherwise connected
- the primary objective of this attack is to degrade the quality of service provided by the network
  - the two victims cannot communicate, and
  - other nodes can also suffer and be coerced to use suboptimal routes
- attack mechanisms that can be used to mount this attack:
  - dropping route request or route reply messages on a vertex cut
  - forging route error messages
  - combining wormhole/tunneling and control packet dropping
  - rushing

Example: Route disruption in DSR with rushing



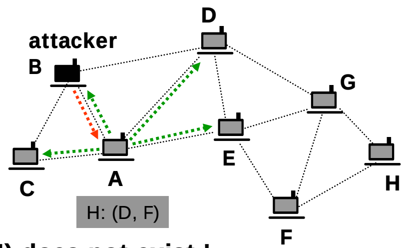
Route diversion

- due to the presence of the adversary, the protocol establishes routes that are different from those that it would establish, if the adversary did not interfere with the execution of the protocol
- the objective of route diversion can be
  - to increase adversarial control over the communications between some victim nodes
    - the adversary tries to achieve that the diverted routes contain one of the nodes that it controls or a link that it can observe
    - the adversary can eavesdrop or modify data sent between the victim nodes easier
  - to increase the resource consumption of some nodes
    - many routes are diverted towards a victim that becomes overloaded
  - degrade quality of service
    - by increasing the length of the discovered routes, and thereby, increasing the end-to-end delay between some nodes
- route diversion can be achieved by
  - forging or manipulating routing control messages
  - dropping routing control messages
  - setting up a wormhole/tunnel

Creation of incorrect routing state

- this attack aims at jeopardizing the routing state in some nodes so that the state appears to be correct but, in fact, it is not
  - data packets routed using that state will never reach their destinations
- the objective of creating incorrect routing state is
  - to increase the resource consumption of some nodes
    - the victims will use their incorrect state to forward data packets, until they learn that something goes wrong
  - to degrade the quality of service
- can be achieved by
  - spoofing, forging, modifying, or dropping control packets

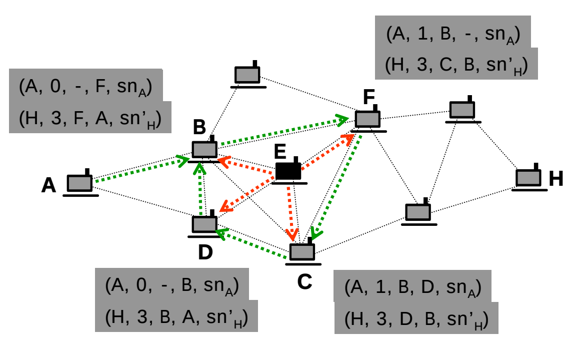
Example: Creation of incorrect routing state in DSR



Route (A, D, F, H) does not exist !

A → *: [RREQ, id, A, H; ()]
B → A: [RREP, <src route>, A, H; (D, F)]

Example: Creation of incorrect routing state in AODV



E (C) → F: [RREP, A, H, 2, sn'_H]
E (D)→ C: [RREP, A, H, 2, sn'_H]
E (B)→ D: [RREP, A, H, 2, sn'_H]
E (F)→ B: [RREP, A, H, 2, sn'_H]

Generation of extra control traffic

- injecting spoofed control packets into the network
- aiming at increasing resource consumption due to the fact that such control packets are often flooded in the entire network

Setting up a gray hole

- an adversarial node selectively drops data packets that it should forward
- the objective is
  - to degrade the quality of service
    - packet delivery ratio between some nodes can decrease considerably
  - to increase resource consumption
    - wasting the resources of those nodes that forward the data packets that are finally dropped by the adversary
- implementation is trivial
  - adversarial node participates in the route establishment
  - when it receives data packets for forwarding, it drops them
  - even better if combined with wormhole/tunneling

Countermeasures

- authentication of control packets
  – using MACs or digital signatures
- protection of mutable information in control packets
  – using MACs or digital signatures
  – often complemented with the use of one-way hash functions
- detecting wormholes and tunnels
- combating gray holes
  – using multi-path routing
  – using a "detect and react" approach

Authentication of control packets

- questions:
  – Who should authenticate the control packets?
  – Who should be able to verify authenticity?
- ***control packets should be authenticated by their originators***
- ***authenticity should be verifiable by the target of the control packet***
- moreover, ***each node that updates its routing state as a result of processing the control packet must be able to verify its authenticity***
  – the adversary can still mount resource consumption attacks
- each node that processes and re-broadcasts or forwards the control packet must be able to verify its authenticity
- as it is not known in advance which nodes will process a given control packet, we need a ***broadcast authentication*** scheme

Protection of mutable information in control packets

- often, intermediate nodes add information to the control packet before re-broadcasting or forwarding it (hop count, node list, etc.)
- this added information is not protected by control packet origin authentication
- ***each node that adds information to the packet should authenticate that information in such a way that each node that acts upon that information can verify its authenticity***
- this works for traceable additions (e.g., adding node identifiers), but what about untraceable additions (e.g., increasing the hop count)?

Protection of traceable modifications

- the entire control packet can be re-signed by each node that modifies it
- problems:
  - signatures can be removed from the end
    - one-way hash chains can be used (e.g., Ariadne)
    - efficient aggregate signatures provide better solution
  - re-signing increases the resource consumption of the nodes (potentially each node needs to re-sign broadcast messages)
    - no easy way to overcome this problem
    - one approach is to avoid mutable information in control packets
    - another approach is to scarify some amount of security (e.g., SRP)
  - corrupted nodes can still add incorrect information and sign it
    - very tough problem …

Protection of untraceable modifications

- no perfect solution exists (trust problem)
- hop counts are often protected by a per-hop hashing mechanism (e.g., SAODV, SEAD)
  - control packets contain a hash value associated with the hop-count
  - when the control packet is forwarded or re-broadcast, the hop-count is incremented and the hash value is hashed once
  - adversarial nodes cannot decrease hop-count values in control packets because that would need to compute pre-images of hash values
  - adversary can still increase the hop-count …
- another approach is to eliminate hop-counts
  - use other routing metrics (e.g., ARAN uses the delay as the routing metric)

Combating gray holes

- two approaches:
  - use multiple, preferably disjoint routes
    - increased robustness
    - but also increased resource consumption
    - resource consumption can be somewhat decreased by applying the principles of error correcting coding
      - data packet is coded and the coded packet is split into smaller chunks
      - a threshold number of chunks is sufficient to reconstruct the entire packet
      - chunks are sent over different routes
  - detect and react
    - monitor neighbors and identify misbehaving nodes
    - use routes that avoid those misbehaving nodes
    - reputation reports about nodes can be spread in the network
    - this approach has several problems
      - how to detect reliably that a node is misbehaving?
      - how to prevent false accusations and spreading of negative reputations?

Some secure ad hoc network routing protocols

- SRP (on-demand source routing)
- Ariadne (on-demand source routing)
- endairA (on-demand source routing)
- S-AODV (on-demand distance vector routing)
- ARAN (on-demand, routing metric is the propagation delay)
- SEAD (proactive distance vector routing)
- SMT (multi-path routing combined error correcting)
- Watchdog and Pathrater (implementation of the "detect and react" approach to defend against gray holes)
- ODSBR (source routing with gray hole detection)

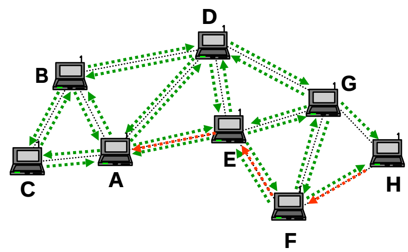
SRP (Secure Routing Protocol)

- SRP is a secure variant of DSR
- uses symmetric-key authentication (MACs)
  - due to mobility, it would be impractical to require that the source and the destination share keys with all intermediate nodes
  - hence there's only a shared key between the source and the destination

  → only end-to-end authentication is possible

  → no optimizations
- SRP is simple but it does not prevent the manipulation of mutable information added by intermediate nodes
  - this opens the door for some attacks

SRP operation illustrated



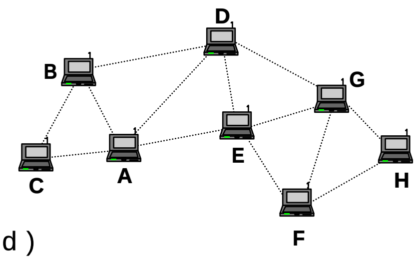
A → * : [RREQ, A, H, id, sn, mac_{AH}, ()]
B → * : [RREQ, A, H, id, sn, mac_{AH}, (B)]
C → * : [RREQ, A, H, id, sn, mac_{AH}, (C)]
D → * : [RREQ, A, H, id, sn, mac_{AH}, (D)]
E → * : [RREQ, A, H, id, sn, mac_{AH}, (E)]
F → * : [RREQ, A, H, id, sn, mac_{AH}, (E, F)]
G → * : [RREQ, A, H, id, sn, mac_{AH}, (D, G)]

H → A : [RREP, A, H, id, sn, (E, F), mac_{HA}]

Ariadne

- Ariadne is another secured variant of DSR
- it uses control message authentication to prevent modification and forgery of routing messages
  - based on signatures, MACs, or TESLA
- it uses a per-hop hash mechanism to prevent the manipulation of the accumulated route information in the route request message

Ariadne with signatures illustrated



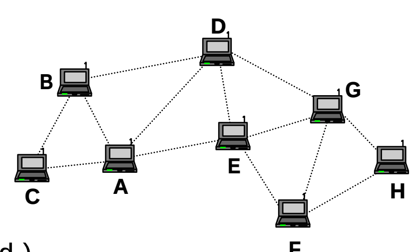
A : h_A = mac_{AH}(RREQ | A | H | id)
A → * : [RREQ, A, H, id, h_A, (), ()]

E : h_E = H(E | h_A)
E → * : [RREQ, A, H, id, h_E, (E), (sig_E)]

F : h_F = H(F | h_E)
F → * : [RREQ, A, H, id, h_F, (E, F), (sig_E, sig_F)]

H → A: [RREP, H, A, (E, F), (sig_E, sig_F), sig_H]

Ariadne with standard MACs illustrated



A : h_A = mac_{AH}(RREQ | A | H | id)
A → * : [RREQ, A, H, id, h_A, (), ()]

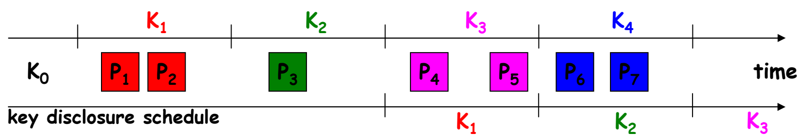
E : h_E = H(E | h_A)
E → * : [RREQ, A, H, id, h_E, (E), (mac_{EH})]

F : h_F = H(F | h_E)
F → * : [RREQ, A, H, id, h_F, (E, F), (mac_{EH}, mac_{EH})]

H → A : [RREP, H, A, (E, F), mac_{HA}]

Symmetric-key broadcast authentication with TESLA

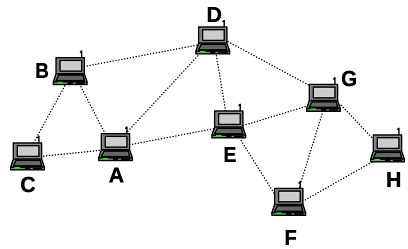
- MAC keys are consecutive elements in a one-way key chain:
  - $K_n \rightarrow K_{n-1} \rightarrow \ldots \rightarrow K_0$
  - $K_i = h(K_{i+1})$
- TESLA protocol:
  - setup: K_0 is sent to each node in an authentic way
  - time is divided into epochs
  - each message sent in epoch i is authenticated with key K_i
  - K_i is disclosed in epoch i+d, where d is a system parameter
  - K_i is verified by checking $h(K_i) = K_{i-1}$
- example:



Ariadne with TESLA

- assumptions:
  - each source-destination pair (S, D) shares a symmetric key K_{SD}
  - each node F has a TESLA key chain $K_{F,i}$
  - each node knows an authentic TESLA key of every other node
- route request (source S, destination D):
  - S authenticates the request with a MAC using K_{SD}
  - each intermediate node F appends a MAC computed with its current TESLA key
  - D verifies the MAC of S
  - D verifies that the TESLA key used by F to generate its MAC has not been disclosed yet
- route reply:
  - D generates a MAC using K_{SD}
  - each intermediate node delays the reply until it can disclose its TESLA key that was used to generate its MAC
  - F appends its TESLA key to the reply
  - S verifies the MAC of D, and all the MACs of the intermediate nodes

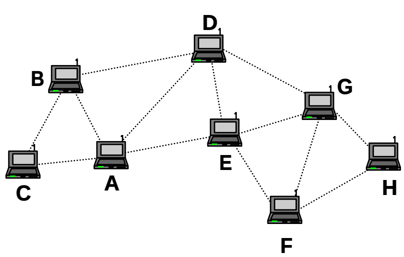
Ariadne with TESLA illustrated



A → *: [RREQ, A, H, id, h_A, (), ()]
E → *: [RREQ, A, H, id, h_E, (E), ($mac_{K_{E,i}}$)]
F → *: [RREQ, A, H, id, h_F, (E, F), ($mac_{K_{E,i}}$, $mac_{K_{F,i}}$)]

H → F: [RREP, H, A, (E, F), ($mac_{K_{E,i}}$, $mac_{K_{F,i}}$), mac_{HA}, ()]
F → E: [RREP, H, A, (E, F), ($mac_{K_{E,i}}$, $mac_{K_{F,i}}$), mac_{HA}, ($K_{F,i}$)]
E → A: [RREP, H, A, (E, F), ($mac_{K_{E,i}}$, $mac_{K_{F,i}}$), $mac_{K_{HA}}$, ($K_{F,i}$, $K_{E,i}$)]

endairA



target verifies:
- there's no repeating ID in the node list
- last node in the node list is a neighbor

each intermediate node verifies:
- its own ID is in the node list
- there's no repeating ID in the node list
- next and previous nodes in the node list are neighbors
- all signatures are valid

source verifies:
- there's no repeating ID in the node list
- first node in the node list is a neighbor
- all signatures are valid

A → * : [RREQ, A, H, id, ()]
E → * : [RREQ, A, H, id, (E)]
F → * : [RREQ, A, H, id, (E, F)]

H → F : [RREP, A, H, id, (E, F), (sig_H)]
F → E : [RREP, A, H, id, (E, F), (sig_H, sig_F)]
E → A : [RREP, A, H, id, (E, F), (sig_H, sig_F, sig_E)]

Properties of endairA

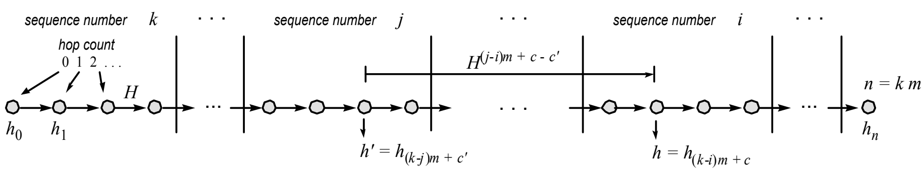
- security
  - endairA is provably secure if the signature scheme is secure against chosen message attacks
- efficiency
  - endairA requires less computation
    - route reply is signed and verified only by the nodes on the route
    - in Ariadne, route request is signed (and potentially verified) by every node in the network

SAODV (Secure AODV)

- SAODV is a secure variant of AODV
- protects non-mutable information with a digital signature (of the originator of the control packet)
- uses hash chains for the protection of the HopCount value
  - new non-mutable fields:
    - MaxHopCount (= TTL)
    - TopHash (= iterative hash of a random seed MaxHopCount times)
  - new mutable field:
    - Hash (contains the current hash value corresponding to the HopCount value)
- operation
  - initially Hash is set to the seed
  - each time a node increases HopCount, it also replaces Hash with H(Hash)
  - verification of the HopCount is done by hashing the Hash field MaxHopCount-HopCount times and checking if the result matches TopHash

SEAD (Secure Efficient Ad hoc Distance vector routing)

- SEAD is a proactive distance vector protocol
  - it can be viewed as a secure variant of DSDV
- SEAD tries to ensure that
  - sequence numbers cannot be increased
  - hop count values cannot be decreased
- operation
  - each node has a hash chain of length k times m (where m is the maximum diameter of the network)
  - when a node sends out a route update message about itself with sequence number i and hop count 0, it reveals $h_{(k-i)m}$
  - any node can increase the hop count by computing $h_{(k-i)m+c}$
  - any node can verify if the sequence number is greater than any previously known value

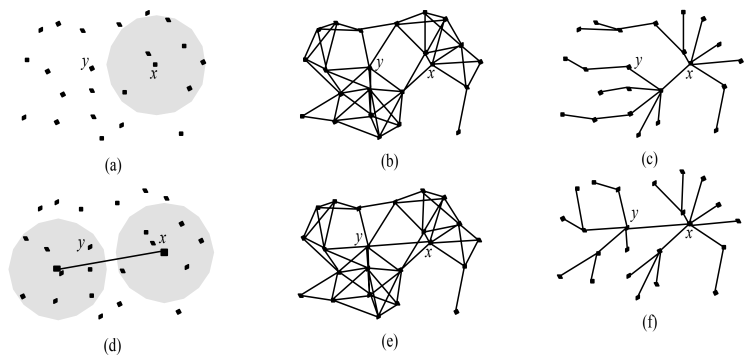


The wormhole attack

- a wormhole is an out-of-band connection, controlled by the adversary, between two physical locations in the network
  - the adversary installs radio transceivers at both ends of the wormhole
  - it transfers packets (possibly selectively) received from the network at one end of the wormhole to the other end via the out-of-band connection, and re-injects the packets there into the network
- notes:
  - adversary's transceivers are not regular nodes (no node is compromised by the adversary)
  - adversary doesn't need to understand what it tunnels (e.g., encrypted packets can also be tunneled through the wormhole)
  - it is easy to mount a wormhole, but it may devastating effects on routing

Effects of a wormhole

- at the data link layer: distorted network topology



- at the network layer:
  - routing protocols may choose routes that contain wormhole links
    - typically those routes appear to be shorter
    - flooding based routing protocols (e.g., DSR, Ariadne) may not be able to discover other routes but only through the wormhole
  - adversary can then monitor traffic or drop packets (DoS)

Classification of wormhole detection methods

- centralized mechanisms
  - data collected from the local neighborhood of every node are sent to a central entity
  - based on the received data, a model of the entire network is constructed
  - the central entity tries to detect inconsistencies (potential indicators of wormholes) in this model
  - can be used in sensor networks, where the base station can play the role of the central entity

- decentralized mechanisms
  - each node constructs a model of its own neighborhood using locally collected data
  - each node tries to detect inconsistencies on its own
  - advantage: no need for a central entity (fits well some applications)
  - disadvantage: nodes need to be more complex

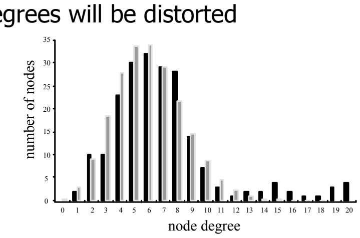
Statistical wormhole detection in sensor NWs

- each node reports its list of believed neighbors to the base station
- the base station reconstructs the connectivity graph (model)
- *a wormhole always increases the number of edges* in the connectivity graph
- this increase may change the properties of the connectivity graph in a detectable way (anomaly)
- detection can be based on statistical hypothesis testing methods (e.g. the χ^2-test)

Two examples

- a wormhole that creates many new edges may increase the *number of neighbors* of the affected nodes

→ distribution of node degrees will be distorted



- a wormhole is usually a shortcut that decreases the length of the shortest paths in the network

→ distribution of the length of the shortest paths will be distorted

Packet leashes

- packet leashes ensure that packets are not accepted "too far" from their source
- geographical leashes
  - each node is equipped with a GPS receiver
  - when sending a packet, the node puts its GPS position into the header
  - the receiving node verifies if the sender is really within communication range
- temporal leashes
  - nodes' clocks are very tightly synchronized
  - when sending a packet, the node puts a timestamp in the header
  - the receiving node estimates the distance of the sender based on the elapsed time and the speed of light

$$d_{est} < v_{light}(t_{rcv} - t_{snd} + \Delta_t)$$

  - note: $v_{light}\,\Delta_t$ must be much smaller than the communication range

Conclusions

- routing is a fundamental function in networking, hence, an ideal target for attacks
- attacks against routing aim at
  - increasing adversarial control over the communications between some nodes;
  - degrading the quality of the service provided by the network;
  - increasing the resource consumption of some nodes (e.g., CPU, memory, or energy)
- many attacks (but not all!) can be prevented by authenticating routing control messages
- it is difficult to protect the mutable parts of control messages
- special attacks (e.g., wormholes and rushing) needs special protection mechanisms
- several secured ad hoc network routing protocols have been proposed
- some of them have weaknesses that are exploitable by attacks

Outline

- New wireless networks and new challenges (25')
- Thwarting malicious behavior
  - introduction to cryptography and security techniques (30')
  - naming and addressing (20')
  - secure routing (30')
- Thwarting selfish behavior
  - introduction to game theory (30')
  - selfishness in packet forwarding (20')
  - border games in cellular networks (20')

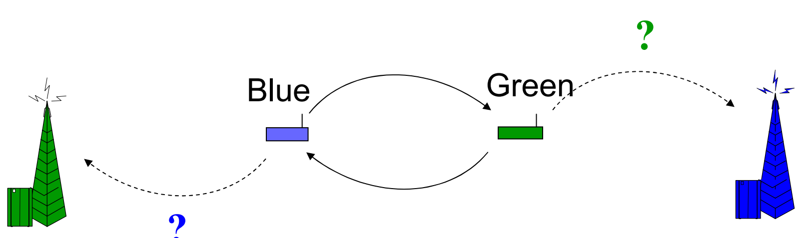
Introduction to game theory

strategic form games;
solution concepts:
- strict dominance
- weak dominance
- Nash equilibrium;
Pareto optimality;
repeated games;

What is game theory?

- Discipline aiming at modeling situations in which actors have to make decisions which have mutual, **possibly conflicting**, consequences
- Classical applications: **economics**, but also politics and biology
- Example: should a company invest in a new plant, or enter a new market, considering that the **competition** *may* make similar moves?
- Most widespread kind of game: **non-cooperative** (meaning that the players do not attempt to find an agreement about their possible moves)

Example 1: The Forwarder's Dilemma



E1: From a problem to a game

- users controlling the devices are ***rational*** = try to maximize their benefit
- game formulation: G = (P,S,U)
  - P: set of players
  - S: set of strategy functions
  - U: set of utility functions →

• Reward for packet reaching the destination: 1
• Cost of packet forwarding: c (0 < c << 1)

- ***strategic-form*** representation

Blue \ Green	Forward	Drop
Forward	(1-c, 1-c)	(-c, 1)
Drop	(1, -c)	(0, 0)

Solving the Forwarder's Dilemma (1/2)

Strict dominance: strictly best strategy, for any strategy of the other player(s)

Strategy s_i strictly dominates if

$$u_i(s_i', s_{-i}) < u_i(s_i, s_{-i}), \forall s_{-i} \in S_{-i}, \forall s_i' \in S_i$$

where: $u_i \in U$ utility function of player *i*

$s_{-i} \in S_{-i}$ strategies of all players except player i

In Example 1, strategy Drop ***strictly dominates*** strategy Forward

Blue \ Green	Forward	Drop
Forward	(1-c, 1-c)	(-c, 1)
Drop	(1, -c)	(0, 0)

Solving the Forwarder's Dilemma (2/2)

Solution by iterative strict dominance:

Blue \ Green	Forward	Drop
Forward	(1-c, 1-c)	(-c, 1)
Drop	(1, -c)	(0, 0)

BUT

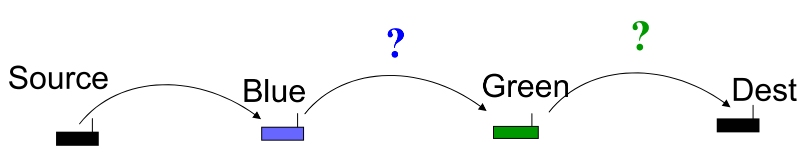
Drop ***strictly dominates*** Forward

Forward would result in a ***better outcome***

} Dilemma

Result: Tragedy of the commons ! (Hardin, 1968)

Example 2: The Joint Packet Forwarding Game



- Reward for packet reaching the destination: 1
- Cost of packet forwarding: c ($0 < c << 1$)

Blue \ Green	Forward	Drop
Forward	(1-c, 1-c)	(-c, 0)
Drop	(0, 0)	(0, 0)

No strictly dominated strategies !

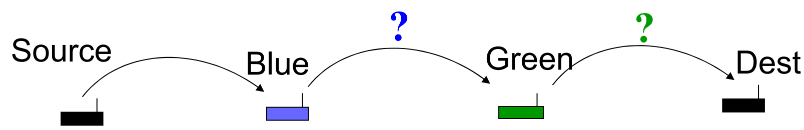
E2: Weak dominance

Weak dominance: strictly better strategy for at least one opponent strategy

Strategy s'_i is weakly dominated by strategy s_i if

$$u_i(s_i^{'}, s_{-i}) \leq u_i(s_i, s_{-i}), \forall s_{-i} \in S_{-i}$$

with strict inequality for at least one s_{-i}



Iterative weak dominance

BUT

The result of the iterative weak dominance is not unique in general !

Blue \ Green	Forward	Drop
Forward	(1-c, 1-c)	(-c, 0)
Drop	(0, 0)	(0, 0)

Nash equilibrium (1/2)

Nash Equilibrium: no player can increase its utility by deviating unilaterally

E1: The Forwarder's Dilemma

Blue \ Green	Forward	Drop
Forward	(1-c, 1-c)	(-c, 1)
Drop	(1, -c)	$\boxed{(0, 0)}$

E2: The Joint Packet Forwarding game

Blue \ Green	Forward	Drop
Forward	$\boxed{(1-c, 1-c)}$	(-c, 0)
Drop	(0, 0)	$\boxed{(0, 0)}$

Nash equilibrium (2/2)

Strategy profile s^* constitutes a **Nash equilibrium** if, for each player *i*,

$$u_i(s_i^*, s_{-i}^*) \geq u_i(s_i, s_{-i}^*), \forall s_i \in S_i$$

where: $u_i \in U$ utility function of player i

$s_i \in S_i$ strategy of player i

The **best response** of player *i* to the profile of strategies s_{-i} is a strategy s_i such that:

$$b_i(s_{-i}) = \arg\max_{s_i \in S_i} u_i(s_i, s_{-i})$$

Nash Equilibrium = Mutual best responses

Caution! Many games have more than one Nash equilibrium

Efficiency of Nash equilibria

E2: The Joint Packet Forwarding game

Blue \ Green	Forward	Drop
Forward	$\boxed{(1\text{-}c, 1\text{-}c)}$	(-c, 0)
Drop	(0, 0)	$\boxed{(0, 0)}$

How to choose between several Nash equilibria ?

Pareto-optimality: A strategy profile is Pareto-optimal if it is not possible to increase the payoff of any player without decreasing the payoff of another player.

Repeated games

- repeated interaction between the players (in stages)
- move: decision in one interaction
- strategy: defines how to choose the next move, given the previous moves
- history: the ordered set of moves in previous stages
  - most prominent games are history-1 games (players consider only the previous stage)
- initial move: the first move with no history
- finite-horizon vs. infinite-horizon games
- stages denoted by t (or k)

Utilities in repeated games

- finite-horizon vs. infinite-horizon games
- myopic vs. long-sighted repeated game

myopic: $\overline{u}_i = u_i(t+1)$

long-sighted finite: $\overline{u}_i = \sum_{t=0}^{T} u_i(t)$

long-sighted infinite: $\overline{u}_i = \sum_{t=0}^{\infty} u_i(t)$

utility with discounting: $\overline{u}_i = \sum_{t=0}^{\infty} u_i(t) \cdot \omega^t$

$0 < \omega \leq 1$ is the discounting factor

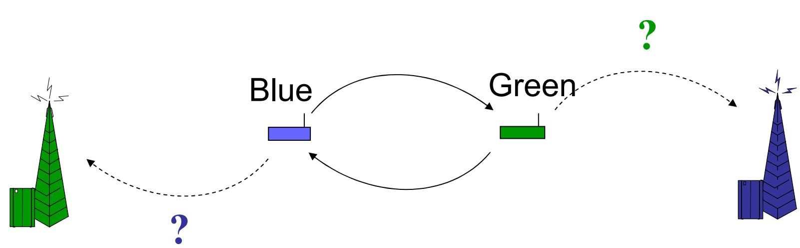
Strategies in repeated games

- usually, history-1 strategies, based on different inputs:
  - others' behavior: $m_i(t+1) = s_i\left[m_{-i}(t)\right]$
  - others' and own behavior: $m_i(t+1) = s_i\left[m_i(t), m_{-i}(t)\right]$
  - utility: $m_i(t+1) = s_i\left[u_i(t)\right]$

Example strategies in the Forwarder's Dilemma:

Blue (t)	initial move	F	D	strategy name
Green (t+1)	F	F	F	AllC
	F	F	D	Tit-For-Tat (TFT)
	D	D	D	AllD
	F	D	F	Anti-TFT

The Repeated Forwarder's Dilemma



Blue \ Green	Forward	Drop
Forward	(1-c, 1-c)	(-c, 1)
Drop	(1, -c)	(0, 0)

stage payoff

Analysis of the Repeated Forwarder’s Dilemma (1/3)

infinite game with discounting: $\overline{u}_i = \sum_{t=0}^{\infty} u_i(t) \cdot \omega^t$

Blue strategy	Green strategy
AllD	AllD
AllD	TFT
AllD	AllC
AllC	AllC
AllC	TFT
TFT	TFT

Blue utility	Green utility
0	0
1	-c
1/(1-ω)	-c/(1-ω)
(1-c)/(1-ω)	(1-c)/(1-ω)
(1-c)/(1-ω)	(1-c)/(1-ω)
(1-c)/(1-ω)	(1-c)/(1-ω)

Analysis of the Repeated Forwarder’s Dilemma (2/3)

Blue strategy	Green strategy
AllD	AllD
AllD	TFT
AllD	AllC
AllC	AllC
AllC	TFT
TFT	TFT

Blue utility	Green utility
0	0
1	-c
1/(1-ω)	-c/(1-ω)
(1-c)/(1-ω)	(1-c)/(1-ω)
(1-c)/(1-ω)	(1-c)/(1-ω)
(1-c)/(1-ω)	(1-c)/(1-ω)

- AllC receives a high payoff with itself and TFT, but
- AllD exploits AllC
- AllD performs poor with itself
- TFT performs well with AllC and itself, and
- TFT retaliates the defection of AllD

TFT is the best strategy if ω is high !

Analysis of the Repeated Forwarder's Dilemma (3/3)

Blue strategy	Green strategy	Blue utility	Green utility
AllD	AllD	0	0
TFT	TFT	$(1-c)/(1-\omega)$	$(1-c)/(1-\omega)$

Theorem: In the Repeated Forwarder's Dilemma, if both players play AllD, it is a Nash equilibrium.

Theorem: In the Repeated Forwarder's Dilemma, both players playing TFT is a Nash equilibrium $c < \omega$.

The Nash equilibrium s_{Blue} = TFT and s_{Green} = TFT is Pareto-optimal (but s_{Blue} = AllD and s_{Green} = AllD is not) !

Conclusions

- Game theory can help modeling greedy behavior in wireless networks
- Discipline still in its infancy
- Alternative solutions
  - Ignore the problem
  - Build protocols in tamper-resistant hardware

Security and Cooperation in Wireless Networks

http://secowinet.epfl.ch/

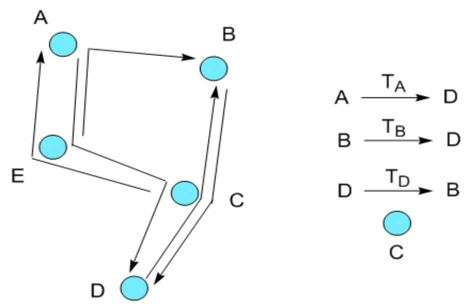
Selfishness in packet forwarding

Introduction

- the operation of multi-hop wireless networks requires the nodes to forward data packets on behalf of other nodes
- however, such cooperative behavior has no direct benefit for the forwarding node, and it consumes valuable resources (battery)
- hence, the nodes may tend to behave selfishly and deny cooperation
- if many nodes defect, then the operation of the entire network is jeopardized
- questions:
  - What are the conditions for the emergence of cooperation in packet forwarding?
  - Can it emerge spontaneously or should it be stimulated by some external mechanism?

Modeling packet forwarding as a game

Players: nodes



Strategy:
cooperation level

Payoff (of node i):
proportion of packets sent by node i reaching their destination

Cost function

Cost for forwarder f_j:

$$\eta_{f_j}(r,t) = -T_s(r)\cdot c\cdot\hat{\tau}_j(r,t)$$

where:

$T_s(r)$ – traffic sent by source s on route r
c – unit cost of forwarding

Example :

$$\hat{\tau}_C(r,t) = \prod_{k\in\{E,C\}} p_{f_k}(t) = p_E(t)\cdot p_C(t)$$

$$\eta_C(r,t) = -T_A(r)\cdot c\cdot\hat{\tau}_j(r,t)$$

Normalized throughput at forwarder f_j:

$$\hat{\tau}_j(r,t) = \prod_{k=1}^{j} p_{f_k}(t)$$

where:

r – route on which f_k is a forwarder
t – time slot
f_k – forwarders on route r
p_{f_k} – cooperation level of forwarder f_k

r (A→D): A → E → C → D

T_A $\quad p_E(t) \quad p_C(t)$

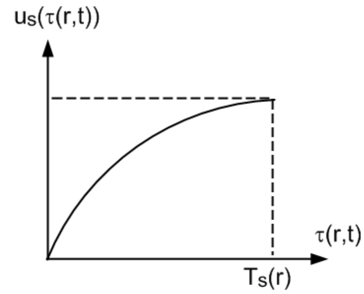
Utility function

Experienced throughput :

$$\tau(r,t) = T_s(r) \cdot \prod_{k=1}^{l} p_{f_k}(t)$$

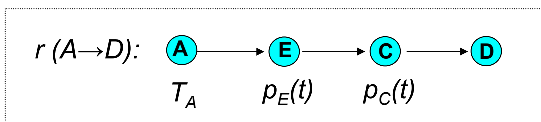
where: s – source
r – route on which s is a source
t – time slot
f_k – forwarders for s
p_{f_k} – cooperation level of forwarder f_k

Utility function :



Example :

$$\tau(r,t) = T_A(r) \cdot p_E(t) \cdot p_C(t)$$



Total payoff

Payoff = Utility - Cost

$$\pi_i(t) = \sum_{q \in S_i(t)} u_i(\tau(q,t)) + \sum_{r \in F_i(t)} \eta_i(r,t)$$

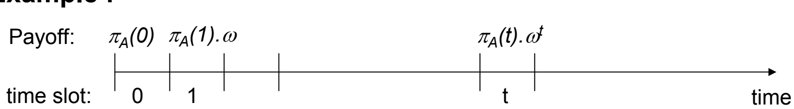
where: $S_i(t)$ – set of routes on which i is a source
$F_i(t)$ – set of routes on which i is a forwarder

The goal of each node is to maximize its total payoff over the game

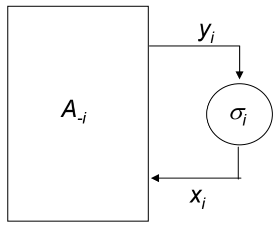
$$\max \ \overline{\pi}_i = \sum_{t=0}^{\infty} \pi_i(t) \cdot \omega^t$$

where: ω – discounting factor
t – time

Example :



Representation of the nodes as players



Strategy function for node *i*:

$$p_i(t) = \sigma_i([\tau(r,t-1)]_{r \in S_i(t-1)})$$

where:

$\tau\,(r,t)$ – experienced throughput
S_i – set of routes on which i is a source

Node *i* is playing against the rest of the network (represented by the box denoted by A_{-i})

Examples of strategies

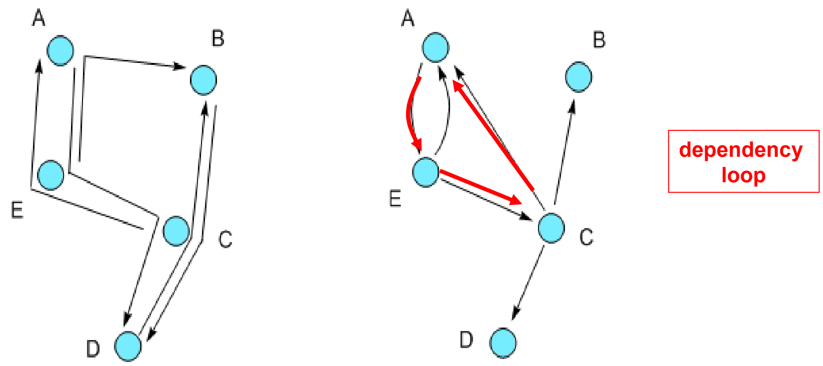
Strategy	Initial cooperation level	Function $\sigma_i(y_i) = x_i$
AllD (always defect)	0	$\sigma_i(y_i) = 0$
AllC (always cooperate)	1	$\sigma_i(y_i) = 1$
TFT (Tit-For-Tat)	1	$\sigma_i(y_i) = y_i$

where y_i stands for the input

- **non-reactive strategies**:
  the output of the strategy function
  is independent of the input (example: AllD and AllC)
- **reactive strategies**:
  the output of the strategy function
  depends on the input (example: TFT)

Concept of dependency graph

dependency: the benefit of each source is dependent on the behavior of its forwarders

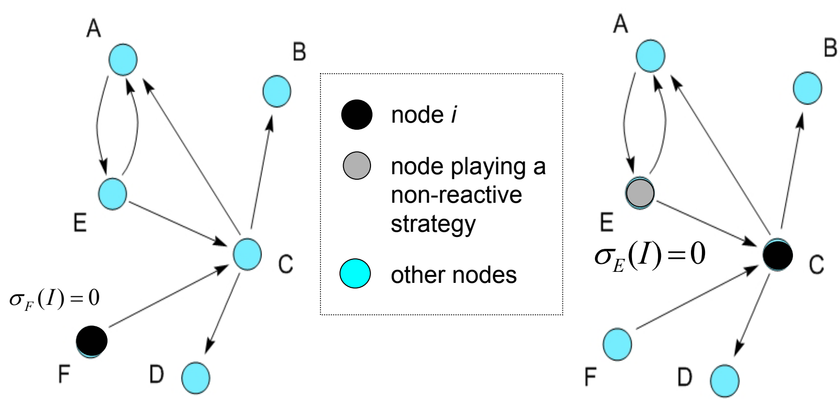


(a) routes (b) dependency graph

Analytical Results (1/2)

Theorem 1: If node *i* does not have any dependency loops, then its best strategy is AllD.

Theorem 2: If node *i* has only non-reactive dependency loops, then its best strategy is AllD.



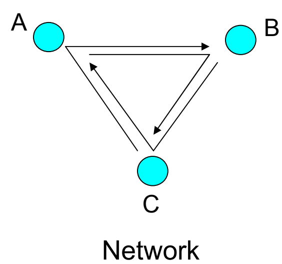
Corollary 1: If every node plays AllD, it is a Nash-equilibrium.

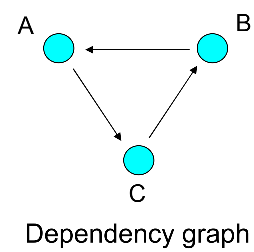
Analytical results (2/2)

Theorem 3 (simplified): Assuming that node *i* is a forwarder, its behavior will be cooperative only if it has a dependency loop with each of its sources

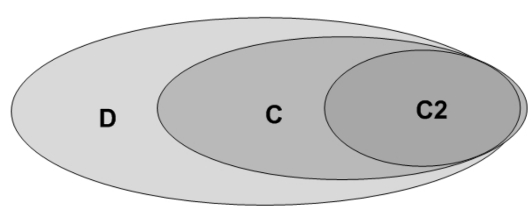
Corollary 2: If Theorem 3 holds for every node, it is a Nash-equilibrium.

Example in which Corollary 2 holds:





Classification of scenarios



D: Set of scenarios, in which every node playing AllD is a Nash equilibrium

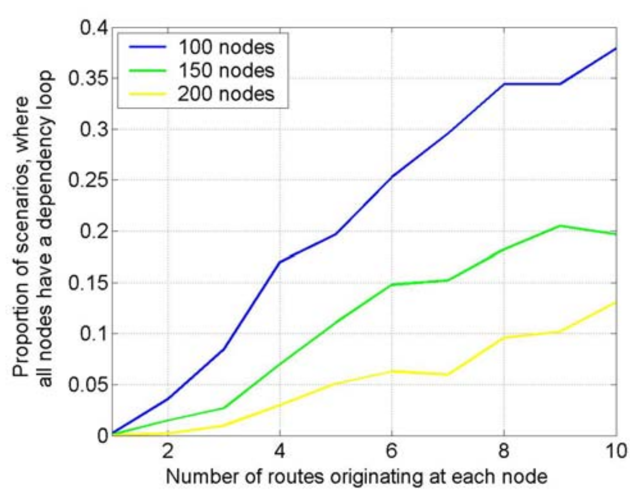
C: Set of scenarios, in which a Nash equilibrium based on cooperation is not excluded by Theorem 1

C2: Set of scenarios, in which cooperation is based on the conditions expressed in Corollary 2

Simulation settings

Number of nodes	100, 150, 200
Distribution of the nodes	random uniform
Area type	torus
Area size	1500x1500m, 1850x1850m, 2150x2150m
Radio range	200 m
Number of routes originating at each node	1-10
Route selection	shortest path
Number of simulation runs	1000

Simulation results



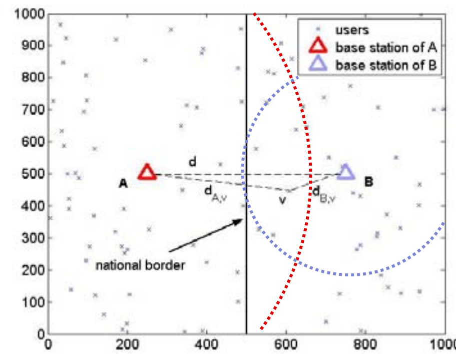
Conclusions

- Analytical results:
  - If everyone drops all packets, it is a Nash-equilibrium
  - **In theory**, given some conditions, a cooperative Nash-equilibrium can exist (i.e., each forwarder forwards all packets)
- Simulation results:
  - **In practice**, the conditions for cooperative Nash-equilibria are very restrictive : the likelihood that the conditions for cooperation hold for every node is extremely small
- Consequences:
  - Cooperation cannot be taken for granted
  - Mechanisms that stimulate cooperation are necessary
    - incentives based on virtual currency
    - reputation systems

Border games in cellular networks

Introduction

- spectrum licenses do not regulate access over national borders
- adjust pilot power to attract more users



Is there an incentive for operators to apply competitive pilot power control?

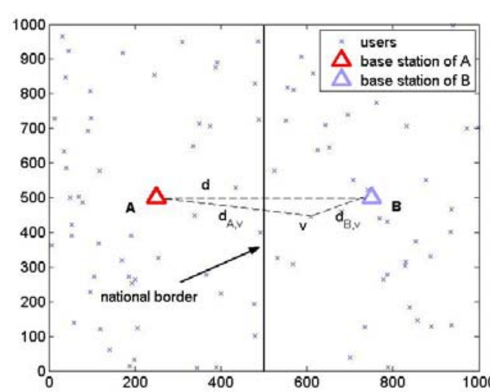
System model (1/2)

Network:

- cellular networks using CDMA
  - channels defined by orthogonal codes
- two operators: A and B
- one base station each
- pilot signal power control

Users:

- roaming users
- users uniformly distributed
- select the best quality BS
- selection based signal-to-interference-plus-noise ratio (*SINR*)



System model (2/2)

pilot signal *SINR*:

$$SINR_{iv}^{pilot} = \frac{G_p^{pilot} \cdot P_i \cdot g_{iv}}{N_0 \cdot \mathrm{W} + I_{own}^{pilot} + I_{other}^{pilot}}$$

$$I_{own}^{pilot} = \varsigma \cdot g_{iv} \cdot \left(\sum_{w \in \mathrm{M}_i} T_{iw} \right)$$

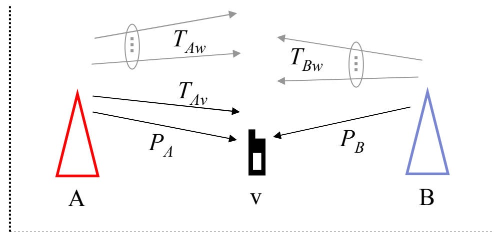
$$I_{other}^{pilot} = \eta \cdot \sum_{j \neq i} g_{jv} \cdot \left(P_j + \sum_{w \in \mathrm{M}_j} T_{iw} \right)$$

traffic signal *SINR*:

$$SINR_{iv}^{tr} = \frac{G_p^{tr} \cdot T_{iv} \cdot g_{iv}}{N_0 \cdot \mathrm{W} + I_{own}^{tr} + I_{other}^{tr}}$$

$$I_{own}^{pilot} = \varsigma \cdot g_{iv} \cdot \left(P_i + \sum_{w \neq v, w \in \mathrm{M}_i} T_{iw} \right)$$

$$I_{other}^{tr} = I_{other}^{pilot}$$



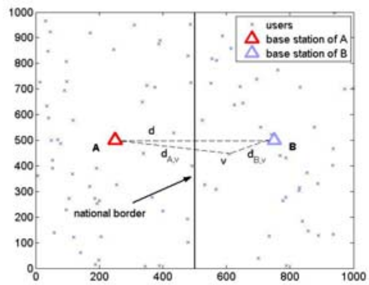
P_i – pilot power of i
G_p^{pilot} – processing gain for the pilot signal
g_{iv} – channel gain between BS i and user v
N_0 – noise energy per symbol
W – available bandwidth
I_{own}^{pilot} – own-cell interference affecting the pilot signal
ς – own-cell interference factor
T_{iv} – traffic power between BS i and user v
M_i – set of users attached to BS i
η – other-to-own-cell interference factor

Game-theoretic model

- Power Control Game, G_{PC}
  - players → networks operators (BSs), A and B
  - strategy → pilot signal power, 0W < P_i < 10W, i = {A, B}
  - standard power, P^S = 2W
  - payoff → profit, $u_i = \sum_{v \in \mathrm{M}_i} \theta_v$ where θ_v is the expected income serving user v
  - normalized payoff difference:

$$\Delta_i = \frac{\max_{s_i} \left(u_i \left(s_i, P^S \right) - u_i \left(P^S, P^S \right) \right)}{u_i \left(P^S, P^S \right)}$$

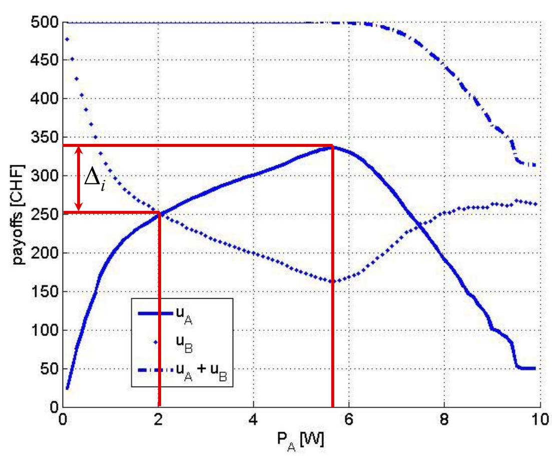
Simulation settings



Parameter	*Value*
simulation area size	1 km^2
BS positions	(250 m, 500 m) and (750 m, 500 m)
default distance between BSs, d	500 m
user distribution	random uniform
number of simulations	500
default path loss exponent, α	4
BS max power	43 dBm = 20 W
BS max load	40 dBm = 10 W
BS standard power, P^s	33 dBm = 2 W
BS min power	20 dBm = 0.1 W
power control step size, P_{step}	0.1 W
orthogonality factor, ζ	0.4
other-to-own-cell interference factor, η	0.4
user traffic types:	audio, $\mathbb{R}^{tr} = 12.2$ kbps
	video, $\mathbb{R}^{tr} = 144$ kbps
	$\boxed{\text{data, } \mathbb{R}^{tr} = 384 \text{ kbps}}$
required CIR (audio, video, data):	-20 dB, -12.8 dB, -9 dB
expected incomes ($\theta_{audio}, \theta_{video}, \theta_{data}$):	10, 20, 50 CHF/month

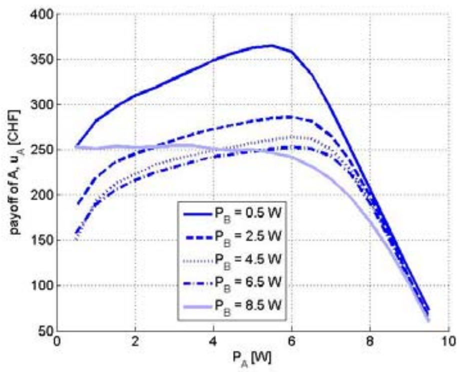
Is there a game?

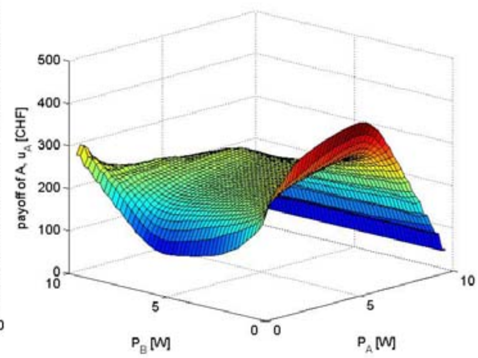
- only A is strategic (B uses $P_B = P^S$)
- 10 data users
- path loss exponent, $\alpha = 2$



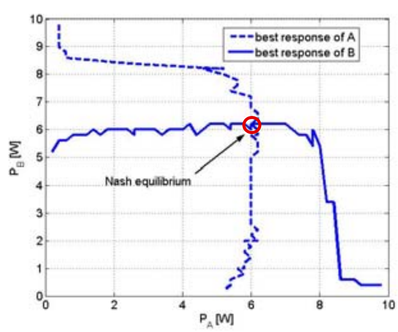
When both operators are strategic

- 10 data users
- path loss exponent, α = 4

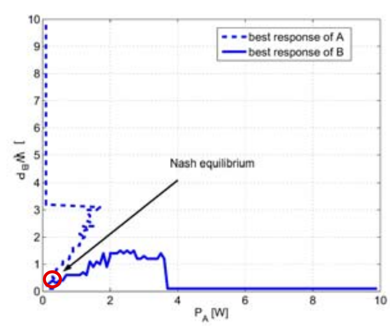




Nash equilibria



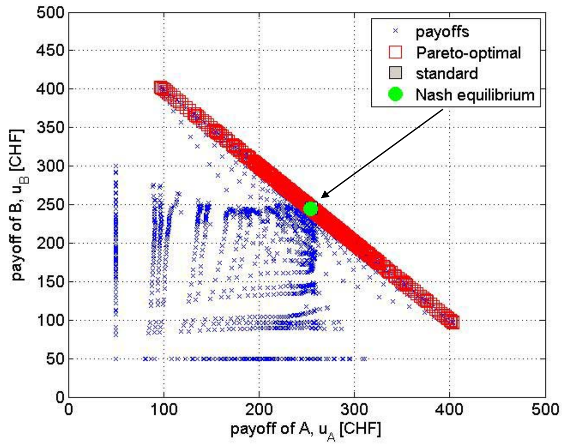
10 data users



100 data users

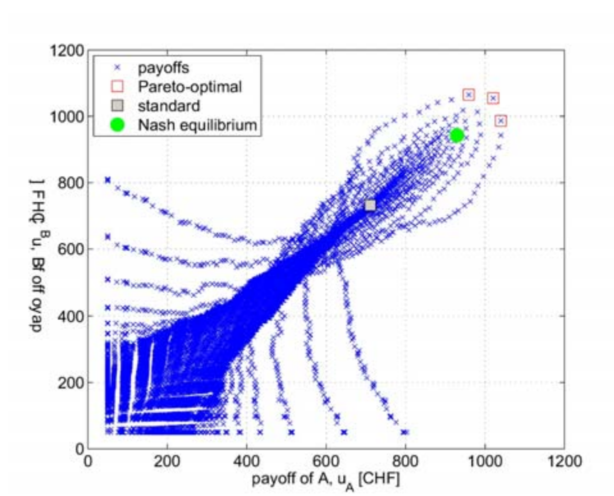
Efficiency (1/2)

- 10 data users



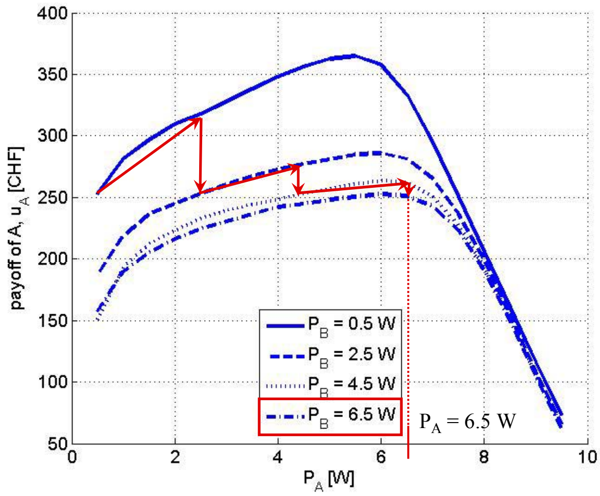
Efficiency (2/2)

- 100 data users



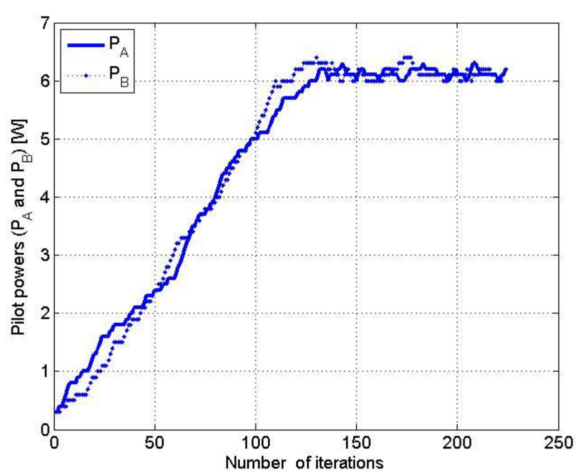
Convergence to NE (1/2)

- convergence based on better-response dynamics
- convergence step: 2 W



Convergence to NE (2/2)

- convergence step: 0.1 W

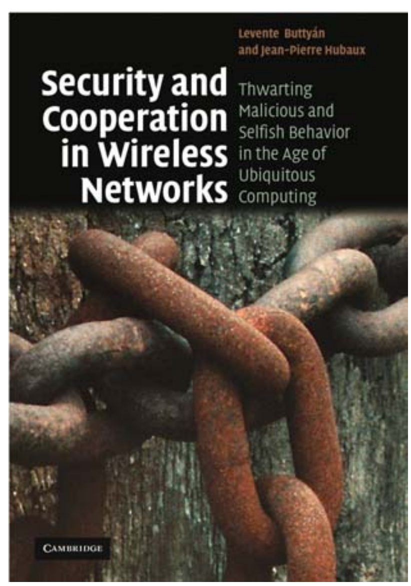


Conclusions

- not only individual nodes may exhibit selfish behavior, but operators can be selfish too
- example: adjusting pilot power to attract more users at national borders
- the problem can be modeled as a game between the operators
  - the game has an efficient Nash equilibrium
  - there's a simple convergence algorithm that drives the system into the Nash equilibrium

A textbook

- written by
  - Levente Buttyan (BME)
  - Jean-Pierre Hubaux (EPFL)
- intended to
  - graduate students
  - researchers and practitioners
- to be published by
  - Cambridge University Press
  - ISBN 9780521873710
- expected publication date
  - November 2007
- material available on-line at **secowinet.epfl.ch**
  - full manuscript in pdf
  - slides for each chapter (progressively)



Acknowledgements

Many thanks to

- Jean-Pierre Hubaux, co-author of the book "Security and Cooperation in Wireless Networks", for initiating and driving the book project, and for many slides of this tutorial

- Mark Felegyhazi, a good friend and colleague, for his major contributions to the part on "Thwarting selfish behavior"