Wireless Security: A Perspective

(aka. What We've Done Wrong, and Some of What We Can Do About It…)

Wade Trappe

Agenda

- Tales from the Dark Side of Security
  - Wireless in Particular
- Decomposing the Problem into Problems
  - Crypto was amateur
  - Performance is still important!
- What we can do about it?
  - Look at adversary models
  - Revisit wireless basics

Tales from the Dark Side of Security: Some Exploits

Generic examples of security flaws in real systems illustrates the challenge of getting security right

- Prepayment in Electricity Meter Systems:
  - Present a (purchased) digital token to a power meter.
  - Digital token would convey an ID so it could not be duplicated or forged…
  - Problem was that the rate information was not protected
- Bank Fraud:
  - A bank would allow customers to present a bank card which had a PIN code encrypted and stored on the magnetic strip
  - Teller had a copy of the encryption key and could check the PINs.
  - Flaw in design: adversary could alter the account number on the card to someone else's, while using his own PIN number… he would check out ok… but the money would be drawn from someone else's account!
  - Flaw in design: PIN number was not connected to account #.

Wireless systems have not faired any better in terms of security design

- Cellular Message Encryption Algorithm (CMEA) was deeply flawed
- 802.11 systems, when originally deployed:
  - Were shipped with security disabled
  - Offered SSID/MAC address filtering as security
  - WEP was seriously flawed
- Routing protocols are hard to get right
  - AODV is inherently insecure
  - Its secure variants (ARAN, SAODV) have not done much better
- The wireless medium is inherently more challenging
  - Eavesdropping is trivial and impossible to detect
  - Open, broadcast medium
    - *Jamming is possible*
- The wireless product space is more diverse
  - Highly programmable platforms available
  - Easy to create one's own device and use it

Cellular security algorithms were poorly designed, leading to numerous attacks

- The Telecommunications Industry Association proposed four cryptographic primitives for use in North America (1995, all are now considered weak):
  - CAVE: A mixing function used for authentication and key generation
  - XOR masking used for voice privacy
  - ORYX: an LFSR-based stream cipher
  - CMEA (Control Message Encryption Algorithm): a block cipher to encrypt control channel
- Consider CMEA:
  - CMEA is its own inverse (every key is a "weak key")
  - CMEA encrypts short blocks, but cellular telephony did not employ CFB, or random IVs→ codebook attacks are a threat (consider there are only 10 digits!)
  - LSB of plaintext is leaked
  - Internal T-box has skewed statistical distribution (reduces search space significantly)
  - Chosen-plaintext attack can succeed with 338 chosen plaintexts and very little work
  - Known plaintext attacks: 3-byte version succeeds with 80 known texts and ~2^{32} complexity; 2-byte attacks only need 4 known plaintexts (undermining IS-95)
- Compromise of control channel can lead to compromise of confidential information shared over control channel:
  - PIN numbers, credit card numbers, bank account information
  - Digits dialed by users might reveal user calling patterns

Early 802.11 used SSID/MAC address filtering, which could not achieve any security

- SSID:
  - AP periodically broadcasts SSID in a beacon.
  - End station listens to these broadcasts and chooses an AP to associate with based upon its SSID.
  - Use of SSID – weak form of security as beacon management frames on 802.11 WLAN are always sent in the clear.
  - A hacker can use analysis tools (eg. AirMagnet, Netstumbler, AiroPeek) to identify SSID.
  - Some vendors use default SSIDs which are pretty well known (eg. CISCO used tsunami)
- MAC Address Filtering: The system administrator can specify a list of MAC addresses that can communicate through an access point.
  - Increases Administrative overhead
  - Determined hackers can still break it by sniffing MAC addresses and spoofing MAC addresses

Early 802.11 proposed WEP to address security concerns, but design was inherently weak

- Designed to provide confidentiality to a wireless network similar to that of standard LANs.
- WEP is essentially the RC4 symmetric key cryptographic algorithm (same key for encrypting and decrypting).
  - Transmitting station concatenates 40 bit key with a 24 bit Initialization Vector (IV) to produce pseudorandom key stream.
  - Plaintext is XORed with the pseudorandom key stream to produce ciphertext.
  - Ciphertext is concatenated with IV and transmitted over the Wireless Medium.
  - Receiving station reads the IV, concatenates it with the secret key to produce local copy of the pseudorandom key stream.
  - Received ciphertext is XORed with the key stream generated to get back the plaintext.
- WEP has been broken! Walker (Oct 2000), Borisov et. al. (Jan 2001), Fluhrer-Mantin -Shamir (Aug 2001).
- Unsafe at any key size : Testing reveals WEP encapsulation remains insecure whether its key length is 1 bit or 1000 or any other size.

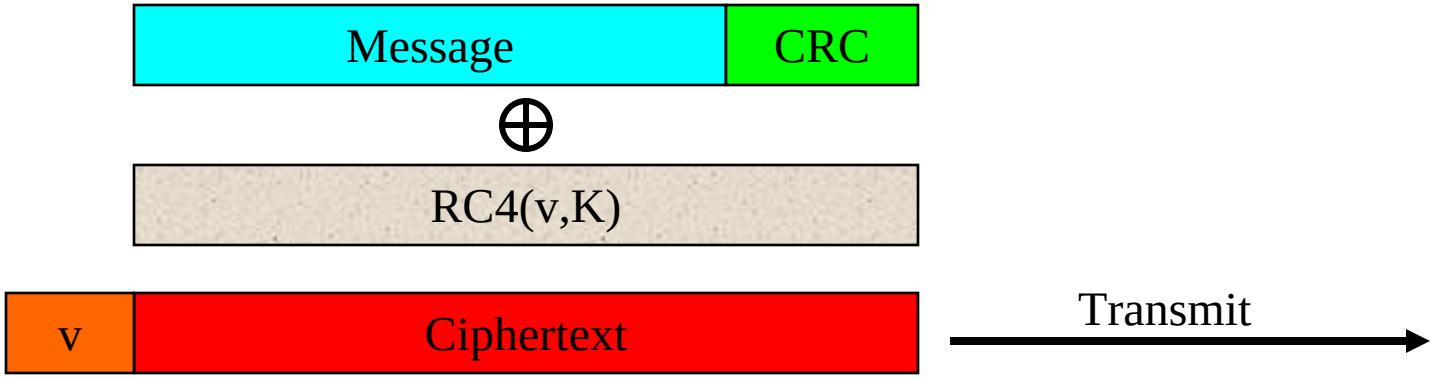
The basic WEP packet included checksums, RC4 and an IV field

- WEP relies on a shared key K between communicating parties

1. **Checksum:** For a message M, we calculate c(M). The plaintext is P={M,c(M)}
2. **Encryption:** The plaintext is encrypted using RC4. RC4 requires an initialization vector (IV) v, and the key K. Output is a stream of bits called the keystream. Encryption is XOR with P.

$$C = P \oplus RC4(v, K)$$

3. **Transmission:** The IV and the ciphertext C are transmitted.



WEP was intended to provide three main security goals so as to be "Equivalent" to wired security

- WEP had three main security goals:
  - Confidentiality: Prevent eavesdropping
  - Access Control: Prevent inappropriate use of 802.11 network, such as facilitate dropping of not-authorized packets
  - Data Integrity: Ensure that messages are not altered or tampered with in transit
- The basic WEP standard uses a 40-bit key (with 24bit IV)
- Additionally, many implementations allow for 104-bit key (with 24bit IV)
- None of the three goals are provided in WEP due to serious security design flaws and the fact that it is easy to eavesdrop on WLAN

A basic flaw in WEP was Vernam Key Stream Reuse

- Vernam-style stream ciphers are susceptible to attacks when same IV and key are reused:

$$C_1 = P_1 \oplus RC4(v, K)$$
$$C_2 = P_2 \oplus RC4(v, K)$$
$$C_1 \oplus C_2 = P_1 \oplus RC4(v, K) \oplus P_2 \oplus RC4(v, K)$$
$$= P_1 \oplus P_2$$

- Particularly weak to known plaintext attack: If P_1 is known, then P_2 is easy to find (as is RC4).
  - This might occur when contextual information gives P_1 (e.g. application-level or network-level information reveals information)
- Even so, there are techniques to recover P_1 and P_2 when just (P_1 XOR P_2) is known (frequency analysis, crib dragging)
  - Example, look for two texts that XOR to same value

Vernam key stream reuse was inadequately prevented in WEP's design

- WEP's engineers were aware (it seems??) of this weakness and required a per-packet IV strategy to vary key stream generation
- Problems:
  - Keys, K, typically stay fixed and so eventual reuse of IV means eventual repetition of keystream!!
  - IVs are transmitted in the clear, so its trivial to detect IV reuse
  - Many cards set IV to 0 at startup and increment IV sequentially from there
  - Even so, the IV is only 24 bits!
- Calculation: Suppose you send 1500 byte packets at 5Mbps, then 2^{24} possible IVs will be used up in 11.2 hours!
- Even worse: we should expect to see at least one collision after 5000 packets are sent!
- Thus, we will see the same IV again… and again…

A consequence of key stream reuse is that IV decryption dictionaries can be built

- Once a plaintext is known for an IV collision, the adversary can obtain the key stream for **that specific** IV!
- The adversary can gather the keystream for each IV collision he observes
  - As he does so, it becomes progressively easier to decrypt future messages (and he will get improved context information!)
  - The adversary can build a dictionary of (IV, keystream)
- This dictionary attack is effective regardless of keysize as it only depends on IV size!

WEP also failed to achieve its message authentication goals

- The checksum used by WEP is CRC-32, which is not a cryptographic checksum (MAC)
  - Purpose of checksum is to see if noise modified the message, not to prevent "malicious" and intelligent modifications
- Property of CRC: The checksum is a linear function of the message

$$c(x \oplus y) = c(x) \oplus c(y)$$

- This property allows one to make controlled modifications to a ciphertext without disrupting the checksum:
  - Suppose ciphertext C is:

$$C = RC4(v, K) \oplus \{M, c(M)\}$$

  - We can make a new ciphertext C' that corresponds to an M' of our choosing
  - Then we can spoof the source by: A→B: {v,C'}

WEP also failed to achieve its message authentication goals, pt. 2

- Our goal: Produce an M'=M+δ, and a corresponding checksum that will pass checksum test. (Hence, we will need to make a plaintext P'={M',c(M')} and a corresponding ciphertext C')
- Start by choosing our own δ value, and calculate checksum.
- Observe:

$$\begin{aligned} C' &= C \oplus \{\delta, c(\delta)\} \\ &= RC4(v,K) \oplus \{M, c(M)\} \oplus \{\delta, c(\delta)\} \\ &= RC4(v,K) \oplus \{M \oplus \delta, c(M) \oplus c(\delta)\} \\ &= RC4(v,K) \oplus \{M', c(M \oplus \delta)\} \\ &= RC4(v,K) \oplus \{M', c(M')\} \end{aligned}$$

- Thus, we have produced a new plaintext of our choosing and made a corresponding ciphertext C'
- Does not require knowledge of M, actually, we can choose δ to flip bits!

WEP allows for easy message injection, and does not provide any form of access control

- Property: The WEP checksum is an unkeyed function of the message.
- If attacker can obtain an entire plaintext corresponding to a frame, he will then be able to inject arbitrary traffic into the network (for same IV):

1. Get RC4(v,K)
2. For any message M' form $C' = RC4(v, K) \oplus \{M', c(M')\}$

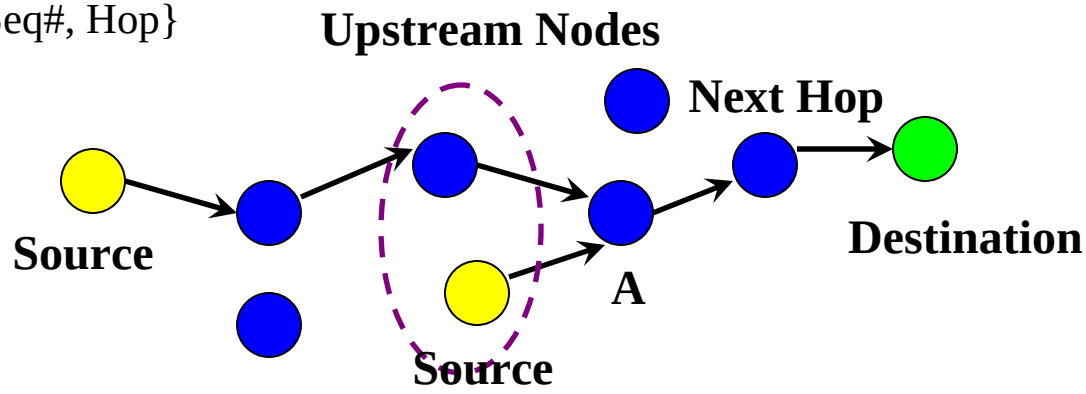
- Why did this work? c(M) only depended on M and not on any key!!!
- (Note: An adversary can easily masquerade as an AP since there are no mechanisms to prevent IV reuse at the AP-level!)

There are numerous other security flaws in 802.11, but some issues addressed in WPA2

- Another example is the Evil Twin AP:
  - An adversary installs a rogue AP near a regular AP.
  - This rogue AP may use the same SSID as the regular AP. It may or may not use the same MAC address as the regular AP. Rogue AP transmits with a stronger signal power.
    - *Clients automatically associate with rogue AP due to higher signal strength.*
    - *The rogue AP may drop all traffic from the clients that connect to it.*
  - Defenses:
    - *Perform network authentication. Requires the establishment of a known, shared key!!!*
    - *For open networks, always try both AP's and see which one provides service. Once a good AP is found, use signal strength as a consistency check.*
- Many issues can be addressed by employing WPA2
  - WPA2 addressed TKIP-legacy flaws in WPA
  - Includes pre-shared (home/office) key mode and EAP extensions for enterprise
- Major issue that remains is the protection of control frames (e.g. association frames)

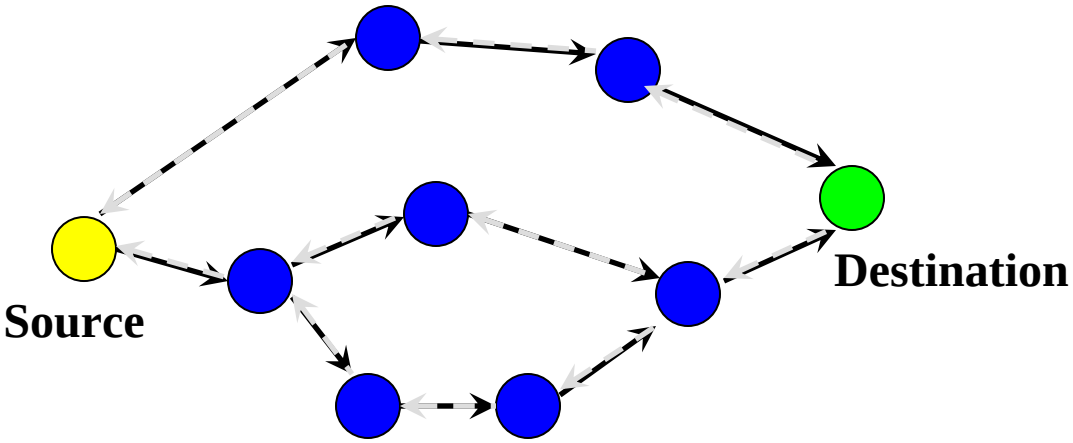
AODV is a popular routing scheme used in MANET and mesh networking

- Characteristics
  - Provides unicast, broadcast functionalities
  - Reactive: Initiate route discovery only on demand
  - Two dimensional routing metric: {Seq#, Hop}
- Routing Table
  - Destination Address, Seq Number
  - Next Hop, Hop Count
  - Lifetime
  - Upstream Nodes



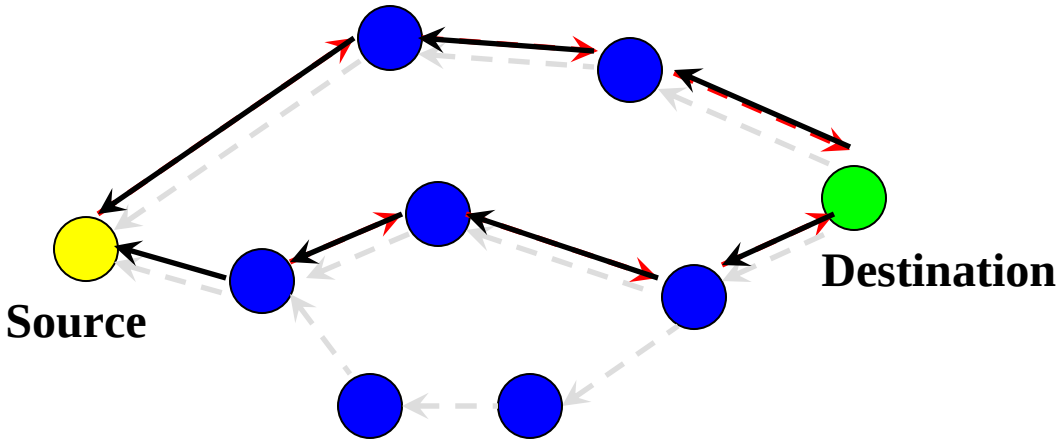
- Sequence Number
  - Created by each node to be included along with routing messages
  - Prevents routing loop, larger sequence number means a fresher route
  - A node increments its own sequence number when it
    - *Originates a route discovery*
    - *Originates a route reply in response to a route discovery request*
  - A node increments other node's sequence number by one only:
    - *In response to a lost or expired link to the next hop towards that destination*
- Hop Count
  - Smaller hop count means better route

The first stage in AODV route establishment is Route Discovery via RREQ to the destination



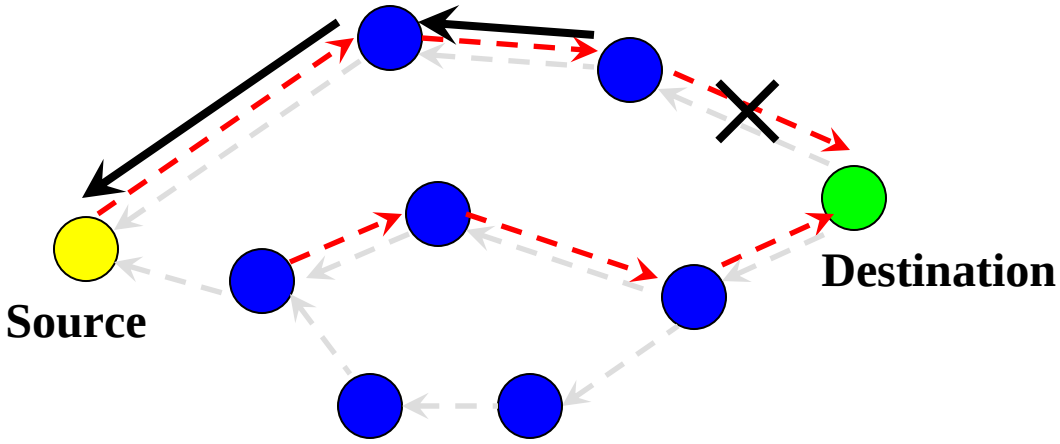
- Reverse Path Setup
  - Source broadcasts Route Request (RREQ)
  - RREQ: {S, D, ID, SrcNum, DstNum, Hop}
  - Nodes keep track of {S, ID}, discard redundant RREQs
  - Intermediate nodes update their routing entries toward Source
  - Intermediate nodes update DstNum if they have a higher copy
  - A node can reply to a RREQ if
    - *It is the destination*
    - *Has a fresh enough route to the destination*
  - Otherwise, increase the hop count and forward the RREQ

The second stage in AODV route establishment is Route Establishment via RREP to the source



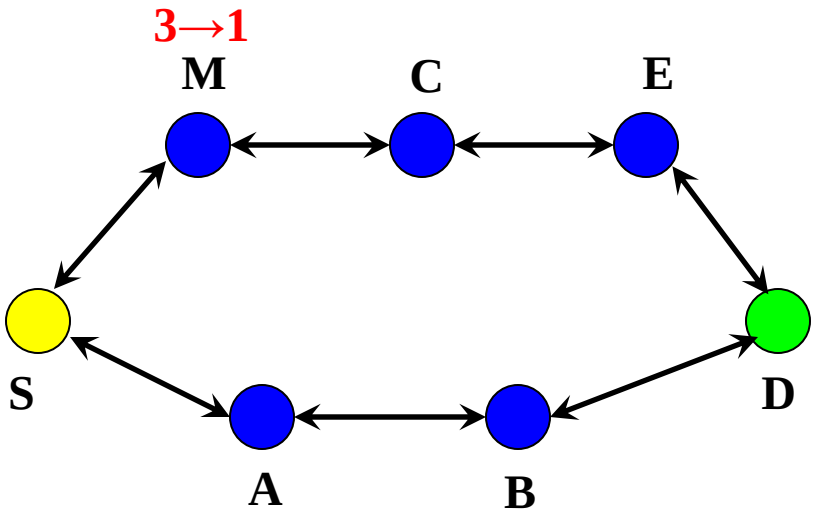
- Forward Path Setup
  - Destination or intermediate nodes with “fresh enough” route to Destination unicasts Route Reply (RREP) back to Source
  - RREP: {S, D, DstNum, Hop, Lifetime}
  - Intermediate nodes update the routing entry toward destination if it is a better or newer route, use reverse path to forward RREP
  - Intermediate nodes increase the hop count and forward the RREP

AODV route maintenance involves returning RERR messages to the source when a link breaks



- Route Maintenance
  - Node multicasts Route Error (RERR) to upstream nodes when
    - *It detects a link break for the next hop while transmitting data*
    - *It gets a data packet destined to a node for which it does not have an active route and is not repairing*
    - *It receives a RERR from a neighbor for one or more active routes.*
  - RERR: {D–List, DstNum–List}

AODV is susceptible to a variety of attacks against its basic maintenance functions



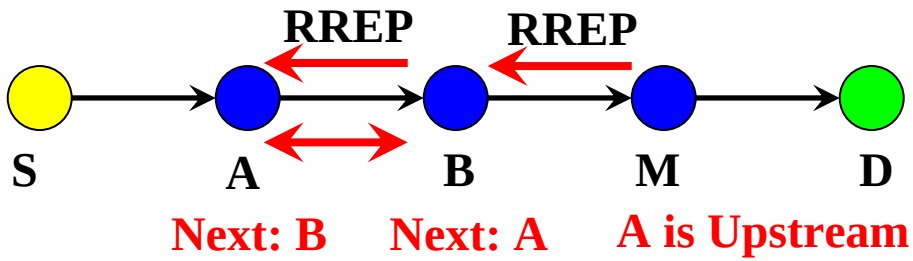
- Attacks on AODV
  - Forge RREQs/RREPs/RERRs on behalf of other nodes
  - Reduce the hop count in RREQs/RREPs
  - Increase the originator sequence number in RREQs
  - Increase the destination sequence number in RREPs
  - Selectively forward/reply RREQs, RREPs, and RERRs
  - Wormhole Attacks
- Securing AODV needs Authentication
  - End to end Authentication is not enough (how can intermediate nodes verify?)

Secure AODV (SAODV) uses public key cryptography to authenticate RREQ/RREP/RERR

- Zapata et. al. proposed SAODV in 2001
- Characteristics
  - Authenticate RREQs/RREPs/RERRs
  - Based on public key cryptography
- Assumption
  - Each node has a pair of public/private keys
- Security Extension
  - Intermediate nodes will not replace the destination sequence number in RREQ even if it has a newer copy
  - Single Signature Extension
  - Double Signature Extension
  - Hop count is protected through the use of a one-way hash function
- Details
  - Signature and hash functions are added to routing messages
  - Intermediate nodes verify signature and hashes before forwarding
  - Apply hash function to get next hop value

SAODV is nonetheless susceptible to a variety of simple attacks

- Problems remain in SAODV
  - Same hop count fraud
    - *Malicious nodes can still choose NOT to increase hop count*
  - Signature DoS
    - *Malicious nodes can flood a large amount of bogus traffic*
  - Formation of routing loops
    - *Intermediate nodes do not authenticate previous hop*

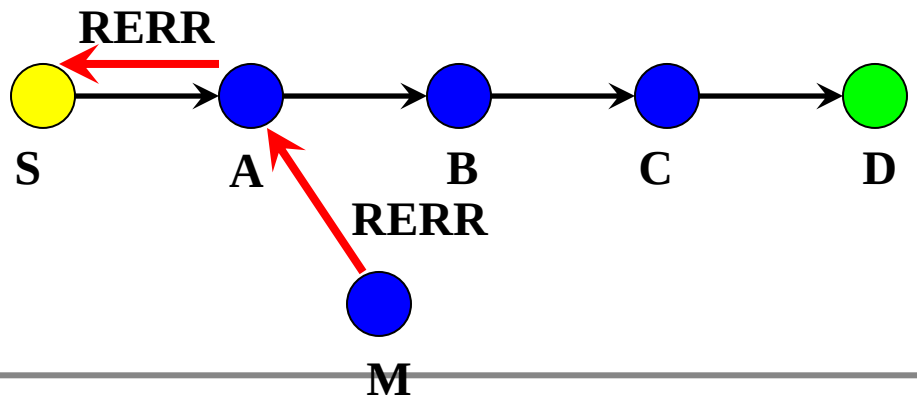


Authenticated Routing for Ad hoc Networks (ARAN) also authenticates RREQ/RREP/RERR

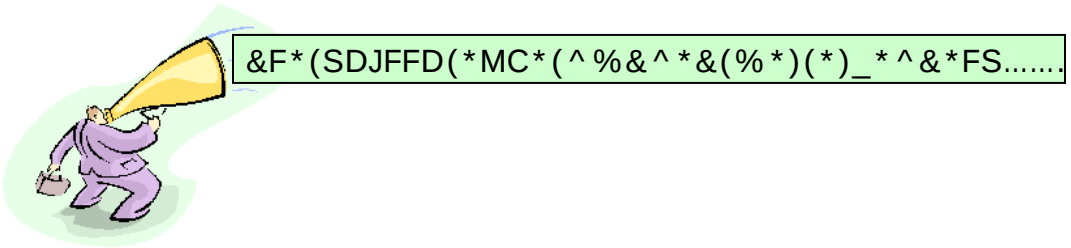
- Sanzgiri proposed ARAN in 2002
- Characteristics
  - Authenticate RREQs/RREPs/RERRs
  - Based on public key cryptography
- Assumption
  - Each node has a pair of public/private keys
- Security Extension
  - Use nonce plus timestamp to provide the freshness of a route
  - Nonce and timestamp stay unchanged during propagation
  - Does not use hop count to select optimal path, use first received RREQ
  - RERR is forwarded along the way without modification
- Details
  - Signature and certificate are added to routing messages
  - First hop nodes verify the signature and add their own signature and certificate to routing message
  - Other nodes verify the signatures and replace the previous hop signature and certificate with its own

ARAN is also susceptible to simple attacks targeted at exploiting routing functionality

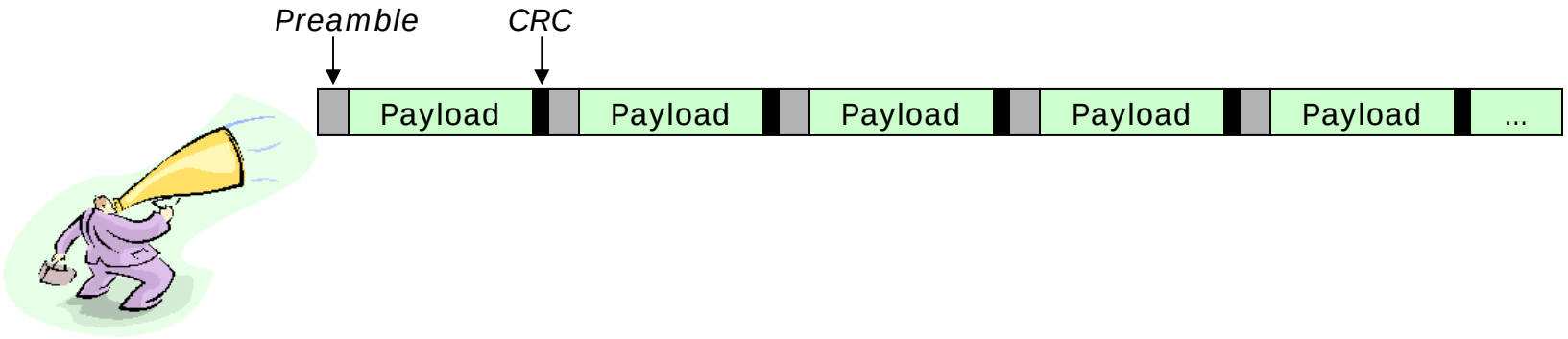
- Problems
  - Same hop count fraud
    - *Malicious nodes can still choose NOT to replace the signature and certificate of previous hop*
  - Signature DoS
    - *Malicious nodes can flood a large amount of bogus traffic*
  - Error spoofing attack
    - *Only authenticate the original source of the RERR*



Wireless networks are susceptible to interference attacks that target layer 1 and 2

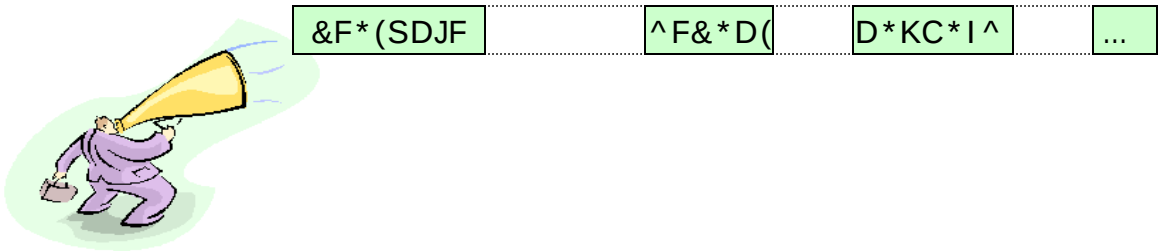


- Constant jammer:
  - Continuously emits a radio signal

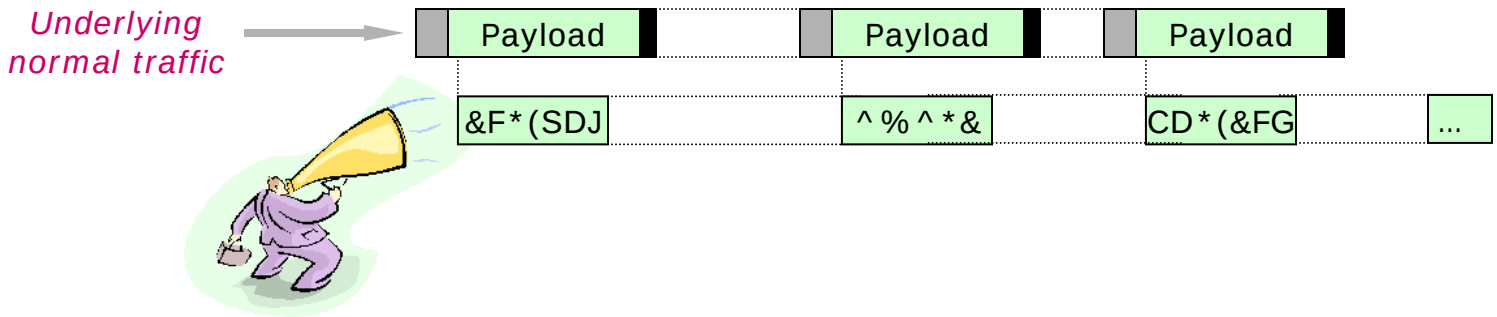


- Deceptive jammer:
  - Constantly injects regular packets to the channel without any gap between consecutive packet transmissions
  - A normal communicator will be deceived into the receive state

A variety of L2 jammer models have been proposed, including a powerful reactive jammer



- Random jammer:
  - Alternates between sleeping and jamming
    - *Sleeping period: turn off the radio*
    - *Jamming period: either a constant jammer or deceptive jammer*



- Reactive jammer:
  - Stays quiet when the channel is idle, starts transmitting a radio signal as soon as it senses activity on the channel.
  - Targets the reception of a message

Decomposing the Problem into Many Problems and What We Can Do About It…

Caveat Cryptor: Designer Beware!

- The lesson learned from these stories:
  - The adversary can be very powerful and clever!
  - Engineers make cruddy security analysts
  - There is a body of knowledge in the crypto community that never makes it to the engineering world
- We must assume that the adversary has complete control over the network…
  - Be paranoid! Alice should not blindly trust what she is getting from “Bob”! And vice-versa!
  - If we can build a system that we trust in this **Seriously Caustic** environment, then we can trust it in more general (day-to-day) computing scenarios

History has shown that security is not an easy task, and we must stand on the giants before us

- Building secure systems and protocols is not easy.
- In general, its not an easy matter to prove that some protocol is indefinitely secure.
  - Denning-Sacco protocol took 12 years for a protocol failure to be exposed
  - Needham-Schroeder survived for 17 years before a man-in-the-middle attack was found
- Attacks of today must always be considered when building systems
  - Attacks of tomorrow aren't known yet…
  - That's the challenge!
- What can we do?
  - Formal verification logics?
  - Basic design guidelines?
  - Teach people to be attackers and defenders?
  - Read Schneier and Ferguson?

We need to revisit Needham's security design guidelines

- Needham has given several guidelines for building secure systems

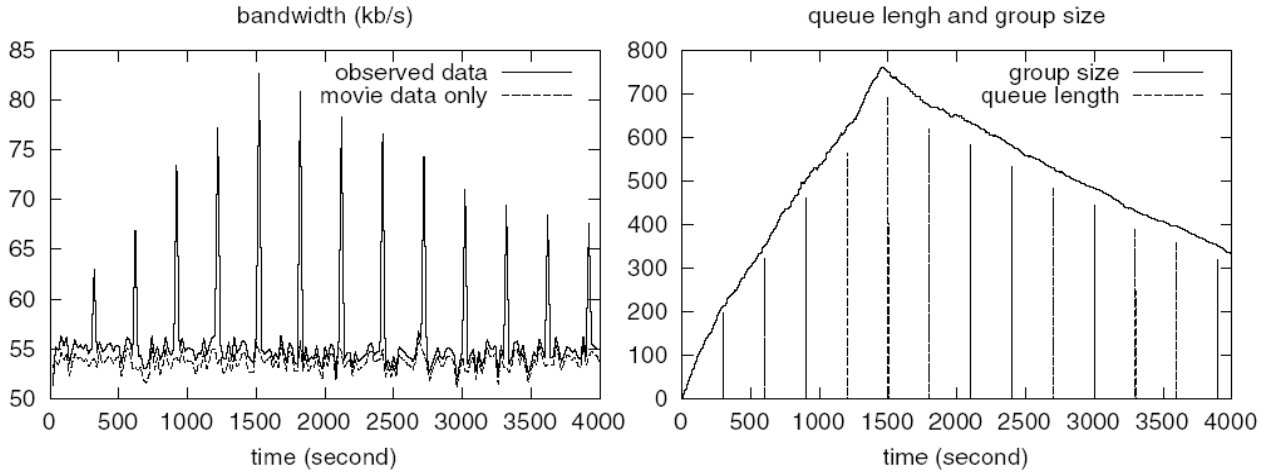
1. Be clear of security goals and assumptions
2. When using encryption, know why you are using it (secrecy? Authenticity? Binding? PRNG?) . Encryption is not security!
3. Be careful about temporal associations
4. Don't assume the identity of a participant can be excluded from a message. Generally, it should be explicitly included in a message!
5. Have redundancy in your message!
6. Know the properties and weaknesses of the cryptographic protocols you are using.
7. Signatures do not imply that the signer knows what the message is that he is signing!
8. Don't trust others to keep their secrets secret!
9. When responding to queries, be careful about encrypting, decrypting, or signing. You might be used as an oracle by an adversary!
10. Decryption is not the same as digital signatures- they have different purposes!
11. Distinguish between different runs of the protocol!

There are many other considerations that somehow must be factored into problem!!!

- KISS (Keep It Simple Stupid) is often desirable from an engineering point of view…
  - Its generally **BAD** from a security point of view!
  - Removing some data fields because they seem like they can be inferred (and thus shorten the message) can result in severe protocol failures!
- However, we still have to consider some key performance issues as we may destroy the very system we aim to protect!
- Deny by default might be desirable, but impractical
- It is not all about crypto– there are many “wireless” problems that can be exploited

3GPP tried to secure MBMS but forgot to address basic network performance issues

- 3G Multimedia Broadcast/Multicast Service provides media to a group of users
  - Qualcomm S3-030040 3GPP proposal sought to "Make the Session Key change so frequently that the cost of attacking is more expensive than the cost of subscribing to the service"
- Bandwidth: network resources will be wasted on sending out SK_RAND.
  - *SK_RAND has to be appended to each package.*
  - *For higher level of security, SK_RAND has to be large.*
- BAK update problem: at the moment that a new BAK is used, every USIM will send out a BAK request to BMSC
  - *BAK implosion problem*
  - *High peak bandwidth*



We need to train wireless security experts that understand security fundamentals and wireless

- Many of mistakes arise from lack of awareness of the strengths and weaknesses of the tools they use
  - Textbook crypto is generally weak
    - *Often a weak confidentiality model, in which the enemy is a* passive *eavesdropper, is used*
    - *We should consider an active adversary– they may modify a ciphertext or calculate a plaintext and send the result to a user to get an oracle service*
  - Adversary models are generally too simplistic
    - *Adversaries are often too localized*
    - *Adversaries are often too constrained in their resources (e.g. antennas)*
    - *Adversaries are often only outsiders… security people don't trust friends!*
- There are many tools that remain to be developed or to migrate into mainstream use
  - ID-crypto is just beginning to become popular
  - Trusted platform modules are generally relegated to the DRM community and need to be employed in a broader range of scenarios
  - Many tools (e.g. formal methods) may be helpful, but are at still at a young stage of development
- Let us look at some examples things we should teach…

Dolev-Yao represents the basic "omnipresent" adversary model in distributed systems

- For distributed systems and networks, we often should assume that there are adversaries
  - Everywhere in the network
  - Adversary may: eavesdrop, manipulate, inject, alter, duplicate, reroute, etc…
  - Adversary may control a large number of network nodes that are geographically separated
- Dolev-Yao Threat Model:
  - A very powerful adversarial model that is widely accepted as the standard by which cryptographic protocols should be evaluated
  - Eve, the adversary, can:
    - *Obtain <u>any</u> message passing through the network*
    - *Act as a legitimate user of the network (i.e. can initiate a conversation with any other user)*
    - *Can become the receiver to any sender*
    - *Can send messages to any entity by impersonating any other entity*

The Dolev-Yao model is not all-powerful, but assumes the existence of good crypto

- This seems very powerful, but not entirely so…
- Under Dolev-Yao:
  - Any message sent via the network is considered to have been sent by Eve
  - Thus, any message received "might" have been manipulated by Eve
  - Eve can control how things are sent
- What is not possible:
  - Eve cannot guess a random number which is chosen as part of a security protocol
  - Without knowledge of a key, Eve cannot figure out a plaintext from a ciphertext, nor can she create ciphertexts from a plaintext.
  - Eve can't solve the private-key pairing of a public key
  - Eve cannot control the "memory" of a computing device of a legitimate user (i.e. Eve can only play with the communication)

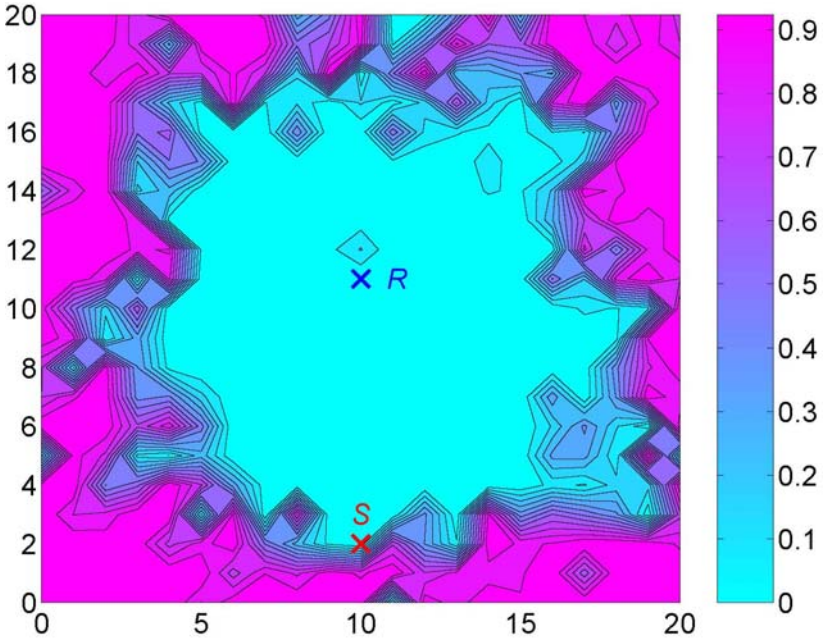
Non-Malleability is a cryptographic attack model that has caused problems in many systems

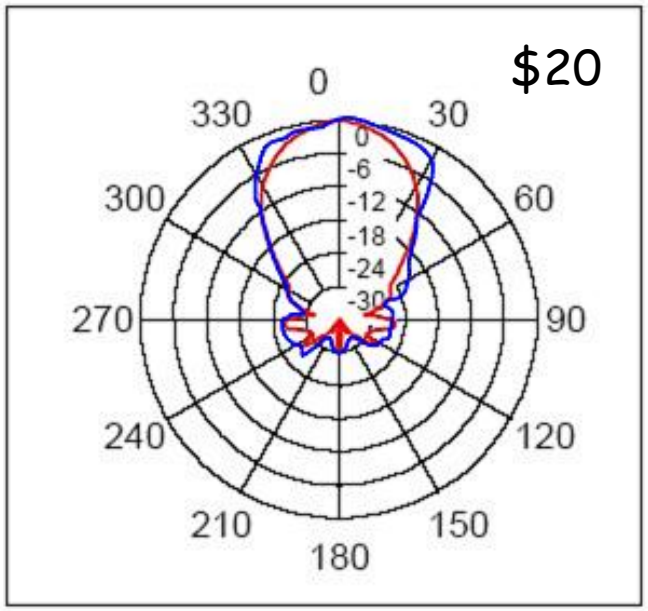
- Non-malleable cryptography: it should not be possible for Eve to modify a plaintext in a meaningfully controllable manner via modifying the ciphertext
- We have seen such a problem before:
  - One-time pads and Vernam ciphers: it is possible to modify select bits
  - We saw this type of weakness in WEP
- In a malleability attack, Eve's objective is, given a ciphertext C, not to learn something about the plaintext M, but instead to wreak havoc upon the eventual decoding
  - Eve needs to create a relationship C→ C' that results in a meaningful relationship M→ M'
- Problem: Most conventional cryptographic algorithms are the result of trapdoor functions
  - Partial information oracles exist for these public key schemes (e.g. the math that lets one learn parity can be the basis for conducting a malleability attack)
- Example: How to double a plaintext in RSA encryption
  - Take $C=M^e \bmod N$ and then produce $C' = C2^e \bmod N$

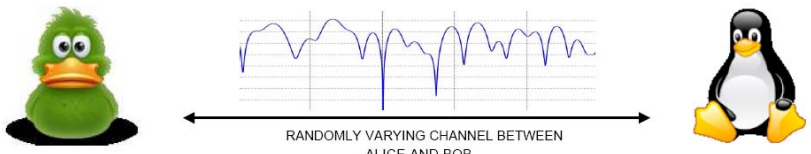
Byzantine threats involve weaknesses from the inside, which capture a key source of threats

- The Dolev-Yao model is the foundation of security analysis for active adversary scenarios, but does not capture everything that an adversary can do:
  - It does not involve entity compromise
- In situations involving many participants (e.g. distributed computing or peer-to-peer), it is natural to ask what can happen if a legitimate entity becomes compromised
- A Byzantine failure is one where a node/entity fails to operate properly, but continues to operate (as opposed to fail-stop failures)
- Example Byzantine Failures:
  - A node may lie about connectivity
  - Flood network with false traffic
  - Falsely describe opinions of another node (e.g. P2P)
  - Capture a strategic subset of devices and collude

On the wireless front, we need to be aware of radio irregularities, antennas and propagation







Alice

Bob

Eve