# Security Challenges in Wireless Networks

## Wireless Architecture and Challenges

- Cellular Networks
- Wireless LAN (Wifi)
- Wireless Mesh Networks
- Wireless Sensor Networks
- Vehicular Networks
- Personal Area Networks

## Security Services - Wireless Network

Authentication
- The most fundamental security service which ensures, that an entity has in fact the identity it claims to have

Integrity
- In some kind, the “small brother” of the authentication service, as it ensures, that data created by specific entities may not be modified without detection

Confidentiality
- The most popular security service, ensuring the secrecy of protected data

Access Control
- Controls that each identity accesses only those services and information it is entitled to

Non Repudiation
- Protects against that entities participating in a communication exchange can later falsely deny that the exchange occurred

## Wireless networks faces all threats that does its wired counterpart

- Masquerade, eavesdropping, authorization violation, loss or modification of transmitted information, repudiation of communication acts, forgery of information, sabotage
- Thus, similar measures like in fixed networks have to be taken

## What is different

- Wireless Network is more accessible for eavesdropping
- The lack of a physical connection makes it easier to access services
- Authentication has to be re-established when the mobile device moves
- Key management gets harder as peer identities can not be pre-determined
- The location of a device / user becomes a more important information that is worthwhile to eavesdrop on and thus to protect
- Injecting bogus messages into the network is easy
- Replaying previously recorded messages is easy
- Illegitimate access to the network and its services is easy

- Denial of service is easily achieved by jamming

## Cellular Network Security

2G had weak security
- Possible attacks from a faked base station
- Cipher keys and authentication data transmitted in clear between and within networks
- Encryption not used in some networks -> open to fraud
- Data integrity not provided

Some improvement with respect to 2nd generation
- Cryptographic algorithms are published
- Integrity of the signaling messages is protected

Cellular Security not a focus but may explore a bit more

## Wifi - WLAN

- Wireless host communicates with base station (i.e. access point(AP))
- Basic Service Set (BSS) in infrastructure mode contains:
    Wireless hosts
    Access point (AP): base station
    Ad hoc mode: hosts only (点对点)

## Wireless mesh network (WMN)

Features: Mesh routers; Multi-hop routing

## WMN Security

Several verification need to be performed:
- WAP (connected to internet) has to authenticate the user terminal
- Each user has also to authenticate the next hop mesh router
- Each mesh router has to authenticate the other mesh routers in the WMN
- The data sent or received by user has to be protected (e.g. to ensure data integrity, non-repudiation and / or confidentiality).
- Denial of service attack possible

Performing these verification has to be efficient and lightweight, especially for the user terminal.

## Sensor Networks

Large number of sensor nodes, a few base stations.
Sensors are usually battery powered:
- Main design criteria: reduce the energy consumption

Multi-hop communication reduces energy consumption:
- Overall energy consumption can be reduced, if packets are sent in several smaller hops instead

of one long hop
- Fewer re-transmissions are needed due to collisions

## Sensor Network Security

Resource constraint
- Limited CPU processing power
- Limited Battery - attacker can deplete
- Need lightweight crypto protocols
Physical Security
- Capture, Cloning, and Tampering easy.
Wireless Programming on Devices possible
- Additional security risk

## Vehicular Ad hoc Network (VANET)

Communication: typically over the Dedicated Short Range Communication (DSRC) (5.9 GHz)
Example of protocol: IEEE 802.11p

## 802.15: Personal Area Network

Less than 10 m diameter
Replacement for cables (mouse, keyboard, headphones)
Ad hoc: no infrastructure
Master / slaves:
- slaves request permission to send (to master)
- master grants requests
802.15: evolved from Bluetooth specification
- 2.4 - 2.5 GHz radio band
- up to 721 kbps

## PAN Security (个人局域网)

Short-range communications, master-slave principle
Eavesdropping is difficult:
- Frequency hopping
- Communication is over a few meters only
Security issues:
- Authentication of the devices to each other
- Confidential channel
  - Based on secret link key

## IOT Devices and Security

The market for wearable wireless sensors is projected to grow to more than 420 million devices by 2014.
Fundamental applications in patient monitoring, personalized health-care, telemedicine, and athlete training.
Security is critical because these devices generate medical data, and challenging given that hey have low power and computation capabilities.

---

# Security in Internet of Things (IoT)

## History of Wireless Sensor Net

1999: Kahn, Katz, Pister: Vision for Smart Dust
2002 Sensys CFP: Wireless Sensor Network research as being composed of “distributed systems of numerous smart sensors and actuators connecting computational capabilities to the physical world have the potential to revolutionize a wide array of application areas by providing an unprecedented density and fidelity of instrumentation”.

## History (IoT)

Early 90s or prior: SCADA systems, Telemetry applications
Late 90s - Products / services from mobile operators (Siemens) to connect devices via cellular network - Primarily automotive telematics
Mid 2000s - Many tailored products for logistics, fleet management, car sfety, healthcare, and smart metering of electricity consumption
Since 2010 - A large number of consortium mushrooming up to bid for a large market share
- ABI Projects US$198M by 2018
- Berg Insight US$187M by 2014

## IoT Enablers

- All IP Network
- Location Positioning Systems (GPS)
- Plethora of devices from sophisticated Wireless Sensor Actuators, tracking and tracing devices, RFID & CLOUD storage
- Ease of programming - Embedded Java
- Software and Service Architectures
- Data Analytic - Big Data (Sensor feed)

## Research Challenges

Heterogeneity, Interoperability, Scalability, Affordable Coverage, Software Architecture /

Middleware, Security and Trustworthiness, Privacy, Big Data - Data Analytic

## Heterogeneity: Standard, Hardware, Platforms

- Bluetooth Low Energy (BLE)
- 6LoWPAN
- LORA
- MQTT
- LTE Cat0
- IEEE 802.15.4
- Internet 0
- RFID
- Sigfox
- Smartdust
- Tera-play
- Xbee
- Z-Wave

Table I
CROSS-SECTION OF CURRENT MOTE PLATFORM SPECIFICATIONS

| Device | MCU | Word Size | Clock |
|---|---|---|---|
| Imote 2 [12] | Intel PXA271 | 32 bit | 104 MHz |
| INGA [13] | ATmega 1284p | 8 bit | 8 MHz |
| Mulle v5.2 [14] | Renesas M16C/62P | 16 bit | 10 MHz |
| SunSPOT v6 [15] | AT91SAM9G20 | 32 bit | 400 MHz |
| TelosB [16] | TI MSP430F1611 | 16 bit | 4 MHz |
| XM1000 [17] | TI MSP430F2618 | 16 bit | 8 MHz |

- Arduino
- Contiki
- Electric Imp
- Gadgeteer
- ioBridge
- Raspberry Pi
- SensorTag
- TinyOS
- Wiring
- Xively
- .....

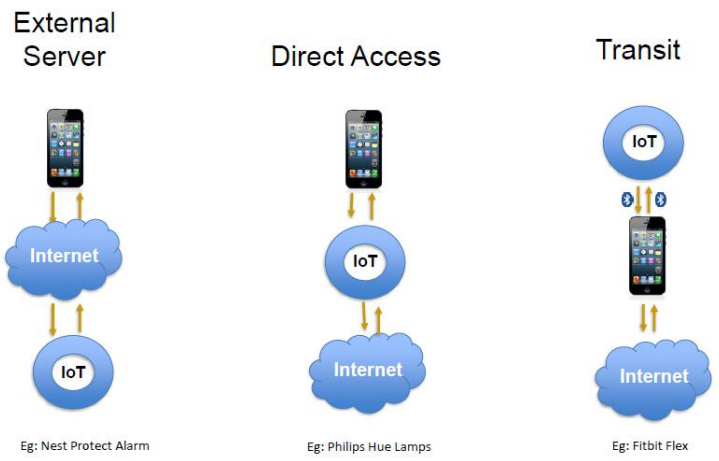
## Interoperability

Devices and Software Services Function speak different language
Trivial Example
- Milk in the fridge going off - Sensor on carton
- Sends TXT “drink milk” - Mobile Phone
- Smart Phone adds milk to shopping list.
Milk carton, fridge, phone, app - all must communicate with each other seamless

## Why IoT Security is Challenging?

Diversity of devices, capabilities, standards, programming environment
Lots of these devices store personal data - e.g. fitness / health, home security...
Application driven field - innovation comes from non-tech people
- First to market may mean no / little security
Potential threats / attacks a guess work

## Typical Operational Models



## Attacks

Wireless medium vulnerable to threats
- An intruder can gain access to WBAN
- Tamper the physiological data
- Inject false commands

## Authentication / Data Origin

Link Fingerprint - RSSI or phase unique between sensor device and a base-station
- Hard to forge, provable session record

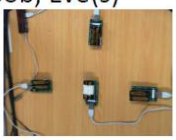
Use standard security primitives (PKI) to secure this signature and data - as needed by applications.

Optimize the fingerprinting process to reduce memory and transmission overheads

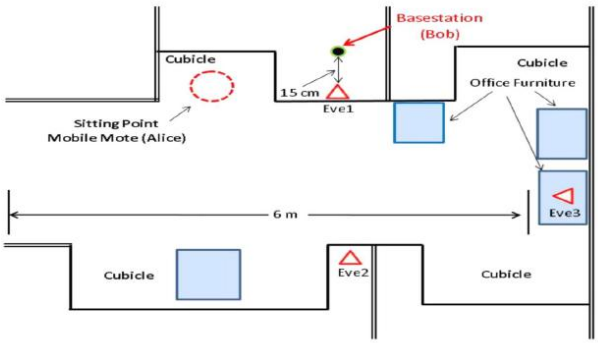
## Experimental Setup

¤ Bodyworn Device –Alice (MicaZ mote)

¤ Basestation – Bob, Eve(s)



¤ Indoor Office Environment




## RSSI Correlation (接受信号强度的变异)

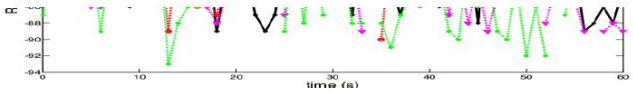
¤ Variation in RSSI vs. time

Alice and Bob

Bob and Eves

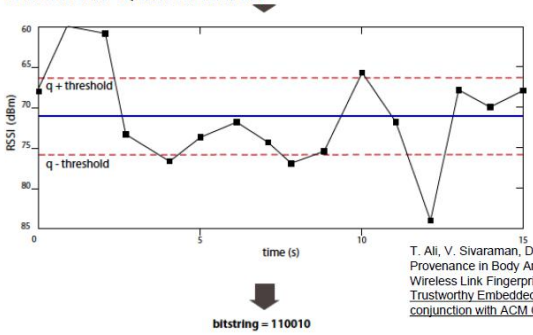
Table 1: Correlation coefficient ($r$) of RSSI measurements observed by various parties

| Experiment | Alice-Bob ($r$) | Alice-Eve1 | Alice-Eve2 | Alice-Eve3 |
|---|---|---|---|---|
| *High Activity* | 0.974 | 0.197 | 0.088 | 0.038 |
| *Low Activity* | 0.950 | 0.129 | 0.102 | 0.158 |
| *High Activity* (filtered) | 0.986 | 0.281 | 0.118 | 0.065 |
| *Low Activity* (filtered) | 0.976 | 0.205 | 0.152 | 0.224 |



## Memory Overhead

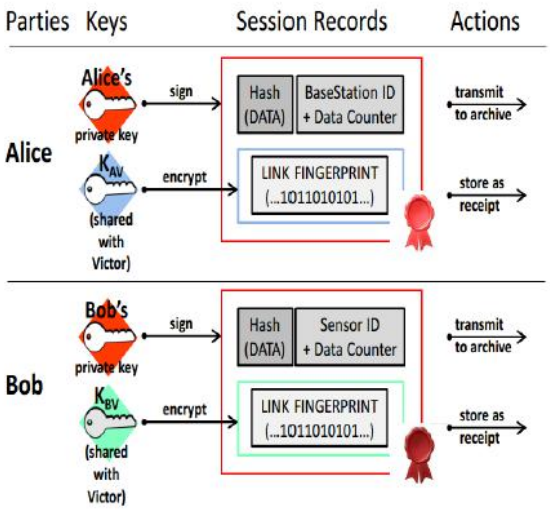
Store RSSI for every transactions – Memory overhead?
Solution: Quantization



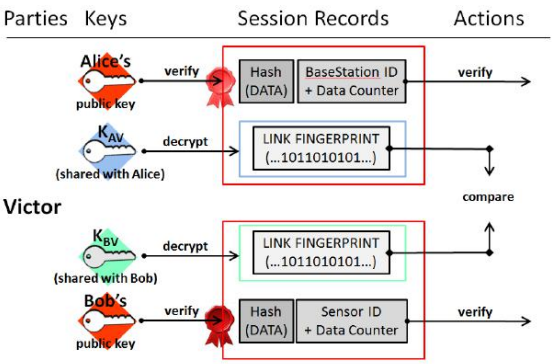

T. Ali, V. Sivaraman, D. Ostry and S. **Jha**, "Securing Data Provenance in Body Area Networks using Lightweight Wireless Link Fingerprints", International Workshop on Trustworthy Embedded Devices (TrustED 2013) held in conjunction with ACM CCS'13, November 4, Berlin, 2013

Figure 5: Level crossing quantization technique

## Authentication Protocol




## Verification Protocol



## Thread Group (ARM, Consortium of Qualcomm, and Samsung....)

Adopts PKC for authentication (Public Key Cryptosystems 公钥密码系统)
AES Symmetric key for confidentiality (Advanced Encryption Standard 高级加密标准)
IPv6 Low-power Wireless Personal Area Networks (6LoWPAN) to minimize the energy consumption from wireless communications
How to build secure-over-air reprogramming for IoT Devices (heterogeneous) ?

## Broadcast Security - for IoT

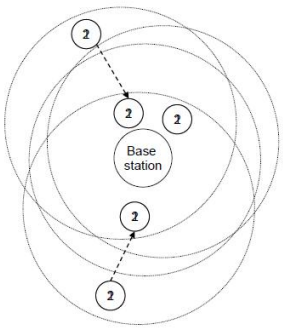
Broadcast applications need security
- Packet injection or eavesdropping is easy
Security solutions for point-to-point communication not scalable for large deployments
Broadcast challenges
- Scale to large audiences
- Dynamic membership
- Low overhead (computation & communication)
- Packet loss

How to achieve reliability in broadcasts?

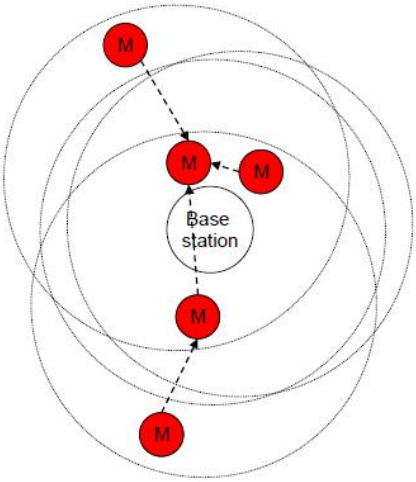
## WSN Code Dissemination

Assumes Homogeneous Sensor Network
Epidemic Communication Model
Exploit spatial multiplexing
- Parallel transmission in various parts of the network
Node with the newer version program image becomes a sender and a node with an older version becomes the receiver
Employ techniques: digital signature, Merkle hash tree, one-way has functions, pairwise encryption.

## WSN Multi-hop code distribution



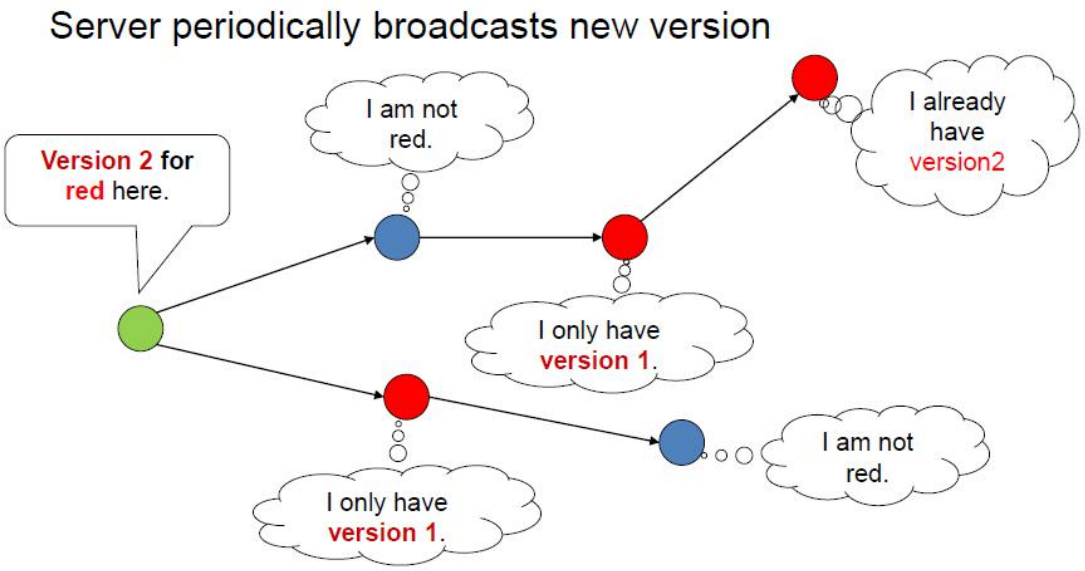
## WSN Secure Network Programming



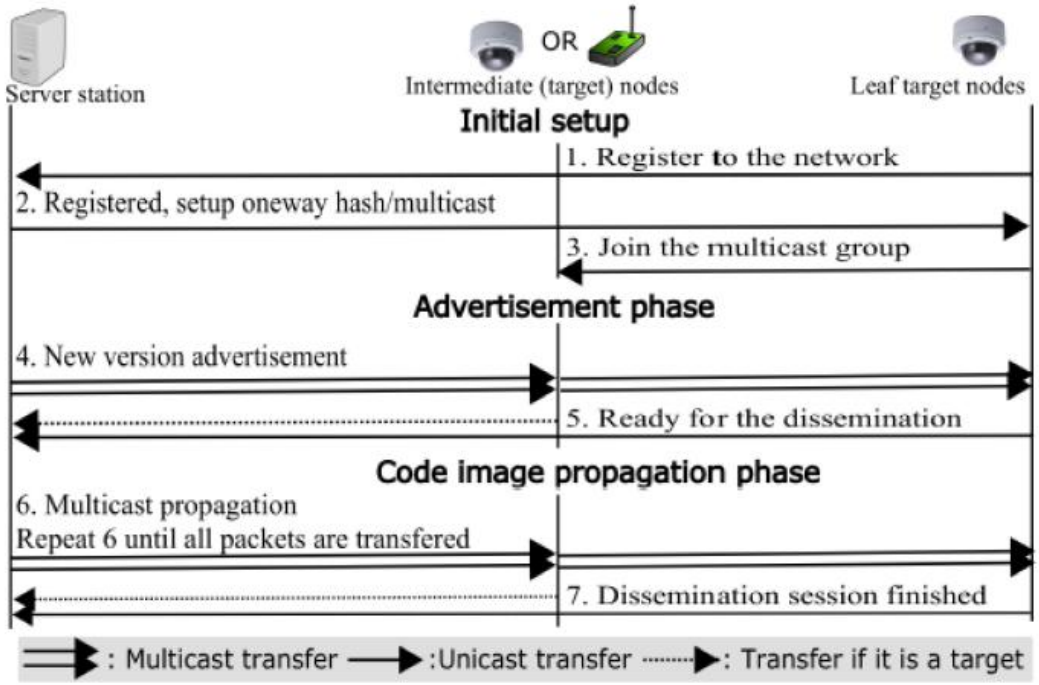

## SEDA: SEcure Over the air code Dissemination Architecture

- Motivation: To produce experimental system which servers as a guideline for future deployment
- Use overlay multicast communication model for efficient dissemination and key distribution
- Public key cryptographic broadcast encryption scheme (BGWt) - for efficient group key distribution / management, and low decryption overhead.
- Identify potential security threats and defensive measures
- Experimentally validate the architecture and provide performance benchmark

## Broadcast propagation



## SEDA Protocol Overview




## Research Contributions

The selection and implementation of a public key cryptographic broadcast encryption scheme e.g. variation of BGW

Experimentally validate through a prototype IoT platform and demonstrate the efficiency in practical settings

Publicly release implementation as an open-source code

---

# Crypto Building Blocks

## Security Fundamentals

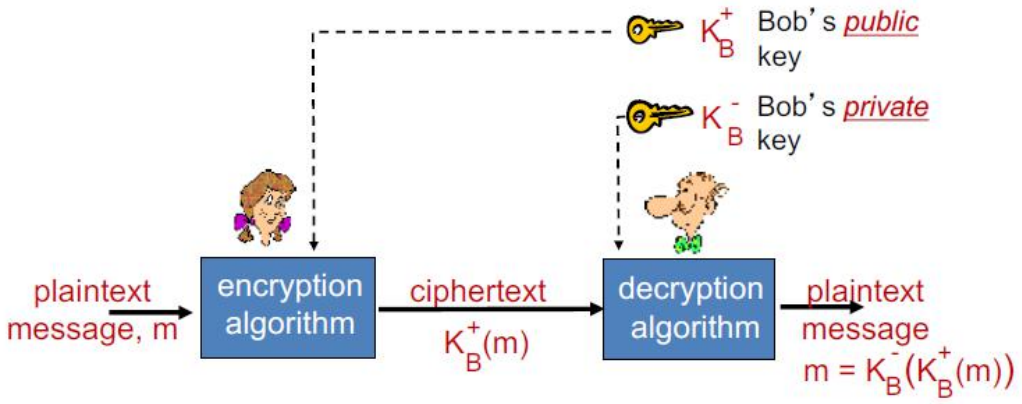
Confidentiality: only sender, intended receiver should “understand” message contents

Authentication: sender, receiver want to confirm identity of each other

Message integrity: ensure message not altered (in transit, or afterwards) without detection

Non-repudiation: associating actions or changes to a unique individual

## Public Key Cryptography




## Public Key encryption algorithms

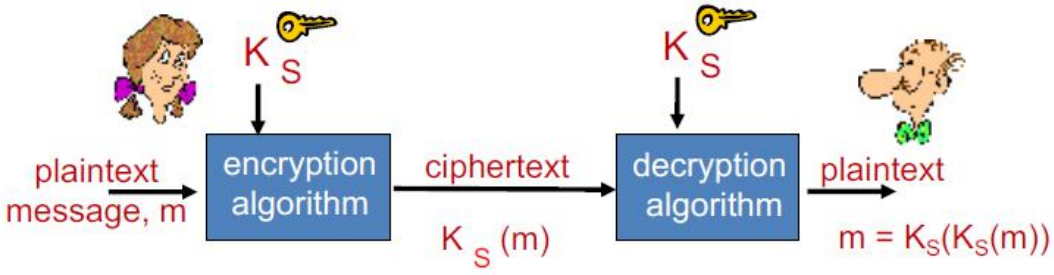
Need $K_B^+(.)$ and $K_B^-(.)$ such that

$K_B^-(K_B^+(m)) = m$

Given public key, it should be impossible to compute private key

RSA: Rivest, Shamir, Adelson algorithm

## Symmetric Key Cryptography




Symmetric key crypto: Bob and Alice share same (symmetric) key: $K_s$

## Session Keys

Public-key cryptography is computationally intensive as compared to symmetric key crypto

Exponentiation is computationally intensive

Symmetric Algorithms mush faster than RSA

**Session key, $K_s$**

Bob and Alice use RSA to exchange a symmetric key $K_s$

Once both have $K_s$, they use symmetric key cryptography

## Confidentiality vs Integrity

Confidentiality: message private and secret
Integrity: protection against message tampering
Encryption alone may not guarantee integrity
- Attacker can modify message under encryption without learning what it is
Public key Crypto Standards (PKCS)
- "RSA encryption is intended primarily to provide confidentiality ... It is not intended to provide integrity"
Both Confidentiality and integrity are needed for security

## Hash / Message Digests (MD)

Function H() that takes as input an arbitrary length message and outputs a fixed-length string: "message signature"
The values returned H() are called **hash values, hash codes, digests,** or simply **hashes.**
Note that H() is a many-to-1 function
H() is often called a "hash function"
Desirable properties:
- Easy to calculate
- Irreversibility: Can't determine m from H(m)
- Collision resistance:
    Computationally difficult to produce m and m' such that H(m) = H(m')
- Seemingly random output

## MD Randomness

Message digest of a hashing should have a random pattern
- Randomness: any bit in digest is "1" half the time
- Change input only one bit, and the hash will change half of the digest bits
- Diffusion: if hash function does not exhibit the avalanche effect to a slight change of input, then it has poor randomization, and thus a cryptanalyst can make predictions about the input, begin given only the output.

## Integrity vs Authentication

Suppose Alice creates msg m, and calculates hash H(m) using a hash function (e.g. SHA - 1)
Alice send the extended message (m, H(m)) to Bob
Bob upon receipt of this message independently calculates H(m') from the message
If H(m) = H(m'), message integrity verified

## HAMC

FIPS 198, RFC 2104

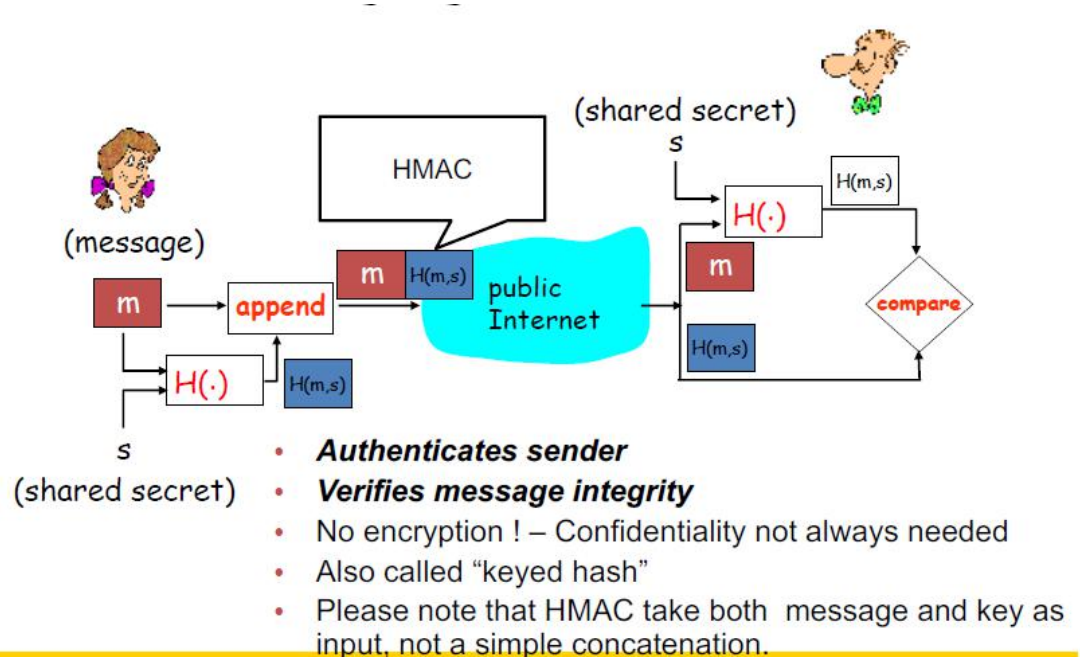
- Keyed-hash message authentication code: a message authentication code that uses a cryptographic key in conjunction with a hash function

Runs data and authentication key through hash function twice

- Hashing is faster than encryption in software
- Allow the use of any hash function, e.g. SHA-256, or SHA-512
- No US export restrictions on HMAC and hash

Used in TLS and IPSec

## HMAC: Integrity and Authentication




## Hash, MAC and HMAC

One way Hash Function: Doesn't take any secret key or its operation

- Easy to compute (both hardware and software solutions possible)
- Concatenation of a Secret Key and Message can be passed through a hash function without any need for encryption / decryption for efficiency.

Message Authentication Code: MAC

- Can use functions like DES, last bits of the cipher-text can be used as code. However, encryption software is slow, hence many be avoided.

HMAC:key is XORed with ipad (part of pad) and then concatenated with the message m. This mix is run through a hash function. The result is then concatenated with XOR of Key and opad (part of pad). Now, this whole mix is run through hash function one more time.

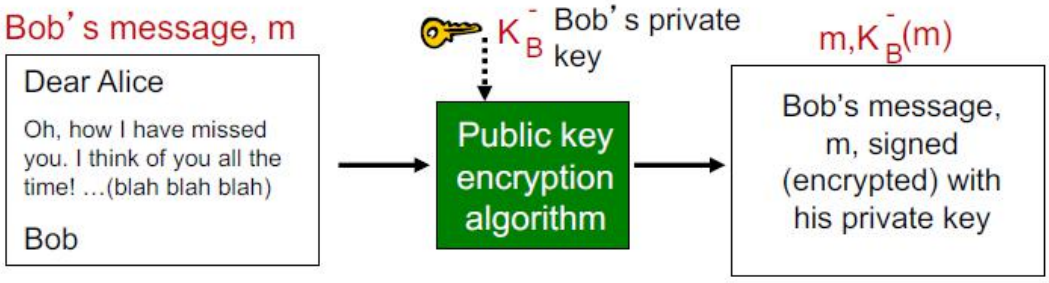
## Digital signatures

Cryptographic technique analogous to hand-written signatures:

- sender (Bob) digitally signs document, establishing he is document owner / creator.
- verifiable, nonforgeable: recipient (Alice) can prove to someone that Bob, and no one else (including Alice), must have signed document

simple digital signature for message m:

- Bob signs m by encrypting with his private key $K_B^-$, creating "signed" message, $K_B^-(m)$




Suppose Alice receives msg m, with signature: m, to $K_B^-(m)$

Alice verifies m signed by Bob by applying Bob's public key $K_B^+$ to $K_B^-(m)$ then checks $K_B^+(K_B^-(m)) = m$

If $K_B^+(K_B^-(m)) = m$, whoever signed m must have used Bob's private key.

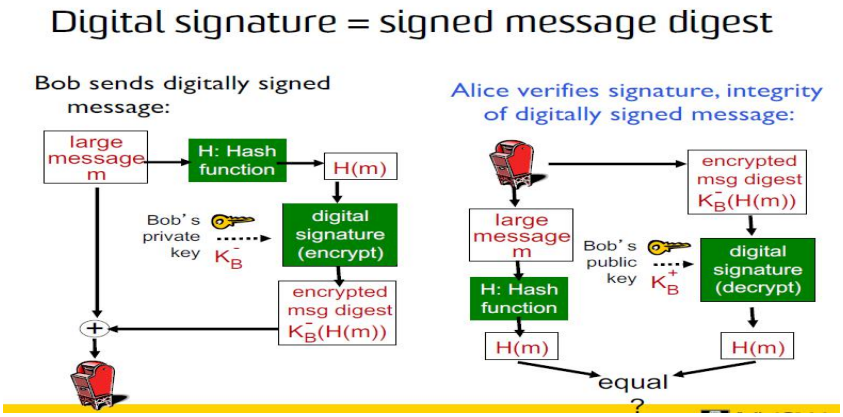
Alice thus verifies that:

- Bob signed m
- no one else signed m
- Bob signed m and not m'

Non-repudiation:

- Alice can take m, and signature $K_B^-(m)$ to court and prove that Bob signed m

## Digital signature = signed message digest



# Symmetric Key Ciphers and WLAN Security

## Two types of symmetric ciphers

Block ciphers

- Break plaintext message in equal-size blocks
- Encrypt each block as a unit
- Used in many Internet protocols (PGP-secure email, SSL (secure TCP), IPsec (secure net-transport layer))

Stream ciphers

- encrypt one bit at time
- Used in secure WLAN

## Block Cipher

- Ciphertext processed as *k* bit blocks
- 1-to-1 mapping is used to map k-bit block of plaintext to k-bit block of ciphertext
- E.g: k=3 (see table)
  - 010110001111 => 101000111001
- Possible permutations = 8! (40,320)
- To prevent brute force attacks
  - Choose large k (64, 128, etc)
- Full-block ciphers not scalable
  - E.g., for k = 64, a table with $2^{64}$ entries required
  - instead use function that simulates a randomly permuted table

| Input | Output |
|---|---|
| 000 | 110 |
| 111 | 001 |
| 001 | 111 |
| 010 | 101 |
| 011 | 100 |
| 100 | 011 |
| 101 | 010 |
| 110 | 000 |

## Cipher Block Chaining

Cipher block: if input block repeated, will produce same cipher text

- Sender creates a random *k*-bit number *r(i)* for *i*th block and calculates
  - $c(i) = K_s( m(i) \oplus r(i) )$
  - Sends ***c(1), r(1), c(2), r(2), c(3), r(3)***....
    - *r(i)* sent in clear but *Ks* not known to attackers.
  - Example: sent text 010010010 if no CBC, sent txt = *101101101*
  - *r(001), r2(111), r3(100)*
  - Use above technique to generate cipher text *c(1) = 100, c(2) = 010, c(3) = 000*: NOTE all three output different even though same plain text 101.
  - Inefficient: twice as many bits sent

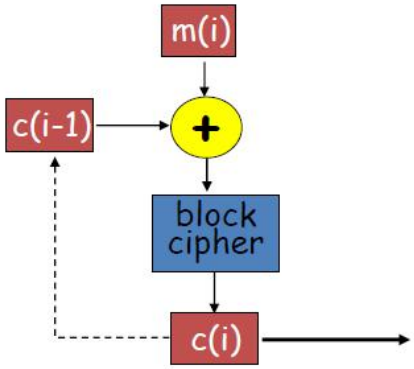
## CBC: Sender

- *cipher block chaining:* XOR $i$th input block, $m(i)$, with previous block of cipher text, $c(i-1)$
  - $c(0)$ is an Initialisation Vector transmitted to receiver in clear
  - First block:

$$c(1) = K_S(m(i) \oplus c(0))$$

  - *Subsequent blocks:*

$$c(i) = K_S(m(i) \oplus c(i-1))$$




## CBC: Receiver

- How to recover $m(i)$?
  - Decrypt with *Ks* to get $s(i) = Ks(c(i)) = m(i) \oplus c(i-1)$
  - Now the receiver knows $c(i-1)$, it can get $m(i) = s(i) \oplus c(i-1)$
  - **IV** sent only once
  - Intruder can't do much with **IV** since it doesn't have $Ks$
  - CBC has important consequence for designing secure network protocols

## Block Ciphers

Block cipher needs to wait for one block before processing it

- Suitable for storage

Operates on a single block of plaintext

- 64 bits for Data Encryption Standard (DES), 128 bits for Advanced Encryption Standard (AES)

AES much (6x) faster than DES

Computationally infeasible to break block cipher by brute-force by cracking the key

- Brute force decryption (try each key)

## Symmetric key crypto: DES

DES: Data Encryption Standard (US encryption standard [NIST 1993])
56-bit symmetric key, 64-bit plaintext input
Block cipher with cipher block chaining
How secure is DES?
- DES Challenge: 56-bit-key-encrypted phrase decrypted (brute force) in less than a day
- no known good analytic attack
Making DES more secure:
- 3DES: encrypt 3 times with 3 different keys

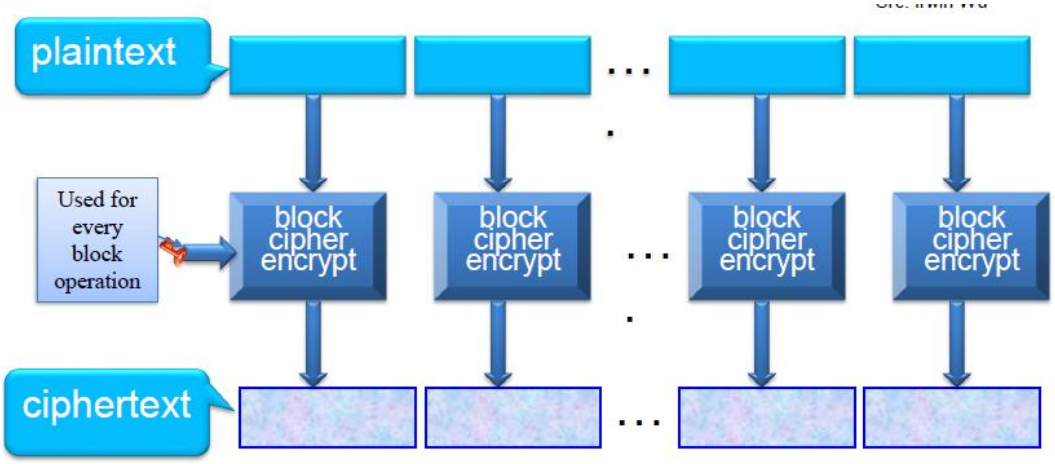
## AES: Advanced Encryption Standard

Symmetric-key NIST standard, replaced DES (Nov 2001)
Processes data in 128 bit blocks
128, 192 or 256 bit keys
Brute force decryption (try each key) taking few secs on DES, take 149 trillion years for AES

## AES Confidentiality Modes

Five confidentiality modes of operation for symmetric key block cipher algorithms, such as AES
Electronic Codebook (ECB),
- Split plaintext into blocks, encrypt each one separately using the block cipher
Cipher Block Chaining (CBC) mode
- Split plaintext into blocks, XOR each block with the result of encrypting previous blocks
Cipher Feedback (CFB), Output Feedback (OFB), and Counter (CTR) modes: stream cipher based on block cipher

## ECB Mode Encryption




- Problem: Identical blocks of plaintext produce identical blocks of ciphertext
- No integrity checks: can mix and match blocks

## Weakness of ECB

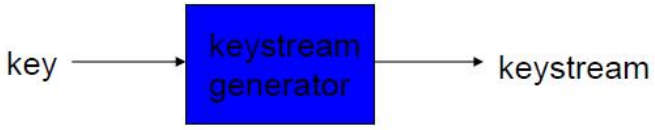
Message repetitions may show in ciphertext

Weakness is due to encrypted message block uniquely mapped to the plaintext block

Main use is sending only a few blocks of short data

If longer messages, use Cipher Block Chaining

## Stream Ciphers




- Process message bit by bit (as a stream)
  - Ideal for real-time communication
  - A keystream must not be reused; otherwise the encrypted messages can be recovered
- *combine each byte of keystream with byte of plaintext to get ciphertext:*
  - m(i) = ith unit of message
  - ks(i) = ith unit of keystream
  - c(i) = ith unit of ciphertext
  - $c(i) = ks(i) \oplus m(i)$ ($\oplus$ = exclusive or)
  - $m(i) = ks(i) \oplus c(i)$

## Rivest Cipher 4 (RC4)

Rivest Cipher 4 : Designed by Ron Rivest

- A proprietary cipher owned by RSA.com
- No longer a trade secret
- Ideal for software implementation, as it requires only byte manipulations

Variable key size (40 to 256 bits), byte-oriented stream cipher

Widely used

- SSL / TLS, Wireless WEP and WPA, Cellular Digital Packet Data, OpenBSD pseudo-ranndom number generator

## Wired Equivalent Privacy (WEP)

Provide security equivalent to Wired Network

Symmetric key crypto

- confidentiality
- end host authorization
- data integrity

Efficient

- implementable in hardware or software

## Stream cipher and packet independence

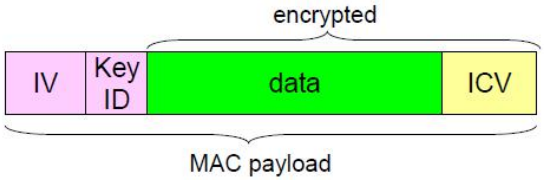
- Design goal: each packet separately encrypted
- If for frame $n+1$, use keystream from where we left off for frame $n$, then each frame is not separately encrypted
  - need to know where we left off for packet $n$ (e.g Cipher Block chain approach)
- WEP approach: initialize keystream with key + new IV for each packet:

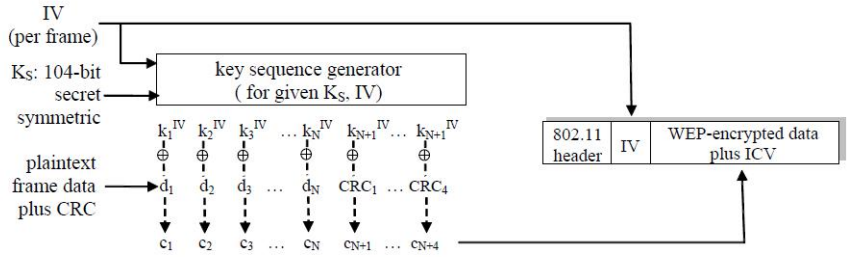

## WEP Pre-shared Key

- Enter a key (password) on access point and then then enter the key on all devices
  - This is the pre-shared key, AKA WEP Key *(Shared Secret)*.
- Not possible to authenticate individuals
  - hard to distinguish who is using service - needs extra work.
- A key compromise for one user means that every device needs to change new key
  - Must be distributed to all users securely

## WEP encryption

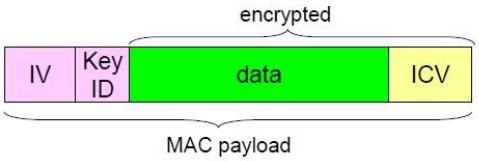
- sender calculates Integrity Check Value (ICV) over data
  - four-bytes for data integrity: uses CRC-32
- each side has 104-bit shared key
- sender creates 24-bit initialization vector (IV), appends to key: gives 128-bit key
- sender also appends keyID (in 8-bit field)
- 128-bit key input into pseudo random number generator PRNG e.g. RC4 to get keystream
- data in frame + ICV is encrypted with RC4:
  - Bytes of keystream are XORed with bytes of data & ICV
  - IV & keyID are appended to encrypted data to create payload




*new IV for each frame*

## WEB decryption

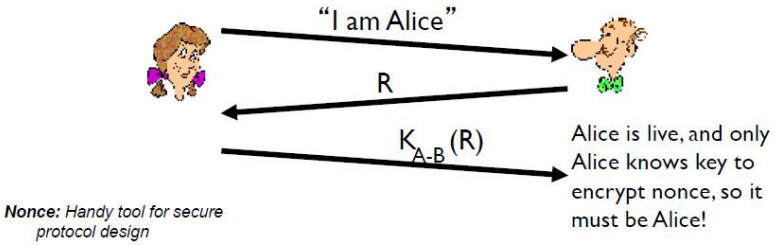



- receiver extracts IV (*received in plaintext*)
- inputs IV, shared secret key into pseudo random generator, gets keystream
- XORs keystream with encrypted data to decrypt data + ICV
- verifies integrity of data with ICV
  - note: message integrity approach used here is CRC-32 different from MAC (message authentication code) and signatures (using PKI).

## End-point authentication w / nonce

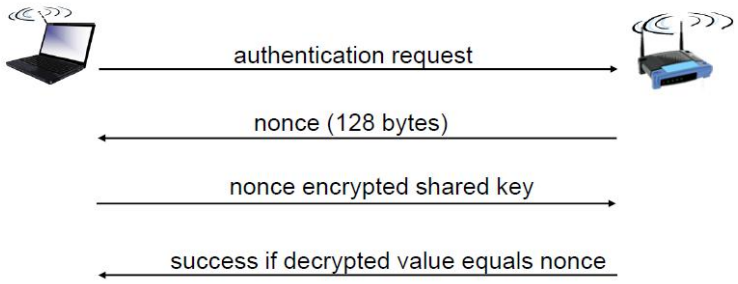
*Nonce:* number (R) used only *once –in-a-lifetime*

*How to prove Alice "live":* Bob sends Alice *nonce*, R. Alice must return R, encrypted with shared secret key




***Nonce:*** *Handy tool for secure protocol design*

## WEP authentication




*Notes:*

- not all APs do it, even if WEP is being used
- AP indicates if authentication is necessary in beacon frame
- done before association

## Breaking 802.11 WEP encryption

*Security hole:*

- 24-bit IV, one IV per frame, -> IV's eventually reused
  ~16Million IVs at high speed exhausted in 2 hours
- IV transmitted in ***plaintext*** -> IV reuse detected

*Attack:*

- Trudy causes Alice to encrypt known plaintext $d_1\ d_2\ d_3\ d_4 \ldots$
- Trudy sees: $c_i = d_i$ XOR $k_i^{IV}$
- Trudy knows $c_i\ d_i$, so can compute $k_i^{IV} = d_i$ XOR $c_i$
- Trudy knows encrypting key sequence $k_1^{IV}\ k_2^{IV}\ k_3^{IV} \ldots$
- Next time IV is used, Trudy can decrypt!

**Problem with Linear Checksum**

- Encrypted CRC-32 used as integrity check Vector (ICV)
  - Fine for random errors, but not malicious ones
  - Bits can be changed in packet without decrypting

- An attacker can change encrypted content (substitute by gibberish), compute a CRC over the substituted text and produce an 802.11 frame that will be accepted by the receiver.

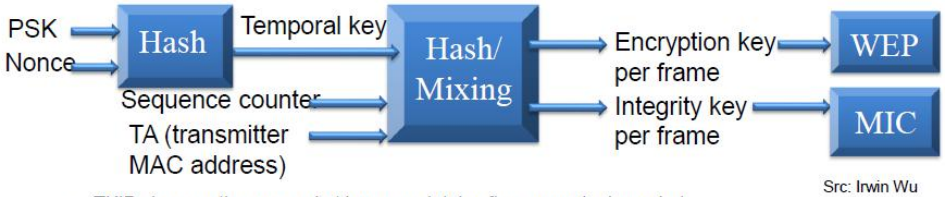
**Wi-Fi Protected Access (WPA)**

- Two versions WPA and WPA2
  - WPA temporary solution to fix WEP while WPA2 developed
- WPA compatible with existing hardware that supported WEP
- WPA uses Temporal Key Integrity Protocol (TKIP)
  - Used RC4 for compatibility
  - Every packet encrypted with unique encryption key

**802.11i: WPA - New Features**

- To provide stronger authentication than in WEP:
  - Special purpose Message Integrity Code (MIC) as opposed to WEP CRC
- To prevent Fluhrer, Mantin and Shamir (FMS) aka FMS-style attacks
  - a new *per-frame key is* constructed using a cryptographic hash
- Temporal Key Integrity Protocol (TKIP) uses a cryptographic mixing function to combine a temporal key, the TA (transmitter MAC address), and the sequence counter into the WEP seed (128 bits)
  - Pre Shared Key (PSK) AKA WPA-Personal similar to WEP-Key
    - However, it is not used for encryption
  - Instead, PSK serves as the seed for hashing the per-frame key

## 802.11i: WPA Contd.



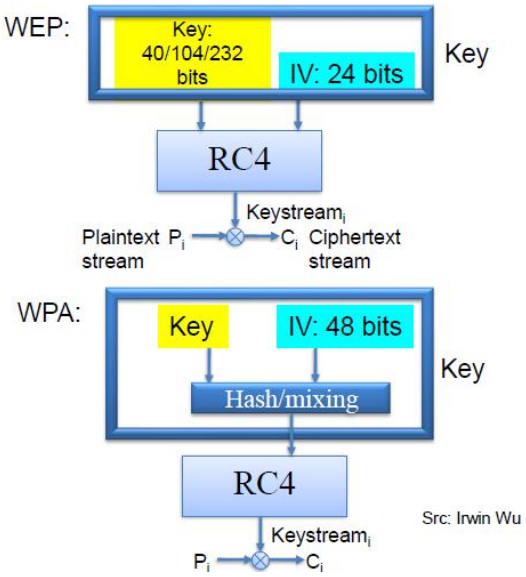

Src: Irwin Wu

- TKIP changes the per packet key completely after every single packet
  - One key for encryption (128 bits)
  - One key for integrity (64 bits)
- The WEP IV is extended to 48 bits, and used as a packet sequence counter
  - A per packet sequence counter is used to prevent replay attacks
  - If a packet is received out of order, it is dropped by the receiving station

802.11 frame with WPA


## Recap: WEP VS WPA security

- WPA temporary solution to fix WEP while WPA2 developed
- WEP IV extended to 48-bit IV
  - Reuse > 100 years for replay of the same IV
- RC4 key = Function(WPA Key||IV)
  - Every packet encrypted with unique encryption key
- IV used as a packet sequence space to prevent replay attack



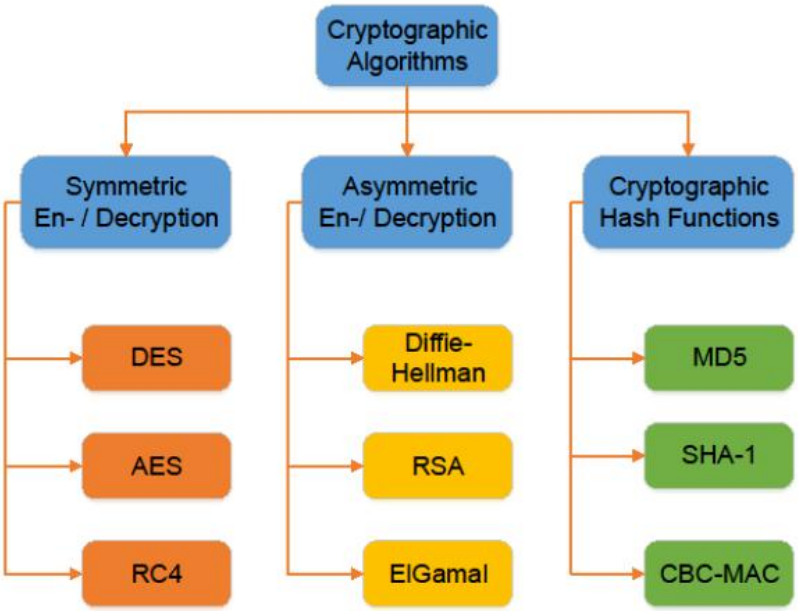

Src: Irwin Wu

## 802.11i: WPA2

- WPA2 2004
  - New AP hardware, 30 Million Instructions/sec, RC4 off-load hardware doesn't do AES or CCMP
  - AES-CCMP 128-bit AES
    - CCMP (Counter Mode with Cipher Block Chaining Message Authentication Code Protocol)
  - Improved 4-way handshake and temporary key generation
- We may cover some details of WPA2 but for now, if you have WPA2, this would be the safest option to use.
- WPA- Enterprise network security (802.1X) in later weeks

# RSA / DH, Certification Authority, Authentication

## Recap




## Public key encryption algorithms

Requirements:

1. need $K_B^+(\cdot)$ and $K_B^-(\cdot)$ such that

$$K_B^-(K_B^+(m)) = m$$

2. given public key $K_B^+$, it should be impossible to compute private key $K_B^-$

RSA: Rivest, Shamir, Adelson algorithm

## Public Key Cryptography

*symmetric key crypto*

- requires sender, receiver know shared secret key
- Q: how to agree on key in first place (particularly if never "met")?

*public key crypto*

- radically different approach [Diffie-Hellman76, RSA78]
- sender, receiver do *not* share secret key
- *public* encryption key known to *all*
- *private* decryption key known only to receiver

## RSA: getting ready

- A message is a bit pattern.
- A bit pattern can be uniquely represented by an integer number.
- Thus encrypting a message is equivalent to encrypting a number.

Example

- m= 10010001. This message is uniquely represented by the decimal number 145.
- To encrypt m, we encrypt the corresponding number, which gives a new number (the ciphertext).

## RSA: Creating public / private key pair

1. Choose two large prime numbers $p$, $q$. (e.g., 1024 bits each)
2. Compute $n = pq$, $z = (p-1)(q-1)$
3. Choose $e$ (with $e<n$) that has no common factors with z. ($e$, $z$ are "relatively prime"). E.g.: 4 and 9 are relatively prime. 6 and 9 are not.
4. Choose $d$ such that $ed-1$ is exactly divisible by $z$. (in other words: $ed$ mod $z = 1$ ).
5. *Public* key is $(n,e)$. *Private* key is $(n,d)$.

$K_B^+$ $K_B^-$

## RSA: Encryption, decryption

0. Given $(n,e)$ and $(n,d)$ as computed above
1. To encrypt bit pattern, $m$ ($m < n$), compute
   $c = m^e \bmod n$ (i.e., remainder when $m^e$ is divided by $n$)
2. To decrypt received bit pattern, $c$, compute
   $m = c^d \bmod n$ (i.e., remainder when $c^d$ is divided by $n$)

Magic happens! $m = (\underbrace{m^e \bmod n}_{c})^d \bmod n$

# RSA example

Bob chooses $p=5$, $q=7$. Then $n=35$, $z=24$.
$e=5$ (so $e$, $z$ relatively prime).
$d=29$ (so $ed-1$ exactly divisible by z).

| | letter | m | $m^e$ | $c = m^e \bmod n$ |
|---|---|---|---|---|
| encrypt: | l | 12 | 1524832 | 17 |

| | c | $c^d$ | $m = c^d \bmod n$ | letter |
|---|---|---|---|---|
| decrypt: | 17 | 481968572106750915091411825223071697 | 12 | l |

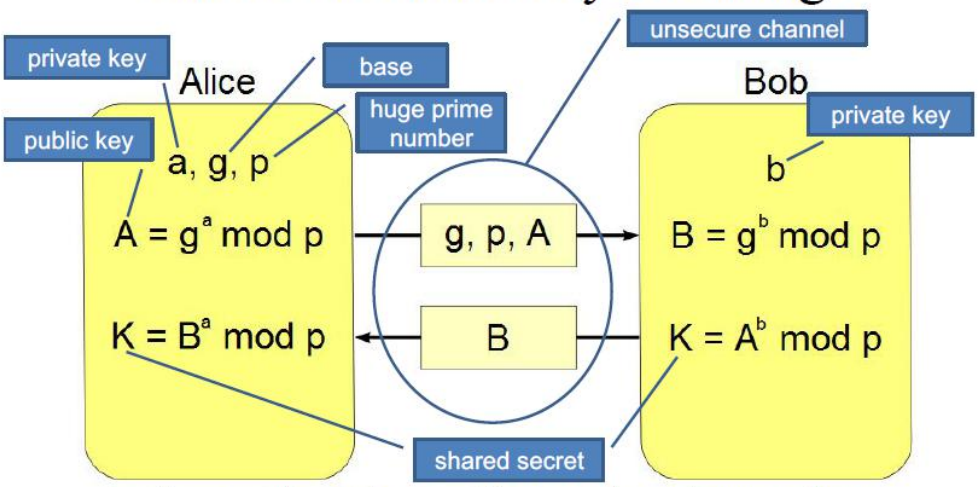
# RSA: another important property

The following property will be *very* useful later:

$$\underbrace{K_B^-(K_B^+(m))}_{\text{use public key first, followed by private key}} = m = \underbrace{K_B^+(K_B^-(m))}_{\text{use private key first, followed by public key}}$$

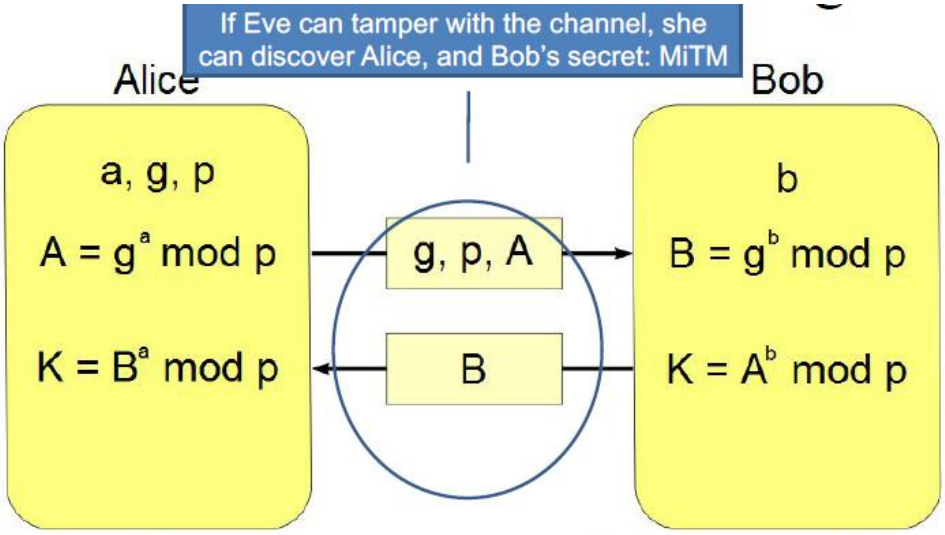
*Result is the same!*

# Diffie-Hellman key exchange




$K = A^b \bmod p = (g^a \bmod p)^b \bmod p = g^{ab} \bmod p = (g^b \bmod p)^a \bmod p = B^a \bmod p$

Alice's private key = 5, Bob's private key = 4, g=3, p=7
Alice's public key = $3^5 \bmod 7 = 5$, Bob's public key = $3^4 \bmod 7 = 4$
Alice's shared key = $4^5 \bmod 7 = 2$, Bob's shared key = $5^4 \bmod 7 = 2$




$K = A^b \bmod p = (g^a \bmod p)^b \bmod p = g^{ab} \bmod p = (g^b \bmod p)^a \bmod p = B^a \bmod p$

Alice's private key = 5, Bob's private key = 4, g=3, p=7

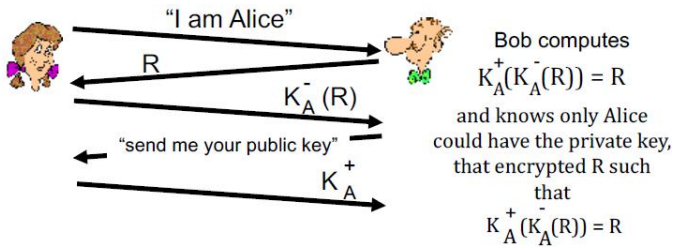
Alice's public key = $3^5 \bmod 7 = 5$, Bob's public key = $3^4 \bmod 7 = 4$

Alice's shared key = $4^5 \bmod 7 = 2$, Bob's shared key = $5^4 \bmod 7 = 2$

# Public-key certification

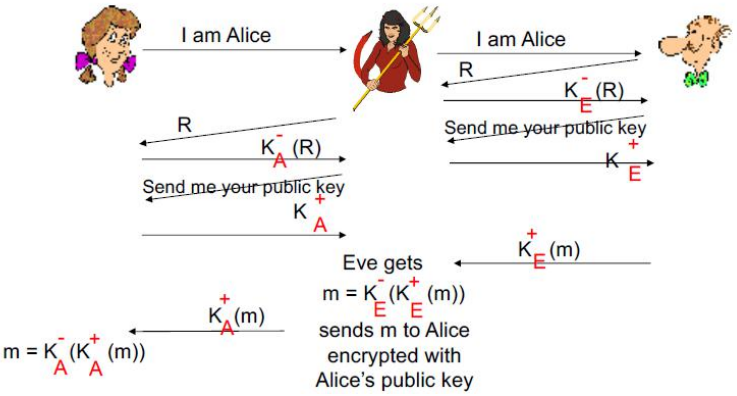
- motivation: Trudy plays pizza prank on Bob
  - Trudy creates e-mail order:
    *Dear Pizza Store, Please deliver to me four pepperoni pizzas. Thank you, Bob*
  - Trudy signs order with her private key
  - Trudy sends order to Pizza Store
  - Trudy sends to Pizza Store her public key, but says it's Bob's public key
  - Pizza Store verifies signature; then delivers four pepperoni pizzas to Bob
  - Bob doesn't even like pepperoni

# Kurose/Ross Authentication: ap5.0



# MiTM: Man in the Middle

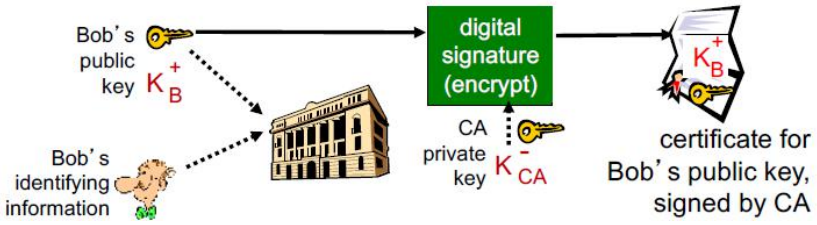
*man (or woman) in the middle attack:* Eve poses as Alice (to Bob) and as Bob (to Alice)



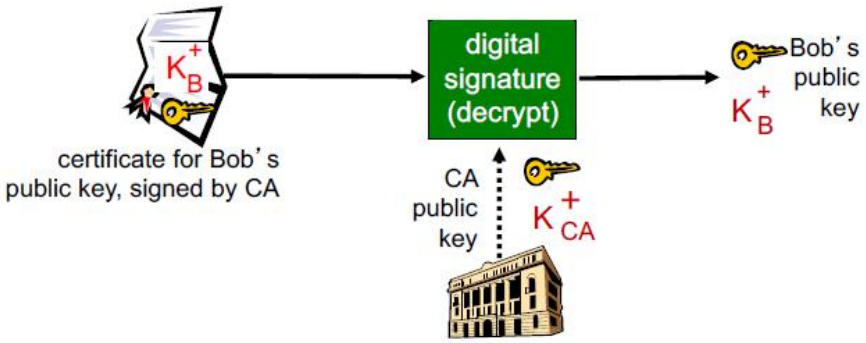

# Certification authorities

*certification authority (CA):* binds public key to particular entity, E.

- E (person, router) registers its public key with CA.
  - E provides "proof of identity" to CA.
  - CA creates certificate binding E to its public key.
  - certificate containing E's public key digitally signed by CA – CA says "this is E's public key"




- when Alice wants Bob's public key:
  - gets Bob's certificate (Bob or elsewhere).
  - apply CA's public key to Bob's certificate, get Bob's public key



# Public Key Distribution of Secret Key

- Prepare a message
- Encrypt that message using conventional encryption using one time session key
- Encrypt the session key using public-key encryption with Alice's **public key**
- Attach the encrypted session key to the message and send it to Alice
- Only Alice can decrypt the session key
- Bob has obtained Alice's public key by means of Alice's **public-key certificate**, must be a valid key

# X.509 Authentication Service

- Distributed set of servers that maintains a database about users.
- Each certificate contains the public key of a user and is signed with the private key of a CA.
- Is used in S/MIME, IP Security, SSL/TLS and SET.
- RSA is recommended to use but not mandatory.
- Digital Signature is assumed to use Hash algorithm
- Digital Certificate: user's id, public-key and CA information as input to hash function. Hash is then encrypted with CA's private key to produce **Digital Certificate**

# Obtaining a User's Certificate

- Characteristics of certificates generated by CA:
  - Any user with access to the public key of the CA can recover the user public key that was certified.
    - User can independently calculate hash, decrypt digital certificate using CA's public key, extract hash and compare if hashes match.
  - No part other than the CA can modify the certificate without this being detected.
- Certificates stored in a Directory server – not part of standard.

# Distributed Directory

- Users can be registered with a CA and would know its public Key
- Now if A got its certificate from CA X1 and B got it from CA X2.
- If A doesn't know CA X2's public key, it can't trust B's certificate issued by CA X2.
- However, if the two CA's have securely exchanged their public keys, then it can work.
  - A obtains the certificate of X2 signed by X1
  - A knows securely X1's public key
  - A obtains X2's public key from certificate and can verify using X1's signature on certificate
  - A can now get B's certificate from CA X2.
  - Since it has trusted public key for CA X2, things work as usual.

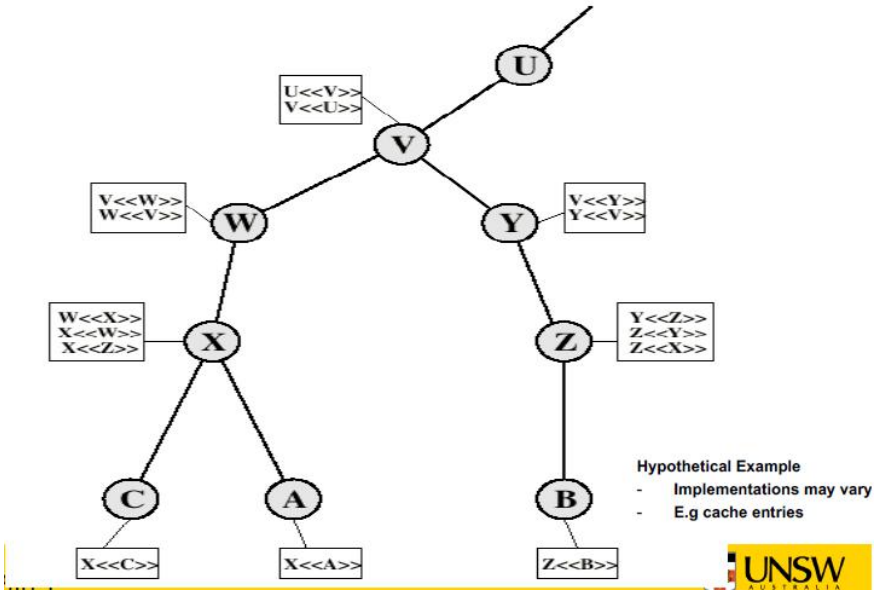
# Distributed Directory: Certificate Chain

- Notation $Y$ <<$X$>> Certificate of user $X$ issued by authority $Y$
- A obtains B's public key using the following X.509 notation

  $X_1$ << $X_2$>> $X_2$ << $B$>>
- B obtains A's public key using the following X.509 notation

  $X_2$<< $X_1$>> $X_1$ << $A$>>
  - *Arbitrary chain is possible as long as consecutive pair ($X_n$, $X_{n+1}$) of CAs have exchanged certificates securely*

# X.509 CA Hierarchy



# Hierarchy of CAs

- Previous figure: Connected Circles hierarchical relationship, boxes shows certificates maintained in each CA's directory
  - Forward Certs: Certs of X generated by other CAs (e.g at circle X, W<<X>>) – PARENT
  - Reverse Certs: Certs generated by X for others. (e.g. at circle X, X<<C>> X<<A>>) - CHILD
- A can acquire the following Certs from the directory to establish as certification to B

  *X <<W>> W <<V>> V <<Y>> Y<<Z>>Z<<B>>*

  *(Try to get A's certificate)*

# Revocation of Certificates

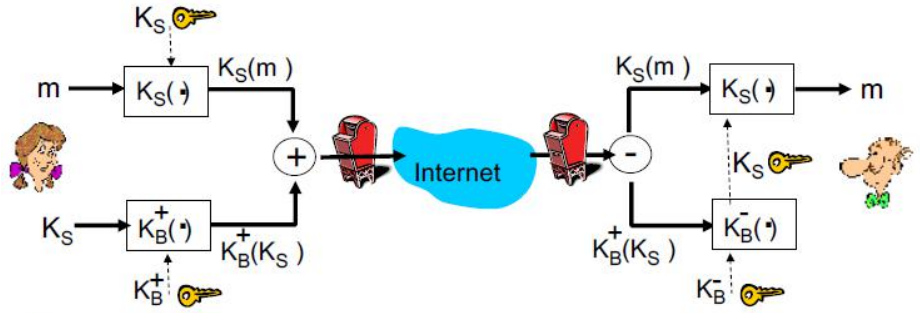
- Reasons for revocation:
  - The users secret key is assumed to be compromised.
  - The user is no longer certified by this CA.
  - The CA's certificate is assumed to be compromised.
- X.509 has a new version 3 with some recommendations for improvement – read in your own time if interested

# Application Layer Security

- Security can be implemented at various layers
- Application Layer Security
  - *A quick glimpse at secure email (PGP)*
- Secure Socket Layer (SSL) / Transport Layer Security (TLS)
  - Part 2 this week
- Network Layer Security
  - IPSec next week
- Link Layer Security
  - WLAN covered earlier, more on Enterprise later

# Secure e-mail: Alice

- Alice wants to send confidential e-mail, m, to Bob.

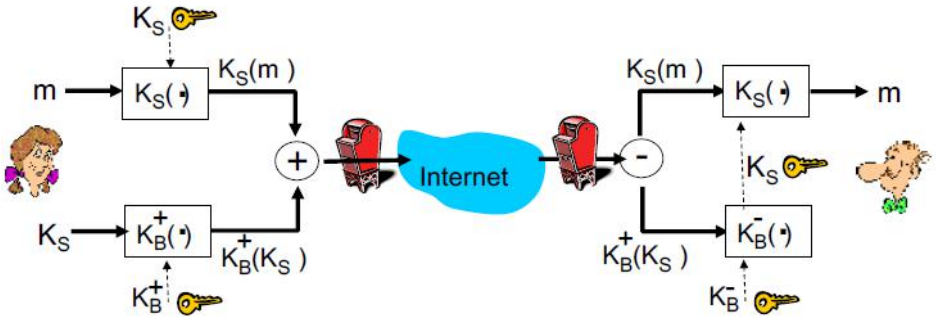



*Alice:*

- generates random *symmetric* private key, $K_S$
- encrypts message with $K_S$ (for efficiency)
- also encrypts $K_S$ with Bob's public key
- sends both $K_S(m)$ and $K_B(K_S)$ to Bob

# Secure e-mail: Bob

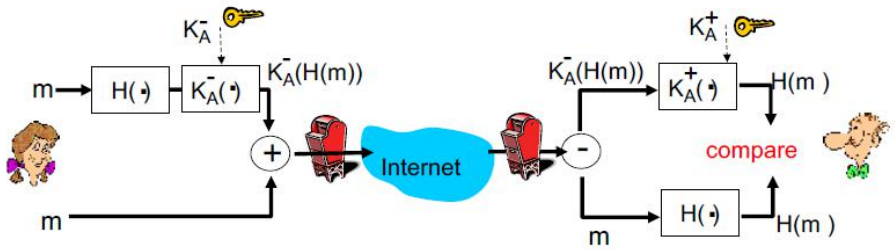
- Alice wants to send confidential e-mail, m, to Bob.




*Bob:*

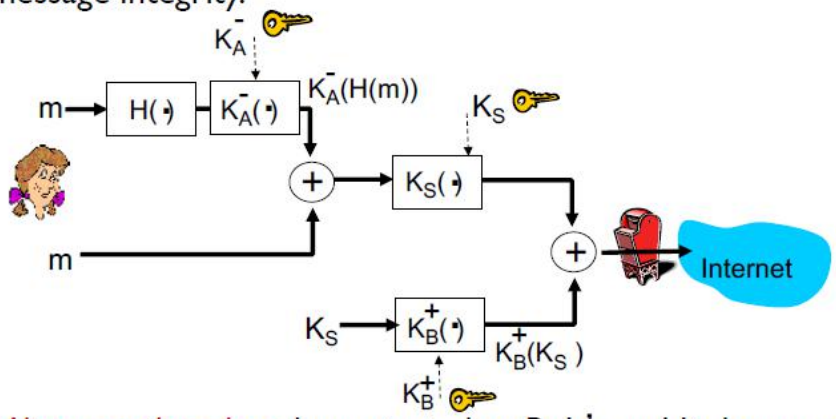
- uses his private key to decrypt and recover $K_S$
- uses $K_S$ to decrypt $K_S(m)$ to recover m

- Alice wants to provide sender authentication message integrity




- Alice digitally signs message
- sends both message (in the clear) and digital signature

- Alice wants to provide secrecy, sender authentication, message integrity.




*Alice uses three keys:* her private key, Bob's public key, newly created symmetric key (Exercise: see how Bob's side works)

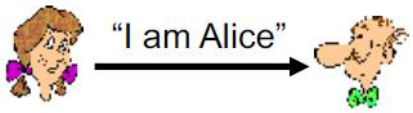
# Revise Authentication Basics

- Next set of foils are for self study you have already done in 3331/9331 (Kurose-Ross Ch8)
- These are basic building blocks and please revise them as they will help understand material covered in this subject.

# Authentication

*Goal:* Bob wants Alice to "prove" her identity to him

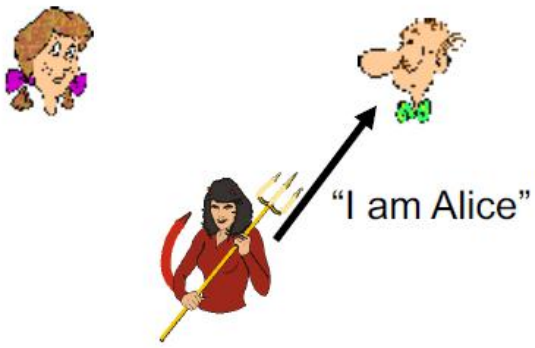
*Protocol ap1.0:* Alice says "I am Alice"




Failure scenario??

*Goal:* Bob wants Alice to "prove" her identity to him

*Protocol ap1.0:* Alice says "I am Alice"



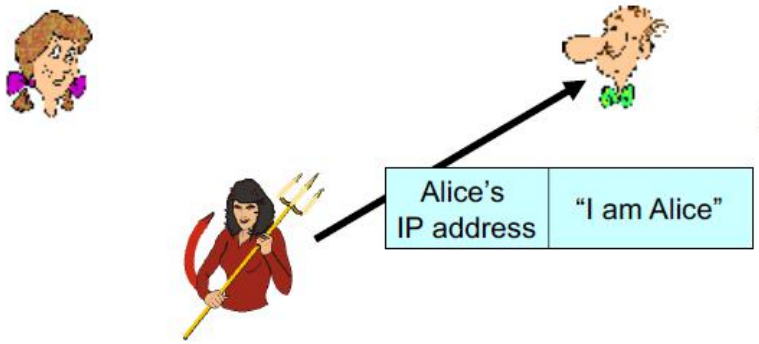

In a network,
Bob can not "see" Alice, so
Eve simply declares
herself to be Alice

## Authentication: another try

*Protocol ap2.0:* Alice says "I am Alice" in an IP packet containing her source IP address

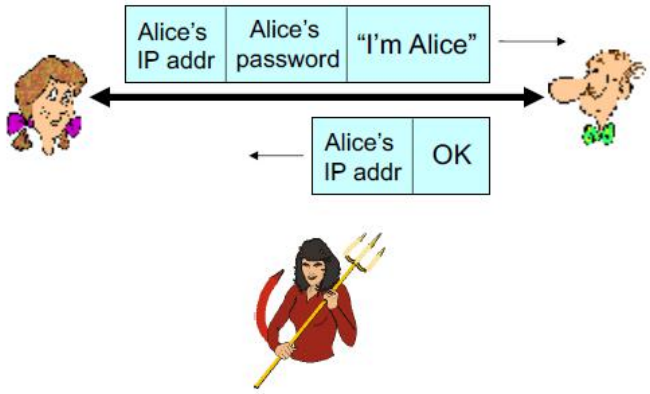

Failure scenario??

*Protocol ap2.0:* Alice says "I am Alice" in an IP packet containing her source IP address



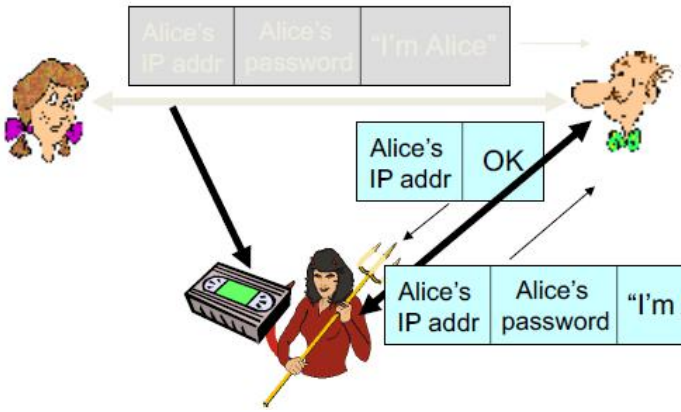

Eve can create
a packet "spoofing"
Alice's address

*Protocol ap3.0:* Alice says "I am Alice" and sends her secret password to "prove" it.



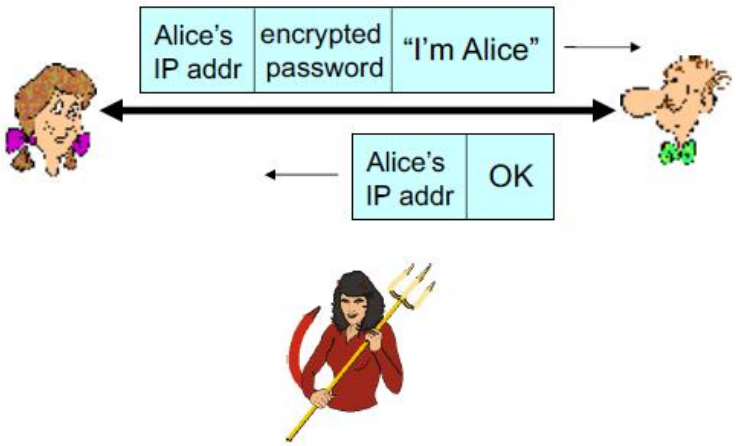

Failure scenario??

*Protocol ap3.0:* Alice says "I am Alice" and sends her secret password to "prove" it.



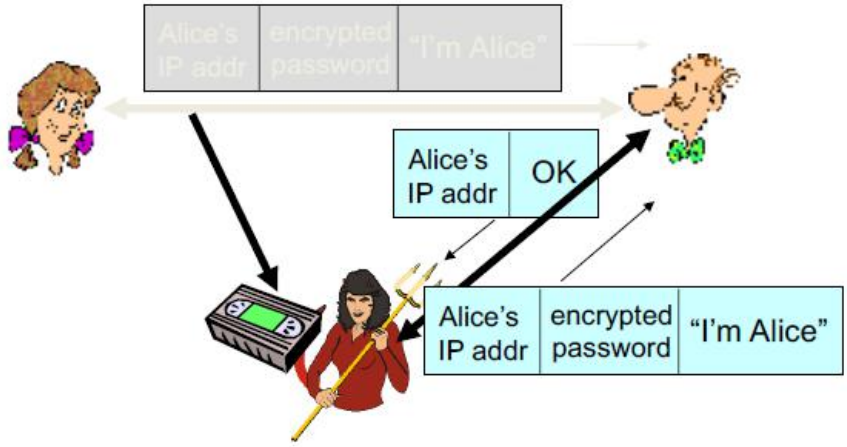

*playback attack:* Eve records Alice's packet and later plays it back to Bob

*Protocol ap3.1:* Alice says "I am Alice" and sends her *encrypted* secret password to "prove" it.




Failure scenario??

*Protocol ap3.1:* Alice says "I am Alice" and sends her *encrypted* secret password to "prove" it.

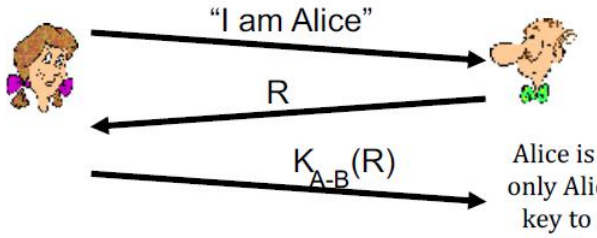



record and playback *still* works!

*Goal:* avoid playback attack

*nonce:* number (R) used only *once-in-a-lifetime*

*ap4.0:* to prove Alice "live", Bob sends Alice *nonce*, R. Alice must return R, encrypted with shared secret key




Alice is live, and only Alice knows key to encrypt nonce, so it must be Alice!

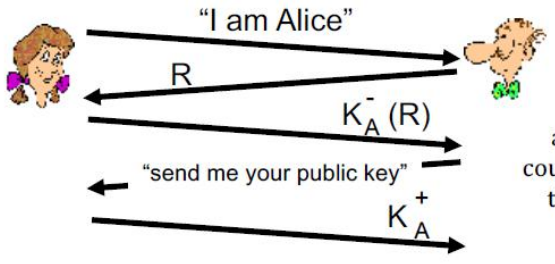
Failures, drawbacks?

# Authentication: ap5.0

ap4.0 requires shared symmetric key

- can we authenticate using public key techniques?

*ap5.0:* use nonce, public key cryptography

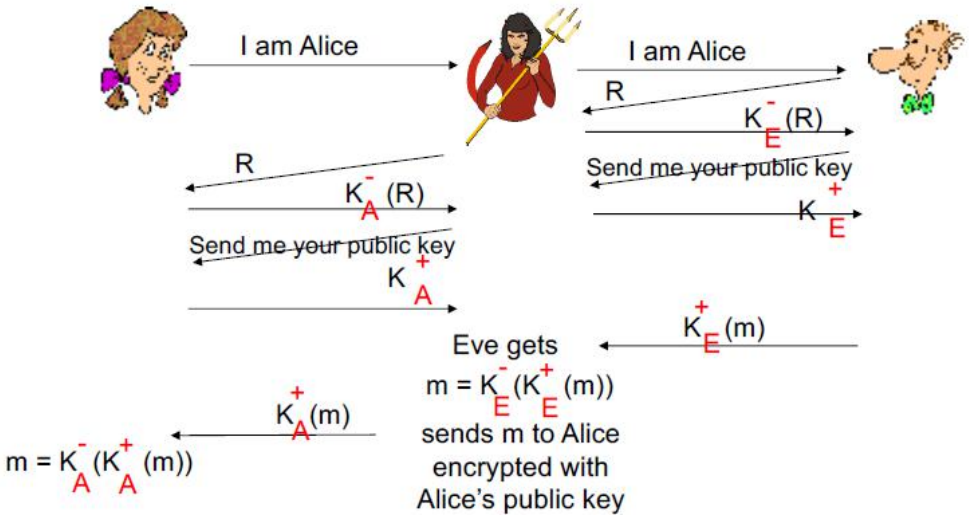



Bob computes $K_A^+(K_A^-(R)) = R$

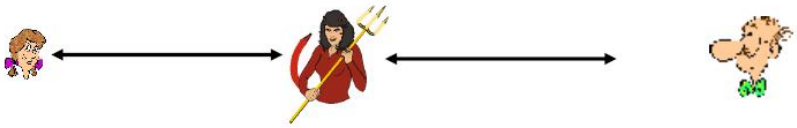
and knows only Alice could have the private key, that encrypted R such that $K_A^+(K_A^-(R)) = R$

# ap5.0: security hole

*man (or woman) in the middle attack:* Eve poses as Alice (to Bob) and as Bob (to Alice)




*man (or woman) in the middle attack:* Eve poses as Alice (to Bob) and as Bob (to Alice)



difficult to detect:

- Bob receives everything that Alice sends, and vice versa. (e.g., so Bob, Alice can meet one week later and recall conversation!)
- problem is that Eve receives all messages as well!

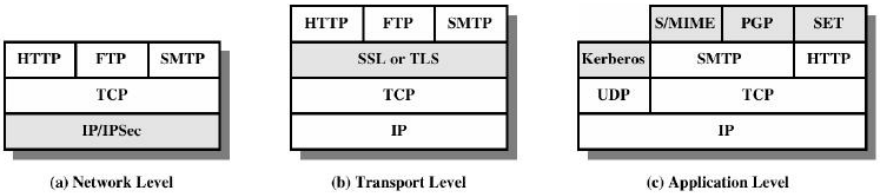
---

## Secure Socket Layer (SSL) and TLS

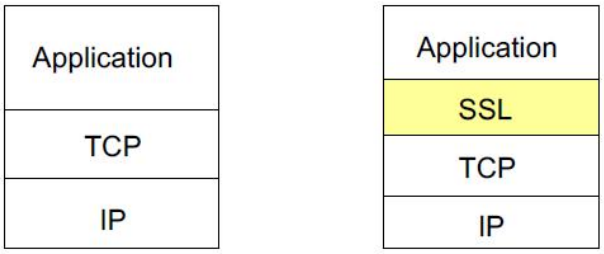
# SSL: Secure Sockets Layer

- widely deployed security protocol
  - supported by almost all browsers, web servers
  - https
  - billions $/year over SSL
- mechanisms: [Woo 1994], implementation: Netscape
- variation -TLS: transport layer security, RFC 2246
- provides
  - *Confidentiality, integrity, authentication*
- original goals:
  - Web e-commerce transactions
  - encryption (especially credit-card numbers)
  - Web-server authentication
  - optional client authentication
  - minimum hassle in doing business with new merchant
- available to all TCP applications
  - secure socket interface

# SSL in TCP/IP protocol stack

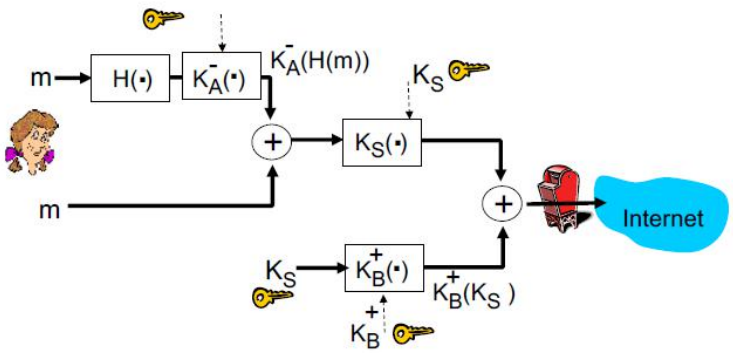



# SSL and TCP/IP




- SSL provides application programming interface (API) to applications
- C and Java SSL libraries/classes readily available

# Could do something like PGP

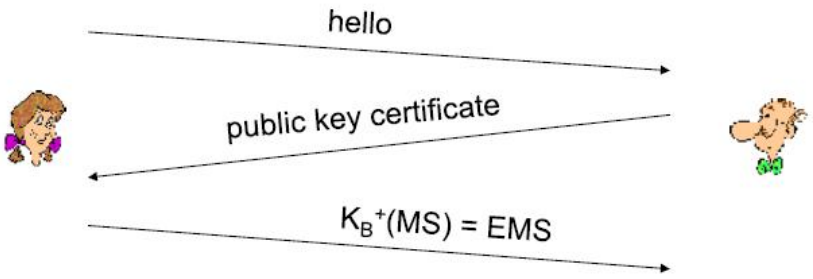



- but want to send byte streams & interactive data
- want set of secret keys for entire connection
- want certificate exchange as part of protocol: handshake phase

# Toy SSL: a simple secure channel

- *handshake:* Alice and Bob use their certificates, private keys to authenticate each other and exchange shared secret
- *key derivation:* Alice and Bob use shared secret to derive set of keys
- *data transfer:* data to be transferred is broken up into series of records
- *connection closure:* special messages to securely close connection

# Toy: a simple handshake



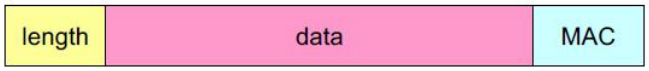

*MS:* master secret
*EMS:* encrypted master secret

# Toy: key derivation

- considered bad to use same key for more than one cryptographic operation
  - use different keys for message authentication code (MAC) and encryption
- four keys:
  - $K_c$ = encryption key for data sent from client to server
  - $M_c$ = MAC key for data sent from client to server
  - $K_s$ = encryption key for data sent from server to client
  - $M_s$ = MAC key for data sent from server to client
- keys derived from key derivation function (KDF)
  - takes master secret and (possibly) some additional random data and creates the keys

# Toy: data records

- why not encrypt data in constant stream as we write it to TCP?
  - where would we put the MAC? If at end of TCP connection, no message integrity until all data processed.
  - e.g., with instant messaging, how can we do integrity check over all bytes sent before displaying?
- instead, break stream in series of records
  - each record carries a MAC
  - receiver can act on each record as it arrives
- issue: in record, receiver needs to distinguish MAC from data
  - want to use variable-length records




# Toy: sequence numbers

- *problem:* attacker can capture and replay record or re-order records
- *solution:* put sequence number into MAC:
  - MAC = MAC($M_x$, sequence||data)
  - *note: no sequence number field*

- *problem:* attacker could replay all records in future
- *solution:* use nonce

# Toy: control information

- *problem:* truncation attack:
  - attacker forges TCP connection close segment
  - one or both sides thinks there is less data than there actually is.
- *solution:* record types, with one type for closure
  - type 0 for data; type 1 for closure
- MAC = MAC($M_x$, sequence||type||data)

# SSL Architecture

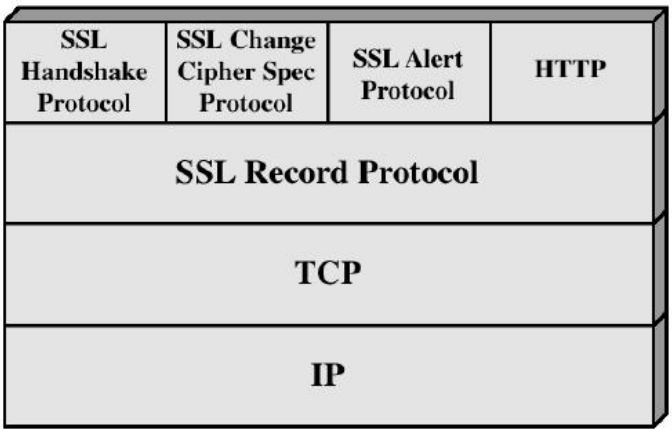



**Figure 7.2 SSL Protocol Stack**

# Real SSL: handshake (1)

*Purpose*

1. server authentication
2. negotiation: agree on crypto algorithms
3. establish keys
4. client authentication (optional)

# Real SSL: handshake (2)

1. client sends list of algorithms it supports, along with client nonce
2. server chooses algorithms from list; sends back: choice + certificate + server nonce
3. client verifies certificate, extracts server's public key, generates pre_master_secret, encrypts with server's public key, sends to server
4. client and server independently compute encryption and MAC keys from pre_master_secret and nonces
5. client sends a MAC of all the handshake messages
6. server sends a MAC of all the handshake messages

# Real SSL: handshaking (3)

Why last two messages with MAC exchanged?

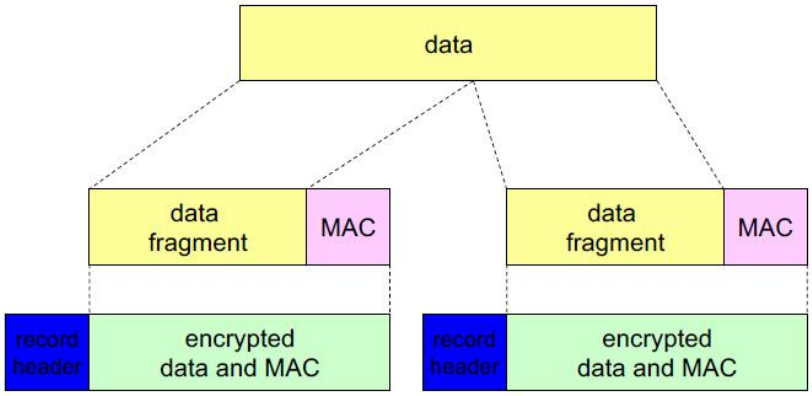
- client typically offers range of algorithms, some strong, some weak
- man-in-the middle could delete stronger algorithms from list
- last 2 steps prevent this
  - last two messages are encrypted

# Real SSL: handshaking (4)

Why two random nonces?

- suppose Trudy sniffs all messages between Alice & Bob
- next day, Trudy sets up TCP connection with Bob, sends exact same sequence of records
  - Bob (Amazon) thinks Alice made two separate orders for the same thing
  - solution: Bob sends different random nonce for each connection. This causes encryption keys to be different on the two days
  - Trudy's messages will fail Bob's integrity check

# SSL record protocol

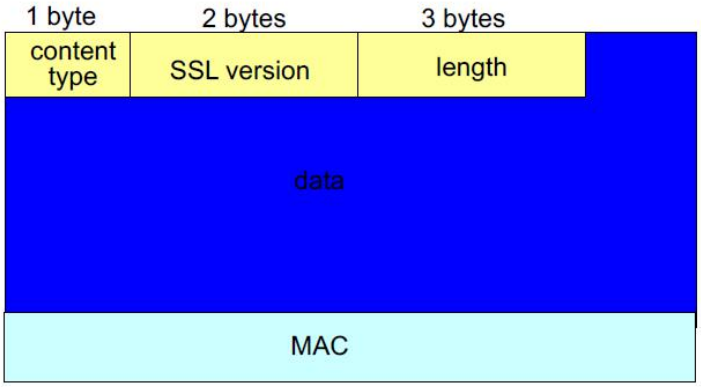



*record header:* content type; version; length

*MAC:* includes sequence number, MAC key $M_x$

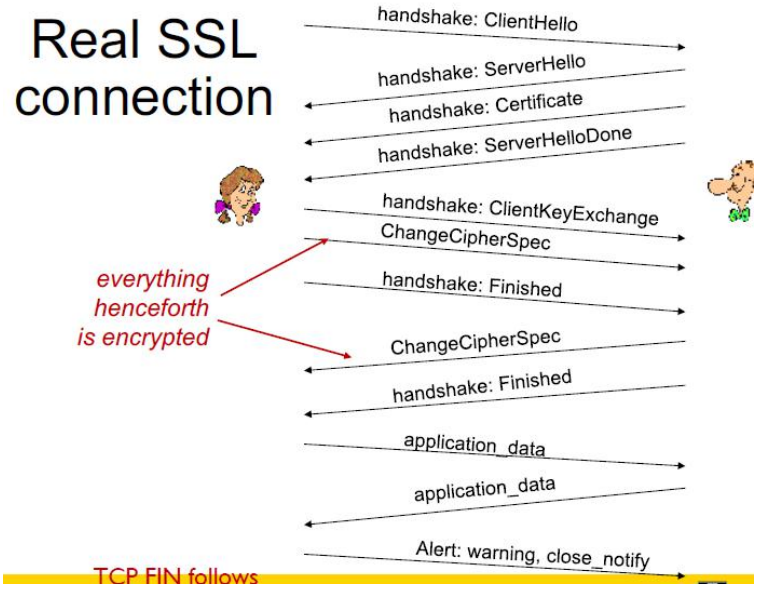
*fragment:* each SSL fragment $2^{14}$ bytes (~16 Kbytes)

# SSL record format




data and MAC encrypted (symmetric algorithm)

# Real SSL connection




# Key derivation

- client nonce, server nonce, and pre-master secret input into pseudo random-number generator.
  - produces master secret
- master secret and new nonces input into another random-number generator: "key block"

- key block sliced and diced:
  - client MAC key
  - server MAC key
  - client encryption key
  - server encryption key
  - client initialization vector (IV)
  - server initialization vector (IV)

# Transport Layer Security

- The same record format as the SSL record format.
- Defined in RFC 2246.
- Similar to SSLv3.
- Differences in the:
  - version number
  - message authentication code
  - pseudorandom function
  - alert codes
  - cipher suites
  - client certificate types
  - certificate_verify and finished message
  - cryptographic computations
  - padding

---

**Network Layer Security: IPsec**

## What is network-layer confidentiality ?

*between two network entities:*

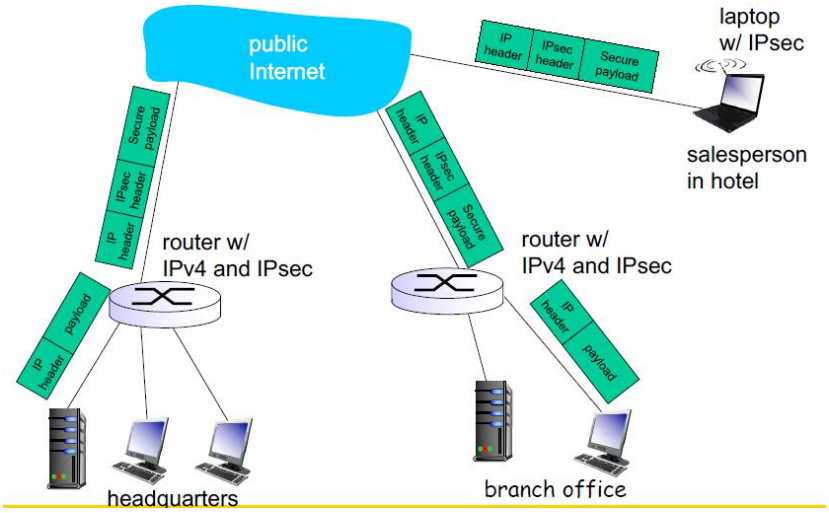
- sending entity encrypts datagram payload, payload could be:
  - TCP or UDP segment, ICMP message, OSPF message ….
    - Implemented below Transport layer (TCP/UDP)
    - Even when implmented in end-systems, doesn't affect higher layers
- all data sent from one entity to other would be hidden:
  - web pages, e-mail, P2P file transfers, TCP SYN packets …
- "blanket coverage"

## Virtual Private Networks (VPNs)

*motivation:*

- institutions often want private networks for security.
  - costly: separate routers, links, DNS infrastructure.
- VPN: institution's inter-office traffic is sent over public Internet instead
  - encrypted before entering public Internet
  - logically separate from other traffic

## Virtual Private Networks (VPNs)



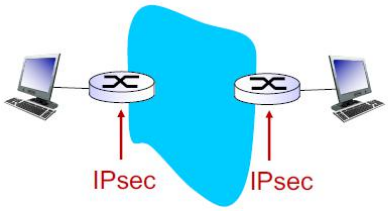

# IPsec services

- Data integrity
- Origin authentication
- Replay attack prevention
- Confidentiality
- Two different offerings (AH and ESP) discussed later

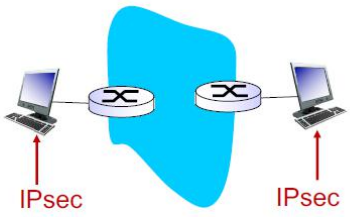
# Tunnel and Transport Modes

- Transport Mode: protects IP Payload from layers above (Higher layer TCP, UDP, ICMP..)
  - There are many detailed nuancene with AH/ESP support, encapsulation etc.
- Tunnel Mode: All of IP including header etc is encapsulated, new IP header is added by Firewall/Routers.
  - End hosts behind firewall don't have to worry about IPSec
  - We will focus on Tunnel Mode here as it is more widely used

## IPsec – tunneling mode




- edge routers IPsec-aware
- IP address of Routers/Gateways used, destination address encrypted




- hosts IPsec-aware
- Also possible that one host is IPSec aware and other behind firewall

## Advantages of tunnel mode (Edge)

- Simple key distribution: fewer keys needed as gateways do encryption/decryption
- Traffic analysis difficult as ultimate destination IP header concealed
- Less processing burden on end-hosts

## Two IPsec protocols

- Authentication Header (AH) protocol
  - provides source authentication & data integrity but *not* confidentiality
- Encapsulation Security Protocol (ESP)
  - provides source authentication, data integrity, *and confidentiality*
  - more widely used than AH

# Four combinations are possible!

| | |
|---|---|
| Transport mode with AH | Transport mode with ESP |
| Tunnel mode with AH | Tunnel mode with ESP |

most common and most important

# Protocol Operations for ESP

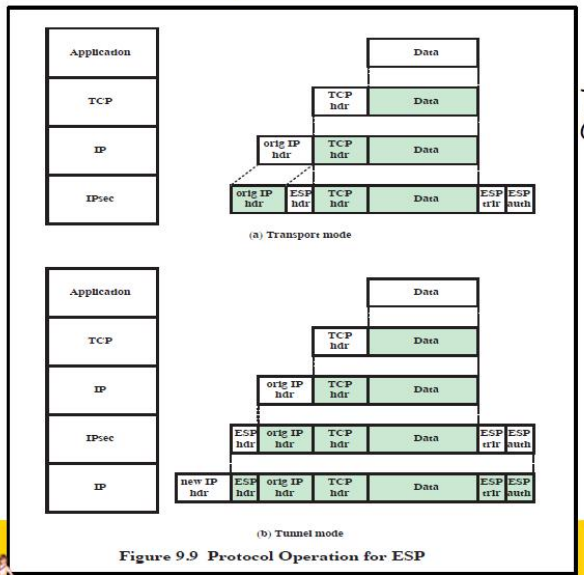



Figure 9.9 Protocol Operation for ESP

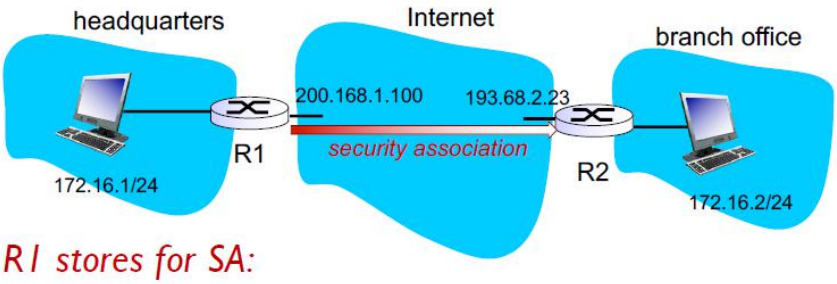
Transport:
Original IP Hdr

Tunnel
New IP Hdr

# Security associations (SAs)

- before sending data, "security association (SA)" established from sending to receiving entity
  - SAs are simplex: for only one direction
- ending, receiving entitles maintain *state information* about SA
  - recall: TCP endpoints also maintain state info
  - IP is connectionless; ***IPsec is connection-oriented!***
- *Combination of Security Associations has many advanced features : Read stallings Chapter9 (not examinable)*

# Example SA from R1 to R2




*R1 stores for SA:*

- 32-bit SA identifier: *Security Parameter Index (SPI)*
- origin SA interface (200.168.1.100)
- destination SA interface (193.68.2.23)
- type of encryption used (e.g., 3DES with CBC)
- encryption key
- type of integrity check used (e.g., HMAC with MD5)
- authentication key

## Security Association Database (SAD)

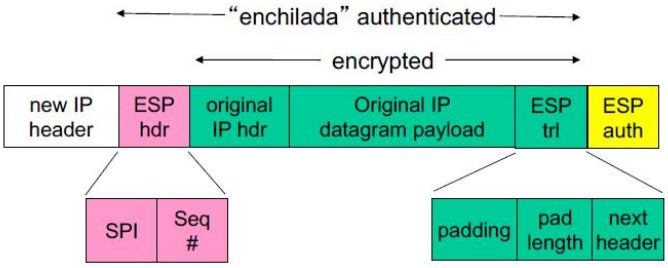
- endpoint holds SA state in *security association database (SAD)*, where it can locate them during processing.
- when sending IPsec datagram, R1 accesses SAD to determine how to process datagram.
- when IPsec datagram arrives to R2, R2 examines SPI in IPsec datagram, indexes SAD with SPI, and processes datagram accordingly.

# SAD Parameters

- Normally defined by the following parameters in a SAD entry:
  - Security parameter index
  - Sequence number counter
  - Sequence counter overflow
  - Anti-replay window
  - AH information
  - ESP information
  - Lifetime of this security association
  - IPsec protocol mode
  - Path MTU

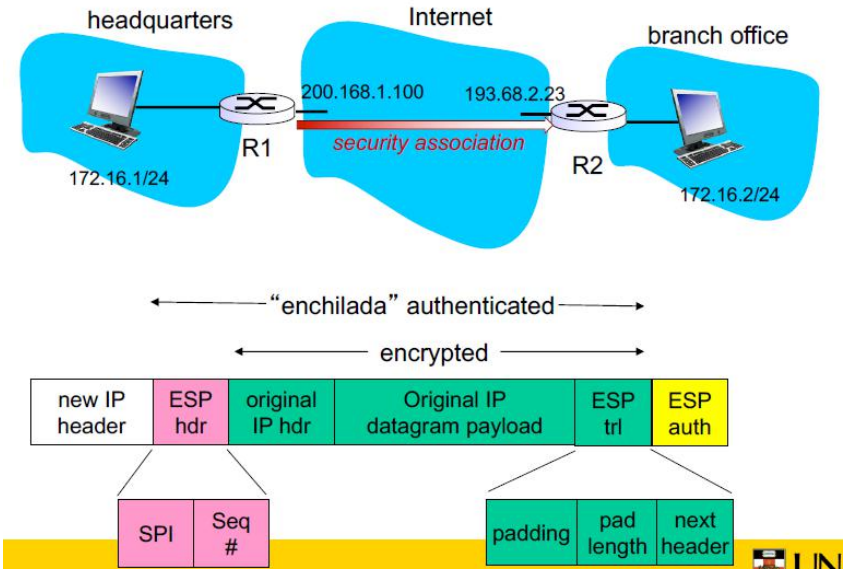
# IPsec datagram

focus for now on tunnel mode with ESP




Note; Original IP address/hdr encrypted, destination Router/GW

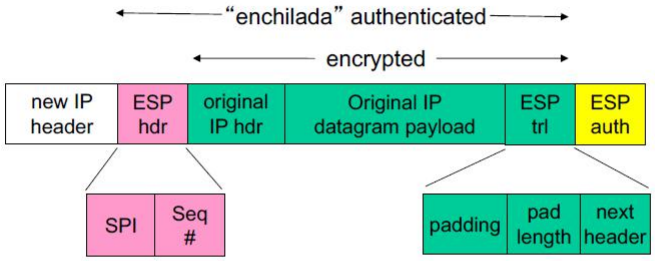
# What happens?




## R1: convert original datagram to IPsec datagram

- appends to back of original datagram (which includes original header fields!) an "ESP trailer" field.
- encrypts result using algorithm & key specified by SA.
- appends to front of this encrypted quantity the "ESP header, creating "enchilada".
- creates authentication MAC over the *whole enchilada*, using algorithm and key specified in SA;
- appends MAC to back of enchilada, forming *payload*;
- creates brand new IP header, with all the classic IPv4 header fields, which it appends before payload.

## Inside the enchilada:




- ESP trailer: Padding for block ciphers
- ESP header:
  - SPI, so receiving entity knows what to do
  - Sequence number, to thwart replay attacks
- MAC in ESP auth field is created with shared secret key (HMAC)
  - Note: ESP header is included in authentication

## IPsec sequence numbers

- for new SA, sender initializes seq. # to 0
  - each time datagram is sent on SA:
  - sender increments seq # counter
  - places value in seq # field
- goal:
  - prevent attacker from sniffing and replaying a packet
  - receipt of duplicate, authenticated IP packets may disrupt service
  - method:
    - destination checks for duplicates
    - doesn't keep track of *all* received packets; instead uses a window (remember from 3331)

# Security Policy Database (SPD)

- policy: For a given datagram, sending entity needs to know if it should use IPsec
- needs also to know which SA to use
  - may use: source and destination IP address; protocol number
- info in SPD indicates "what" to do with arriving datagram
- info in SAD indicates "how" to do it

# Host SPD Example

| Protocol | Local IP | Port | Remote IP | Port | Action | Comment |
|---|---|---|---|---|---|---|
| UDP | 1.2.3.101 | 500 | * | 500 | BYPASS | IKE |
| ICMP | 1.2.3.101 | * | * | * | BYPASS | Error messages |
| * | 1.2.3.101 | * | 1.2.3.0/24 | * | PROTECT: ESP intransport-mode | Encrypt intranet traffic |
| TCP | 1.2.3.101 | * | 1.2.4.10 | 80 | PROTECT: ESP intransport-mode | Encrypt to server |
| TCP | 1.2.3.101 | * | 1.2.4.10 | 443 | BYPASS | TLS: avoid double encryption |
| * | 1.2.3.101 | * | 1.2.4.0/24 | * | DISCARD | Others in DMZ |
| * | 1.2.3.101 | * | * | * | BYPASS | Internet |

Src: Stallings. Table9.2

# Notes on Table 9.2

Table 9.2 provides an example of an SPD on a host system (as opposed to a network system such as a firewall or router). This table reflects the following
configuration: A local network configuration consists of two networks. The basic corporate network configuration has the IP network number 1.2.3.0/24. The local configuration also includes a secure LAN, often known as a DMZ, that is identified as 1.2.4.0/24. The DMZ is protected from both the outside world and the rest of the corporate LAN by firewalls. The host in this example has the IP address 1.2.3.10, and it is authorized to connect to the server 1.2.4.10
in the DMZ.

The entries in the SPD should be self-explanatory. For example, UDP port 500 is the designated port for IKE. Any traffic from the local host to a remote host
for purposes of an IKE exchange bypasses the IPsec processing.

# IPsec Outbound Processing

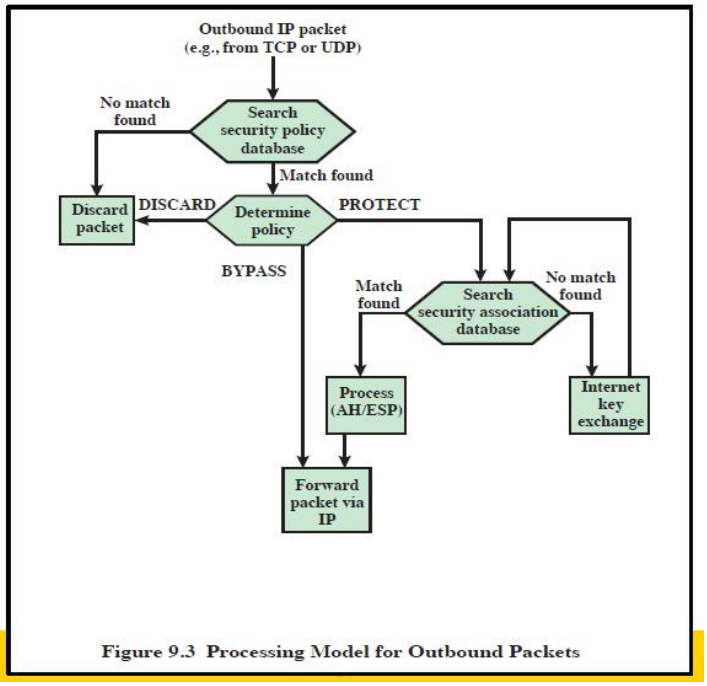



Figure 9.3 Processing Model for Outbound Packets

# IPsec Inbound Processing

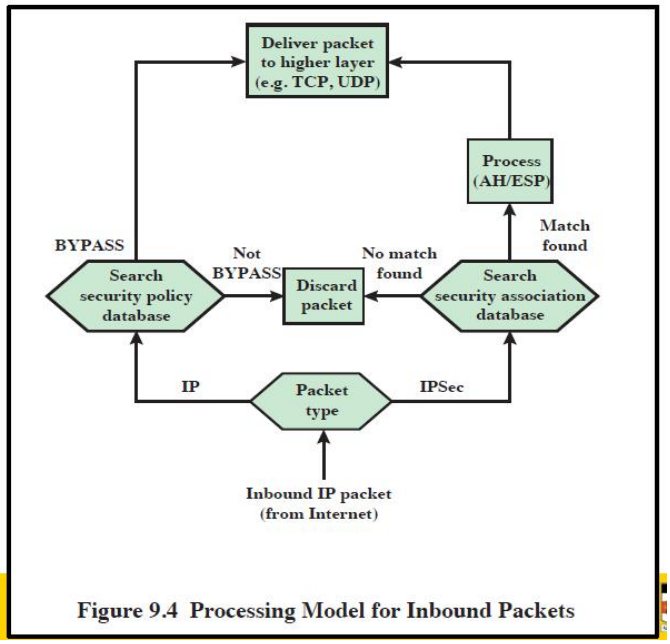



Figure 9.4 Processing Model for Inbound Packets

# IKE: Internet Key Exchange

- *previous examples:* manual establishment of IPsec SAs in IPsec endpoints:
  - *Example SA*
    - SPI: 12345
    - Source IP: 200.168.1.100
    - Dest IP: 193.68.2.23
    - Protocol: ESP
    - Encryption algorithm: 3DES-cbc
    - HMAC algorithm: MD5
    - Encryption key: 0x7aeaca…
    - HMAC key:0xc0291f…
- manual keying is impractical for VPN with 100s of endpoints
- instead use *IPsec IKE (Internet Key Exchange) RFC5996*

# IKE Features

- Each entity has a certificate (incl. Public Key)
- Similarity with SSL handshake
  - Exchange certificates
  - Negotiate authentication and encryption algorithms
  - Securely exchange Key Material for creating session keys in the IPSec SAs

# IKE phases

- IKE has two phases
  - *phase 1:* establish bi-directional IKE SA *different* from IPSec SA
    - Authenticated and encrypted tunnel between two end points for IKE messages
    - Est. a Master Key for use use in IPSec SA in phase 2
    - Another exchange for identity by signing their messages (Identities now protected by IKE SA, can't be sniffed)
      - » Also negotiated encryption/auth algorithms
  - *phase 2: two sides* negotiate IPSec a of SA in each direction
    - No PKI in second phase, hence large number of SA negotiation possible for scalablity.



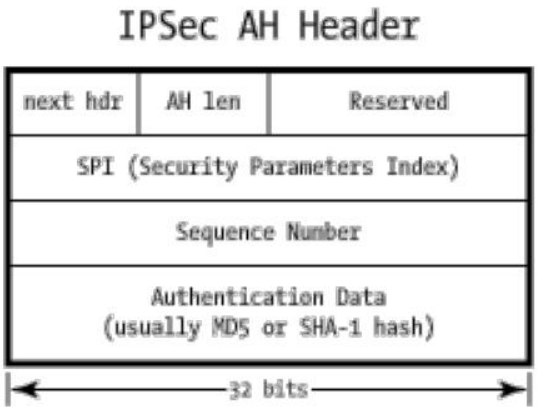
# IPsec summary

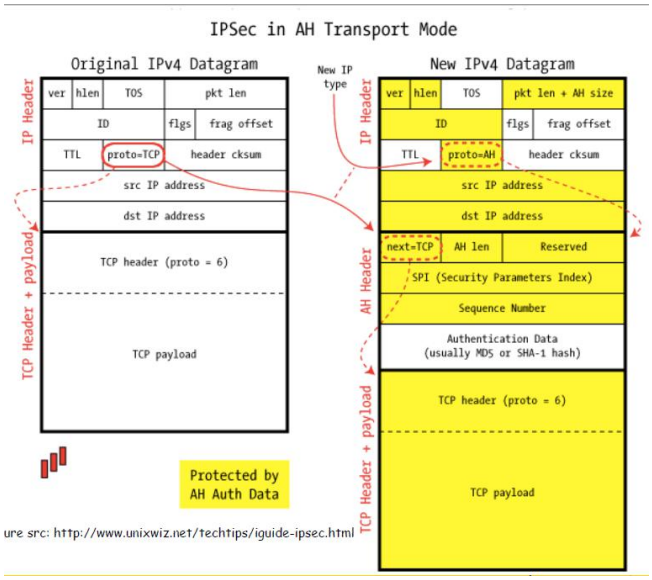
- IKE message exchange for algorithms, secret keys, SPI numbers
- either AH or ESP protocol (or both)
  - AH provides integrity, source authentication
  - ESP protocol (with AH) additionally provides encryption
- IPsec peers can be two end systems, two routers/firewalls, or a router/firewall and an end system

# Authentication Header (AH)

- AH provides ***authentication*** but not ***privacy***
- a special hashing algorithm and a specific key known only to the source and the destination used to generate authentication header
- Parts of the datagram used for the calculation, and the placement of the header, depends on the mode (tunnel or transport) and the version of IP (IPv4 or IPv6)
- Details at https://tools.ietf.org/html/rfc4302

# AH Header




## Tunnel Mode and Transport Mode Functionality (stallings Table 9.1)

No need to memorise

| | Transport Mode SA | Tunnel Mode SA |
|---|---|---|
| AH | Authenticates IP payload and selected portions of IP header and IPv6 extension headers. | Authenticates entire inner IP packet (inner header plus IP payload) plus selected portions of outer IP header and outer IPv6 extension headers. |
| ESP | Encrypts IP payload and any IPv6 extension headers following the ESP header. | Encrypts entire inner IP packet. |
| ESP with Authentication | Encrypts IP payload and any IPv6 extension headers following the ESP header. Authenticates IP payload but not IP header. | Encrypts entire inner IP packet. Authenticates inner IP packet. |

# WLAN 802.1X Authentication

## Challenges for Enterprise

- Pre Shared Key (PSK) not scalable
  - Max 64 hex characters, configure manually in each device
- E.g. 100 Employees, all share same Key.
- One leaves the company
  - Configure 99 devices with new key
- We have learnt the vulnerability with WEP/WPA – and labs.
- WPA2 provides CCMP/AES.
- We learnt about SSL and IPSec
  - Provide lot of flexibility/Option in configuring security at network and transport layers
- Advanced Authentication Methods based on "Extensible Authentication Protocol (EAP)" – topic of this lecture

## AAA

- Authentication: verification of user identity and credentials
  - May be multifactor: biometric etc.
- Authorization: granting access to resources and services
  - Needs authentication first.
- Accounting: tracking network use by users
  - Important to keep log
  - Required by many industry regulators
  - Helpful for billing/charging

## Authentication in WLAN - recap

- Username and passwords
- Digital Certificates
- Dynamic/One Time passwords
- Smartcards or credential on USBs
- Machine authentication (based on embedded identity)
- Pre-shared Keys (We saw WEP, WPA using this earlier)
- WLAN Example of MF: A registered computer and a legitimate user which has entry in a DB e.g. A Microsoft Active Directory

# Authorization in WLAN

- Various applications and higher layer protocols have their own authorization schemes.
- WLAN can provide authorization via 801.X framework at Layer-2 (can be used with Robust Security Network (RSN))
  - Port based access control (more later), for both wired and wireless network
  - Lot of standard documents for various bits/pieces – not focus of this subject
- Accounting is an important part but not within scope of WSN
  - Useful for forensics though

# IEEE 802.1X Port based Authentication

- Port Based: User must authenticate to switch they are physically connected to.
- Involves 3-party communications (nomenclature from 802.1X standard)
  - Supplicant
    - User
  - Authenticator
    - Ethernet switch, wireless access point
  - Authentication server
    - RADIUS (Remote access dial-in user service) database, Kerberos, LDAP or AD (Can be co-located with Authenticator)


# Supplicant

- Device to be authenticated for resource use
- Uses EAP protocol to connect to Auth. Server
- Until identity verified – can't use higher layer protocols (3 – 7)
- Can be software/app running 802.1X client
- OS based supplicants:
  - Microsoft Wireless Zero Conf – WZC
    - Known problems with supplicant software
  - Apple's airport client
- Chipset vendors may provide supplicant software
  - Intel, Atheros, Broadcom

# Authenticator (Access Point)

- For EAP, acts as a relay between Supplicant and Auth. Server
- Two Virtual Ports:
  - Uncontrolled : allows EAP authentication traffic
  - Controlled: Only authenticated traffic
- With WLAN Bridging solution:
  - Root bridge (a nominated bridge) is authenticator and other connected ones are supplicant
- Configured with address of Authentication Server
  - Possible co-location of Auth. Server with Authenticator
  - Shared Secret with Auth. Server

# Authentication Server: RADIUS (1)

- RADIUS provides centralized authentication, authorization and accounting management for user/host to access a network service/resource
  - Details in RFC 2865
- Supports AAA (Authentication, Authorization and Accounting) – a.k.a " Triple A"
  - RFC 3579 (AAA protocols such as RADIUS/EAP)
  - RADIUS is used to shuttle RADIUS-encapsulated EAP Packets between authenticator and an authentication server
- Most network equipment supports RADIUS
  - Wireless AP, VPN appliance, SSL, etc.
- Keeps an audit log of user's activity – accountability
- Radius Server
  - Standalone – local DB
  - Use External DB – e.g Active Directory
  - UDP Port 1812 for Auth, 1813 for Acct.
- Any other server can also be directly used in place of RADIUS

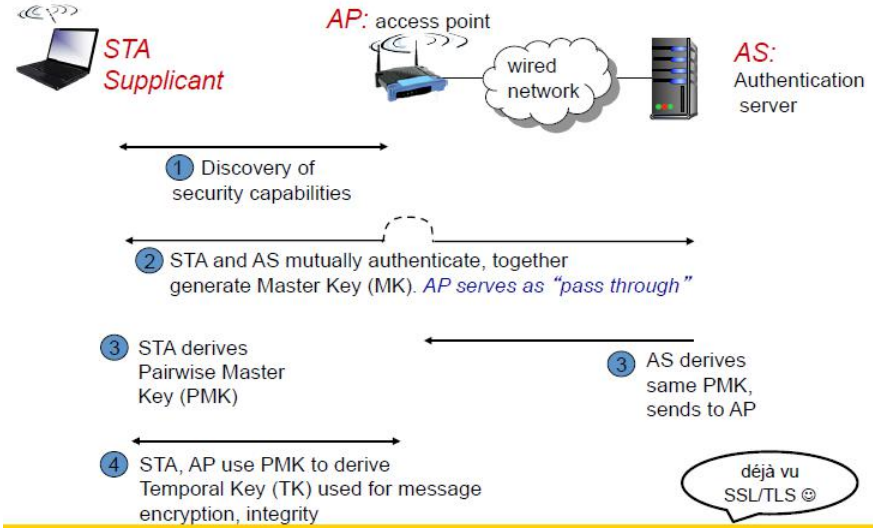
# RADIUS (2)

- Radius Server and Authenticator configured with a shared secret.
- Authenticator sends a RADIUS Access Request message to the RADIUS server, requesting authorization to grant access via the RADIUS protocol
  - This request includes access credentials (e.g., username and password)
  - Authentication server checks the credentials using the RADIUS server, Kerberos server, LDAP or Active Directory server
  - returns one of three responses
    - Access Accept, Access Reject, Access Challenge for extra credentials

# RADIUS server examples

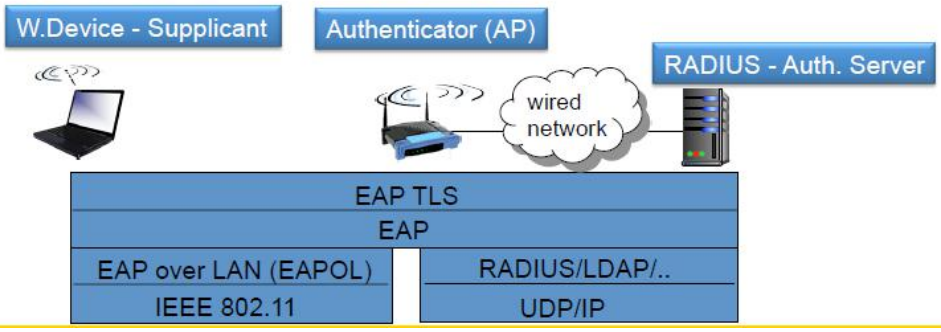
- Elektron (US$750) is an entry-level and user-friendly server
- ClearBox (US$599) is designed for small networks, but it also scales to larger networks
- FreeRADIUS (open source) is a solid and economical choice for Unix/Linux admins offering the most customization and flexibility

# Four phases of operation (Short Story)




# EAP: extensible authentication protocol (Short Story)

- EAP: end-end client (mobile) to authentication server protocol
- EAP sent over separate "links"
  - WirelessDevice-to-AP (EAP over LAN)
  - AP to authentication server (RADIUS over UDP)



# 802.1X protocol - Long Story Continue

- When a new client (supplicant) is connected to an authenticator, the port on the switch/wireless AP (authenticator) is enabled and set to the "unauthorized" state
  - In this state, only 802.1X traffic is allowed
  - Other traffic, such as DHCP and HTTP, is blocked at the data link layer
  - Steps
    - Authenticator sends out the EAP-Request identity to the supplicant
    - Supplicant responds with the EAP-response packet that the authenticator forwards to the authenticating server
    - If the authenticating server accepts the request, the authenticator sets the port to the "authorized" mode and normal traffic is allowed
    - When the supplicant logs off, it sends an EAP-logoff message to the authenticator; the authenticator then sets the port to the "unauthorized" state, once again blocking all non-EAP traffic

# EAP (1)

- EAP was designed for use in local network access authentication, where IP layer connectivity may not be available
  - When operating as a "pass-through authenticator" in the RADIUS/EAP scenario (RFC 3579), an authenticator performs checks on the Code, Identifier, and Length fields in accordance with EAP packet format and forwards EAP packets received from the supplicant to the authentication server

| code | ID | Length | Data |
|---|---|---|---|

- The Code field is one octet and identifies the Type of EAP packet. EAP Codes are assigned as follows:
  - 1 Request
  - 2 Response
  - 3 Success
  - 4 Failure
- EAP defines message formats in RFC 3748
- RFC 5247 specifies the EAP key hierarchy and provides a framework for the transport and usage of keying material and parameters generated by EAP authentication algorithms, known as "methods"
- *Read RFCs if interested – not examinable unless explicity requested for exam.*

# EAPOL Messages

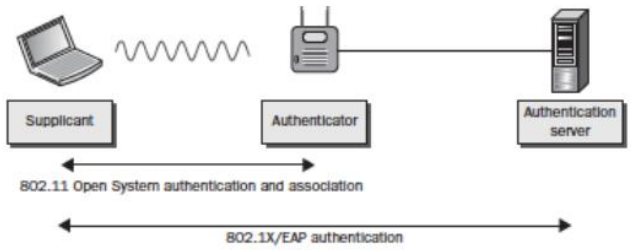
**TABLE 4.2** EAPOL Messages

| Packet Type | Name | Description |
|---|---|---|
| 0000 0000 | EAP-Packet | This is an encapsulated EAP frame. The majority of EAP frames are EAP-Packet frames. |
| 0000 0001 | EAPOL-Start | This is an optional frame that the supplicant can use to start the EAP process. |
| 0000 0010 | EAPOL-Logoff | This frame terminates an EAP session and shuts down the virtual ports. Hackers sometimes use this frame for DoS attacks. |
| 0000 0011 | EAPOL-Key | This frame is used to exchange dynamic keying information. For example, it is used during the 4-Way Handshake. |
| 0000 0100 | EAPOL- Encapsulated - ASF-Alert. | This frame is used to send alerts, such as SNMP traps to the virtual ports. |

Src: CSWP Text

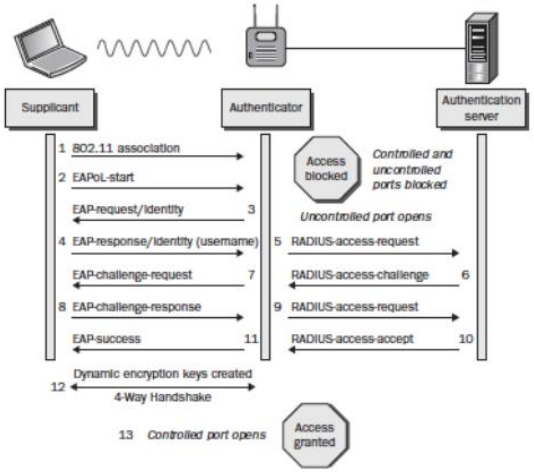
# Association and EAP

- First step is usual 802.11 association to establish L2 connection
- If 802.1X framework used, network unusable unless the show authorization process is complete.




Src: CSWP Text

# Generic EAP Exchange




Src: CWSP Text

# Notes on General Exchange

- Most steps are self explanatory
- Step4: Only identity is sent to the AP in clear text
  - This allows for uncontrolled port to open
- Step8: Supplicant doesn't send password, just MD5 (or other hash of password)
- Step12: A complex process to generate dynamic encryption key
  - uses 4 way handshake (discussed later)
- Step13: Controlled port is unblocked for the user.
  - Proceeds to obtain an IP address using DHCP
- Note: Step4 and 8 create security risk,
  - hash algorithms can be cracked
  - Dictionary attack possible
  - Would it help to have an encrypted tunnel for these steps 4 -9?
- Most schemes use tunneled authentication to pass identity credentials

# Tunneling of EAP

- EAP Methods defined in commonly used modern EAP standards include
  - EAP-TLS (EAP-Transport Layer Security)
    - RFC 5216
  - EAP-SIM (EAP for GSM Subscriber Identity)
    - RFC 4186
  - EAP-AKA (EAP for UMTS Authentication and Key Agreement)
    - RFC 4187
  - PEAP (Protected Extensible Authentication Protocol)
    - RFC 3748, (Microsoft Windows MS-CHAPv2
  - EAP-FAST (Flexible Authentication via Secure Tunneling)
    - RFC 4851
  - EAP-TTLS (EAP-Tunneled Transport Layer Security)
    - RFC 5281
- No need to remember these acronyms or RFCs

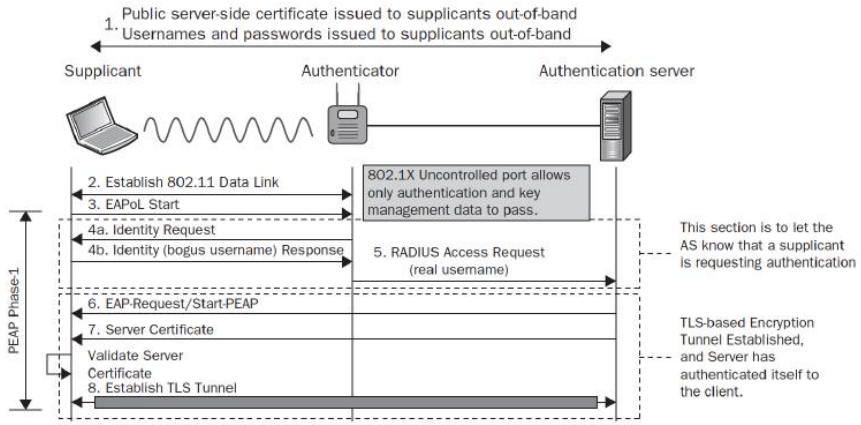
# EAP Tunneled Protocols

- Two supplicant identities used
  - Outer identity: clear text (e.g. anonymous) to satisfy EAP standard (Step4 earlier), a bogus entity
  - Inner identity: true identity that goes inside the encrypted TLS tunnel
- Point to Note:
  - EAP Tunnel only for authentication and authorization to save identity
    - not for encrypting payload
    - Exists for few millisecond
  - Payload encrypted using negotiated key
- Honeypot can be set by using a fake employee name in outer identity
  - employees can inform if they see any activity with this fake name.

# EAP - PEAP

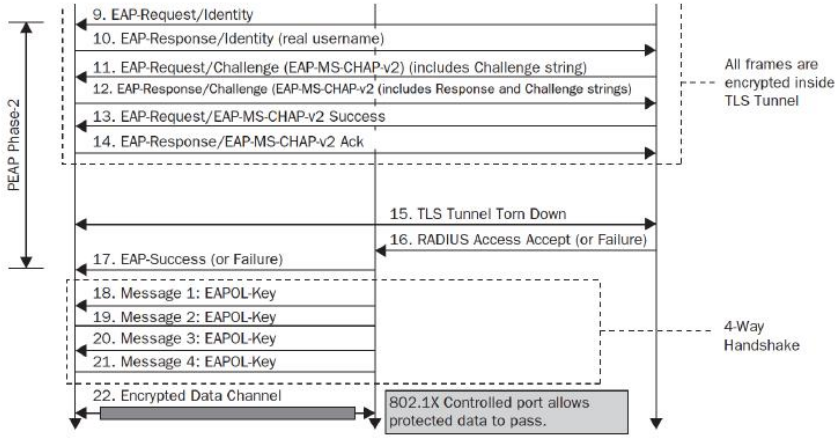
- EAP- Protected Authentication Protocol (EAP-PEAP): most popular, aka "EAP inside EAP"
- Three flavors of this protocol based on inner EAP.
  - Variants from vendors e.g. Microsoft, Cisco – politics of disagreement
  - All need at to establish TLS tunnel
  - Difference in hash algorithms, whether credentials use, tokens, username/password, client-side certificate etc.
  - Read if interested.
- A server side certificate is required for all flavors

# EAP-PEAP Process (Phase1)




Src: CSWP Text

# EAP-PEAP (Phase2)




Src: CSWP Text

# EAP-TTLS

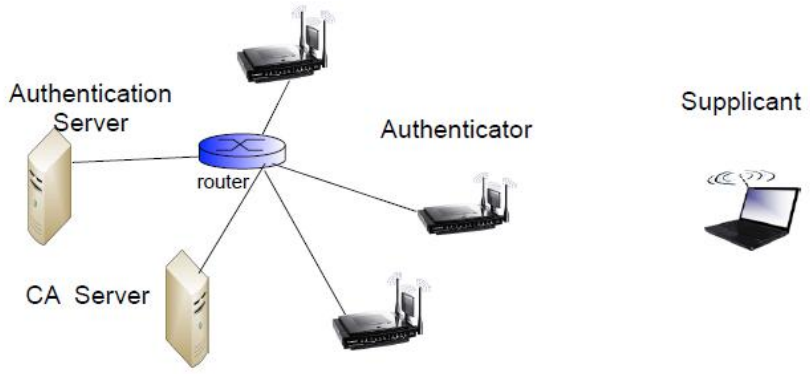
- EAP – Tunneled TTLS
  - Juniper networks mostly deploy this, less popular than PEAP
  - Very similar to PEAP with minor differences
  - Supports many more inner authentication methods, Supplicant credentials normally username and password
  - Optional support for client-side certificates
- Read details if interested

# EAP-TLS

- EAP – Transport Layer Security (TLS)
  - Major differentiator – Client-side certificate
  - Unique digital certificate for each client needs planning, infrastructure
  - Certificate store must be always available and highly secure
- Due to additional cost burden, not a method of choice
- We will skip the details of protocol exchange which has similarity to other methods besides mutual authentication between Supplicant and Server using certificates.

# EAP-TLS Infrastructure

- Deploy Enterprise CA
- Configure Authentication Server (RADIUS)
- Configure Access Point




# EAP-TLS CA configuration

- Clients must be configured via wired (or authenticated Wireless net) before accessing a new network
  - Downloads CA certificate
  - Gets User's certificate
- Configure Radius server
  - Client for Radius is Authenticator (Access point)
  - Setup to accept request from each Access point (including shared secret)
- Configure Access Point
  - Setup Authenticator to the IP address of RADIUS server
  - WPA algorithm e.g. AES
  - Shared secret with Radius
- Operational details/interface etc will vary based on products you use. We are interested in basic architecture

# TABLE 7.1
# IEEE 802.11 TERMINOLOGY

| | |
|---|---|
| Access point (AP) | Any entity that has station functionality and provides access to the distribution system via the wireless medium for associated stations. |
| Basic service set (BSS) | A set of stations controlled by a single coordination function. |
| Coordination function | The logical function that determines when a station operating within a BSS is permitted to transmit and may be able to receive PDUs. |
| Distribution system (DS) | A system used to interconnect a set of BSSs and integrated LANs to create an ESS. |
| Extended service set (ESS) | A set of one or more interconnected BSSs and integrated LANs that appear as a single BSS to the LLC layer at any station associated with one of these BSSs. |
| MAC protocol data unit (MPDU) | The unit of data exchanged between two peer MAC entities using the services of the physical layer. |
| MAC service data unit (MSDU) | Information that is delivered as a unit between MAC users. |
| Station | Any device that contains an IEEE 802.11 conformant MAC and physical layer. |

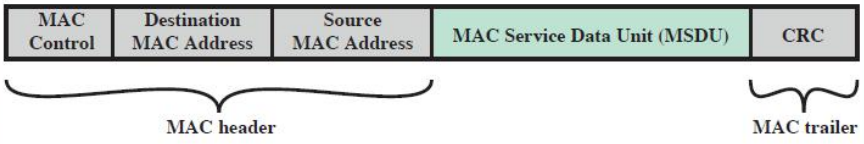
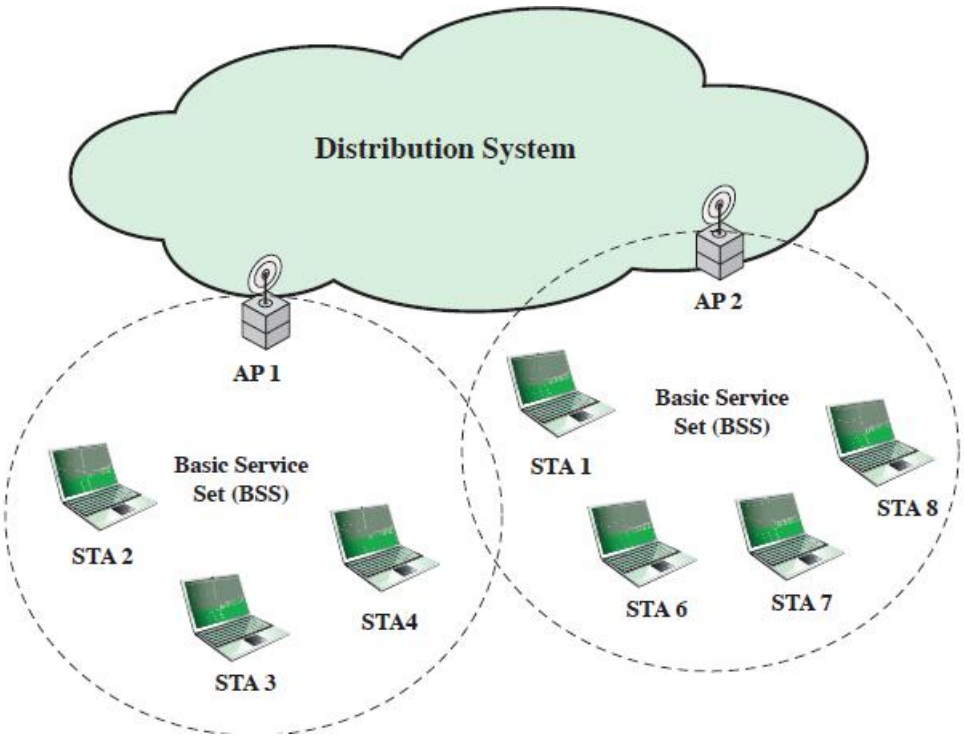


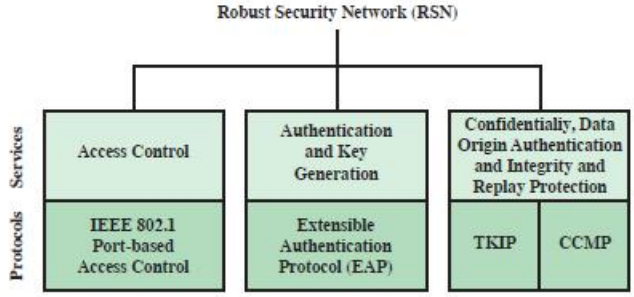

**Figure 7.4 General IEEE 802 MPDU Format**



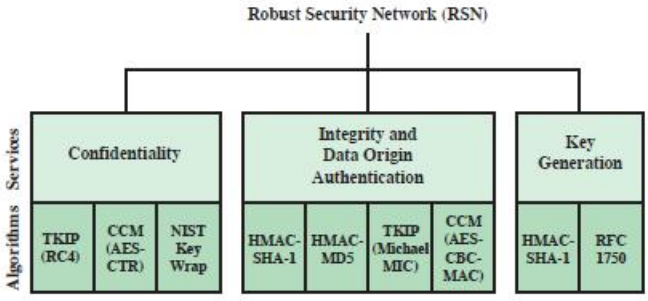
# IEEE 802.11I WIRELESS LAN SECURITY

- There is an increased need for robust security services and mechanisms for wireless LANs

| Wired Equivalent Privacy (WEP) | Wi-Fi Protected Access (WPA) | Robust Security Network (RSN) |
|---|---|---|
| The privacy portion of the 802.11 standard | A set of security mechanisms that eliminates most 802.11 security issues | Final form of the 802.11i standard |
| Contained major weaknesses | Based on the current state of the 802.11i standard | Complex |



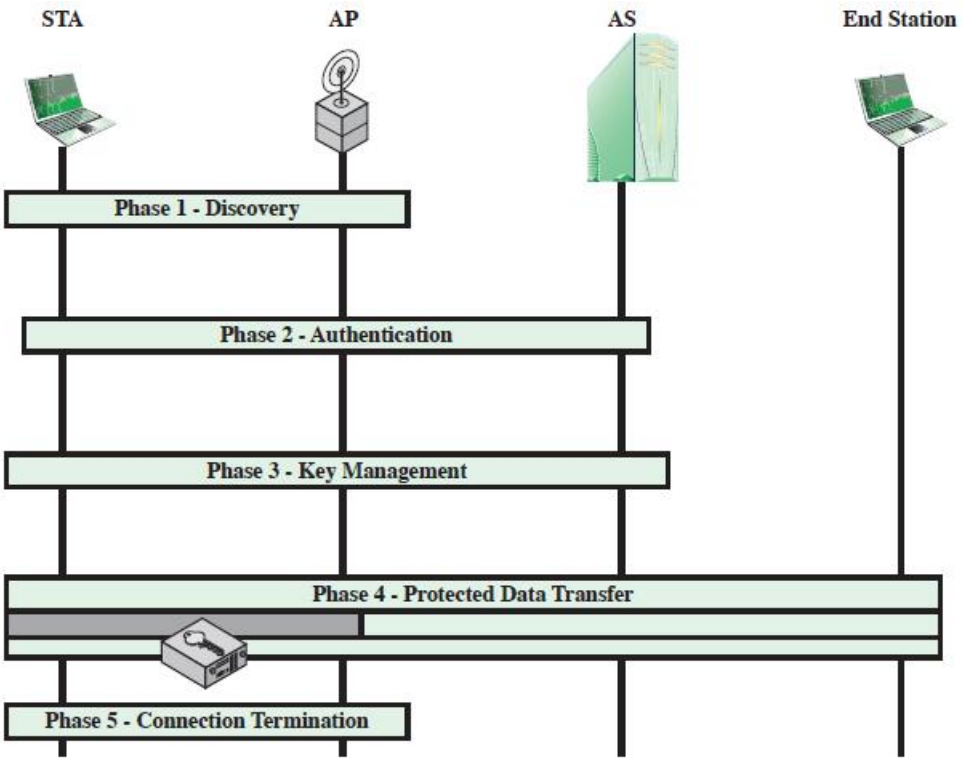

(a) Services and Protocols




(b) Cryptographic Algorithms

CBC-MAC = Cipher Block Block Chaining Message Authentication Code (MAC)
CCM = Counter Mode with Cipher Block Chaining Message Authentication Code
CCMP = Counter Mode with Cipher Block Chaining MAC Protocol
TKIP = Temporal Key Integrity Protocol

**Figure 7.6 Elements of IEEE 802.11i**




Only between STAs to AP. E-2-E security needed outside BSS across Distribution Systems
End Station to AP not secure in this example: For E-2-E Security End station run 802.11i.

**Figure 7.7 IEEE 802.11i Phases of Operation**

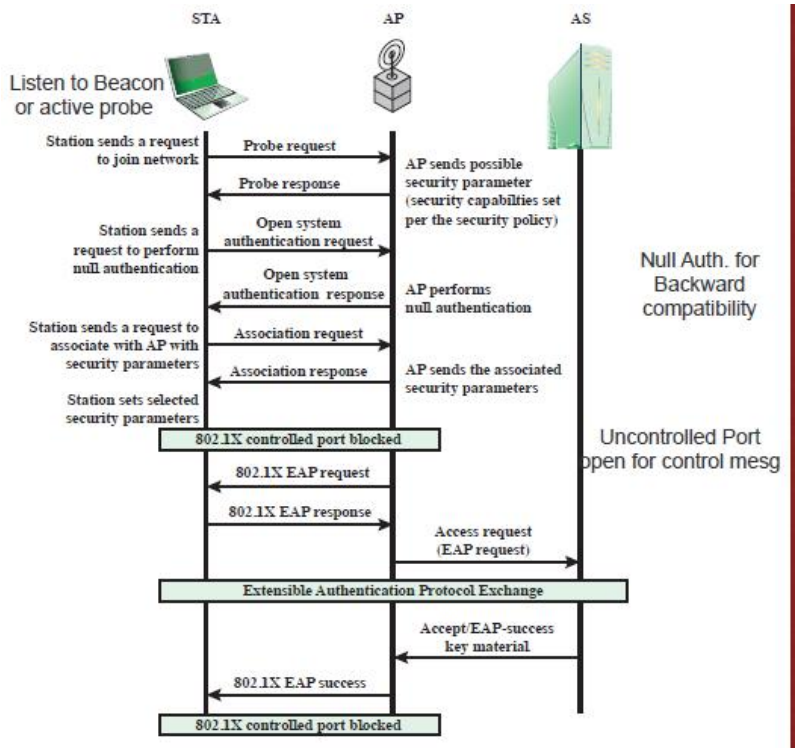
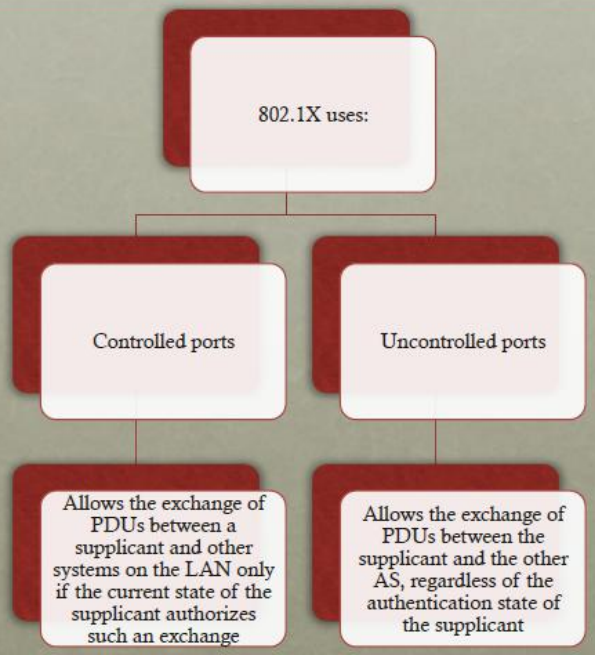



**Figure 7.8 IEEE 802.11i Phases of Operation: Capability Discovery, Authentication, and Association**

# IEEE 802.1X RECAP

- Port-Based Network Access Control
- The authentication protocol that is used, the Extensible Authentication Protocol (EAP), is defined in the IEEE 802.1X standard

STA = Supplicant
AP = Authenticator



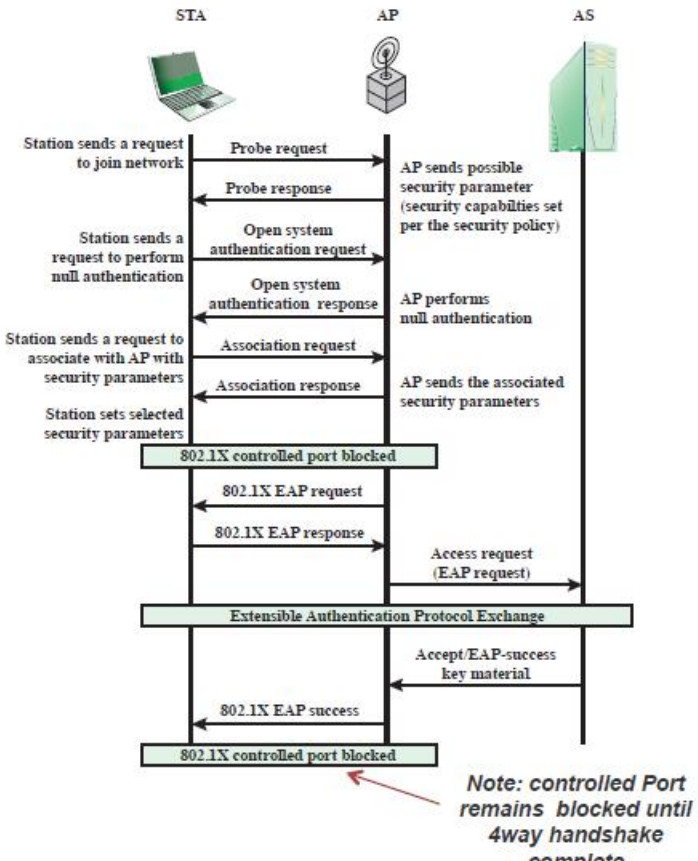
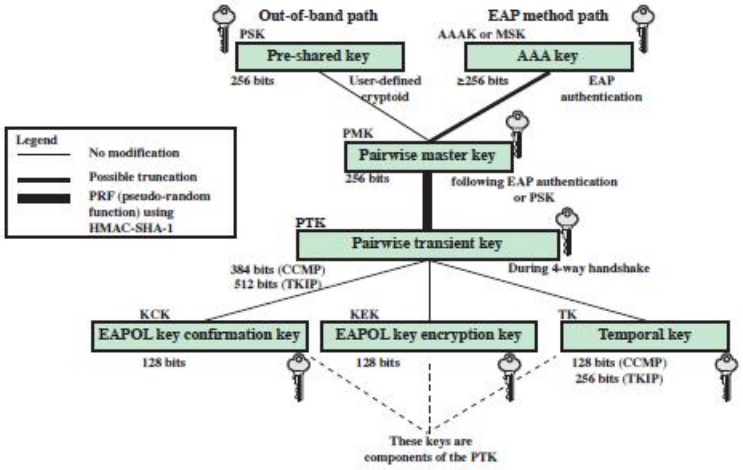
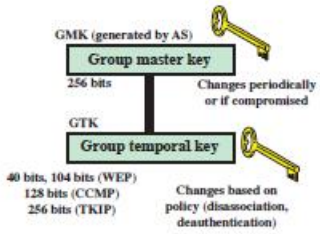




**Figure 7.8 IEEE 802.11i Phases of Operation: Capability Discovery, Authentication, and Association**




(a) Pairwise key hierarchy




(b) Group key hierarchy

| Abbrev-iation | Name | Description / Purpose | Size (bits) | Type |
|---|---|---|---|---|
| AAA Key | Authentication, Accounting, and Authorization Key | Used to derive the PMK. Used with the IEEE 802.1X authentication and key management approach. Same as MMSK. | ≥ 256 | Key generation key, root key |
| PSK | Pre-Shared Key | Becomes the PMK in pre-shared key environments. | 256 | Key generation key, root key |
| PMK | Pairwise Master Key | Used with other inputs to derive the PTK. | 256 | Key generation key |
| GMK | Group Master Key | Used with other inputs to derive the GTK. | 128 | Key generation key |
| PTK | Pair-wise Transient Key | Derived from the PMK. Comprises the EAPOL-KCK, EAPOL-KEK, and TK and (for TKIP) the MIC key. | 512 (TKIP) 384 (CCMP) | Composite key |
| TK | Temporal Key | Used with TKIP or CCMP to provide confidentiality and integrity protection for unicast user traffic. | 256 (TKIP) 128 (CCMP) | Traffic key |
| GTK | Group Temporal Key | Derived from the GMK. Used to provide confidentiality and integrity protection for multicast/broadcast user traffic. | 256 (TKIP) 128 (CCMP) 40, 104 (WEP) | Traffic key |
| MIC Key | Message Integrity Code Key | Used by TKIP's Michael MIC to provide integrity protection of messages. | 64 | Message integrity key |
| EAPOL-KCK | EAPOL-Key Confirmation Key | Used to provide integrity protection for key material distributed during the 4-Way Handshake. | 128 | Message integrity key |
| EAPOL-KEK | EAPOL-Key Encryption Key | Used to ensure the confidentiality of the GTK and other key material in the 4-Way Handshake. | 128 | Traffic key / key encryption key |
| WEP Key | Wired Equivalent Privacy Key | Used with WEP. | 40, 104 | Traffic key |

**Table 7.3**

**IEEE 802.11i Keys for Data Confidentiality and Integrity Protocols**

# PAIRWISE KEYS

- **Used for communication between a pair of devices, typically between a STA and an AP**
  - These keys form a hierarchy beginning with a master key from which other keys are derived dynamically and used for a limited period of time
- **Pre-shared key (PSK)**
  - A secret key shared by the AP and a STA and installed in some fashion outside the scope of IEEE 802.11i
- **Master session key (MSK)**
  - Also known as the AAAK, and is generated using the IEEE 802.1X protocol during the authentication phase
- **Pairwise master key (PMK)**
  - Derived from the master key
  - If a PSK is used, then the PSK is used as the PMK; if a MSK is used, then the PMK is derived from the MSK by truncation
- **Pairwise transient key (PTK)**
  - Consists of three keys to be used for communication between a STA and AP after they have been mutually authenticated
  - Using the STA and AP addresses in the generation of the PTK provides protection against session hijacking and impersonation; using nonces provides additional random keying material

# PTK PARTS

- The three parts of the PTK are:

**EAP Over LAN (EAPOL) Key Confirmation Key (EAPOL-KCK)**
- Supports the integrity and data origin authenticity of STA-to-AP control frames during operational setup of an RSN
- It also performs an access control function: proof-of-possession of the PMK

**EAPOL Key Encryption Key (EAPOL-KEK)**
- Protects the confidentiality of keys and other data during some RSN association procedures

**Temporal Key (TK)**
- Provides the actual protection for user traffic

# GROUP KEYS

- Group keys are used for multicast communication in which one STA sends MPDUs to multiple STAs
  - Group master key (GMK)
    - Key-generating key used with other inputs to derive the GTK
  - Group temporal key (GTK)
    - Generated by the AP and transmitted to its associated STAs
    - Distributed securely using the pairwise keys that are already established
    - Is changed every time a device leaves the network

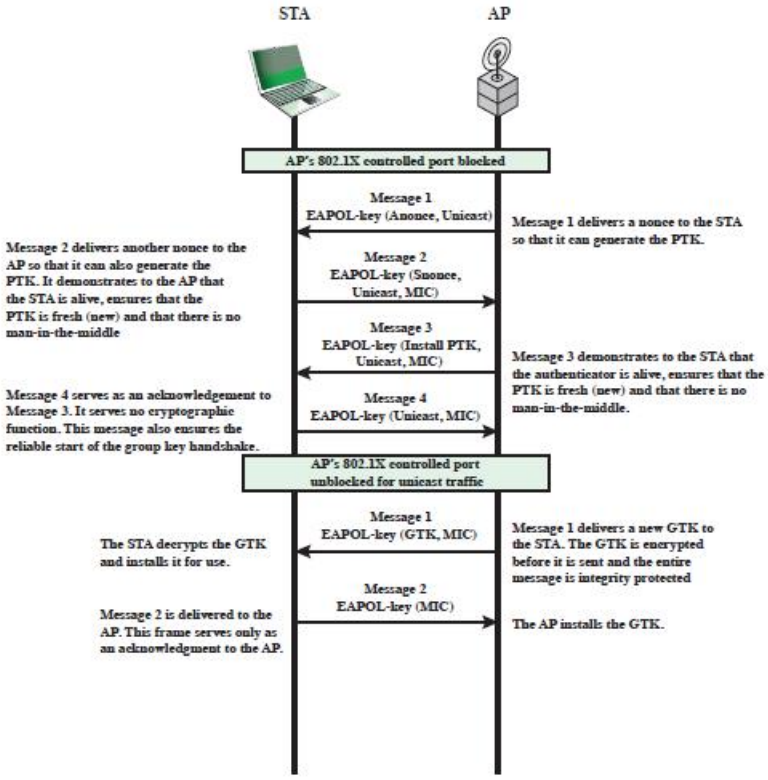
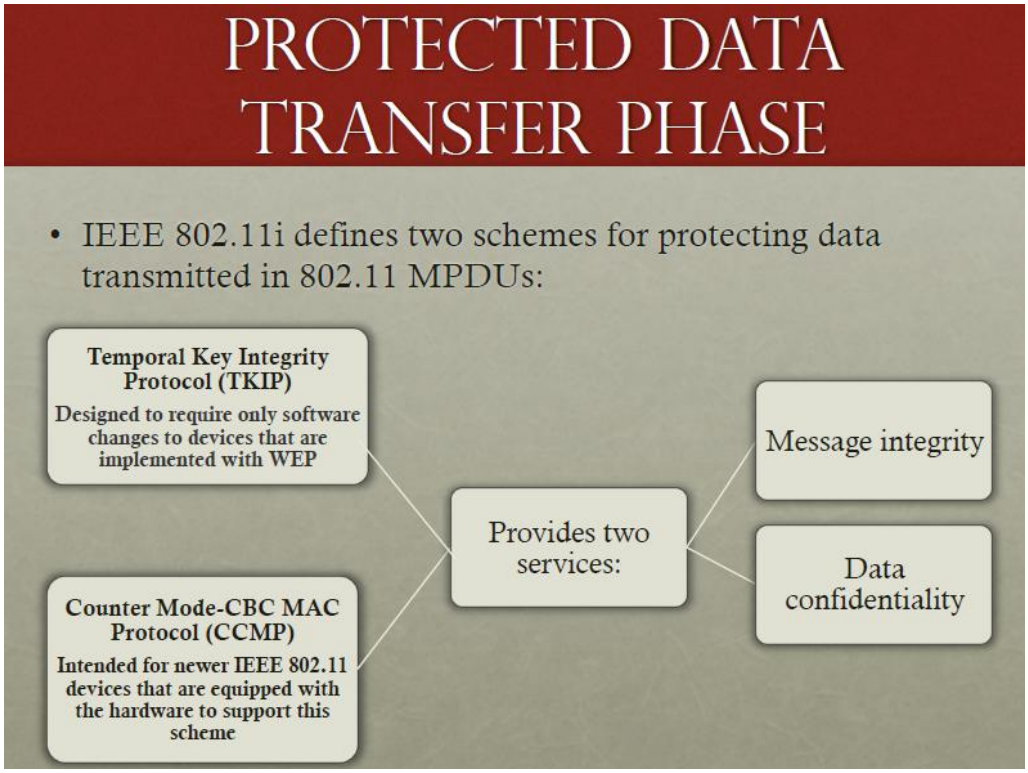



**Figure 7.10 IEEE 802.11i Phases of Operation: Four-Way Handshake and Group Key Handshake**



**Bluetooth Security**

# Bluetooth Evolution

- Bluetooth Special Interest Group (SIG)
- Founded in Spring 1998
- By Ericsson, Intel, IBM, Nokia, Toshiba
- Now more than 2,000 organizations have joined the SIG

# Features

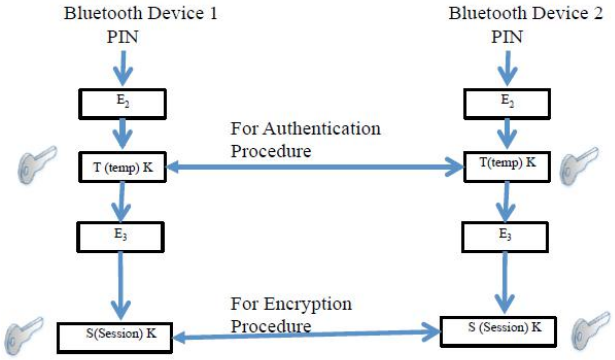
- Bluetooth-enabled devices can automatically locate each other
- Topology is established on a temporary and random basis
- Up to eight Bluetooth devices may be networked together in a master-slave relationship to form a piconet

- One is master, which controls and sets up the network (piconet)
- Two or more piconet interconnected to form a scatter net
- Only one master for each piconet
- A device can't be masters for two piconet
- The slave of one piconet can be the master of another piconet

# Security Issues

- Authenticity: Are you the device you claim you are?
  - Impersonation
- Confidentiality: Is the exchanged data only available to the intended devices?
  - Packet sniffing
- Authorisation: Are only the intended devices accessing the specified data and control?
  - Prerequisite: authenticity and confidentiality

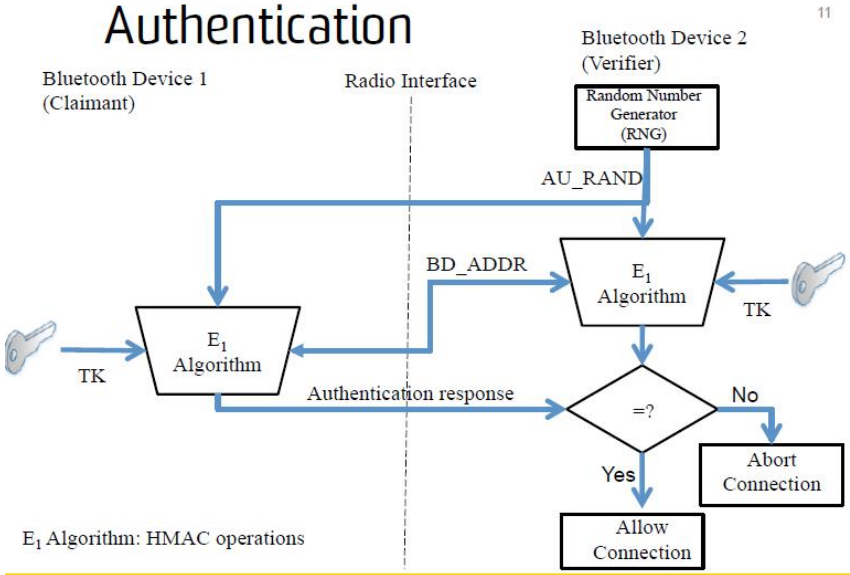
# Temporary Key Generation




$E_1$: HMAC algorithm
$E_2$: it would be discussed in case study
$E_3$: it is described in slide 13.

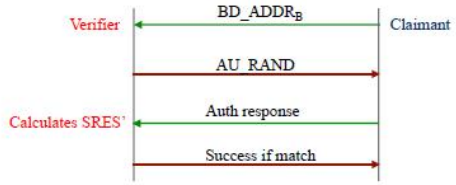
# Authentication






$E_1$ Algorithm: HMAC operations

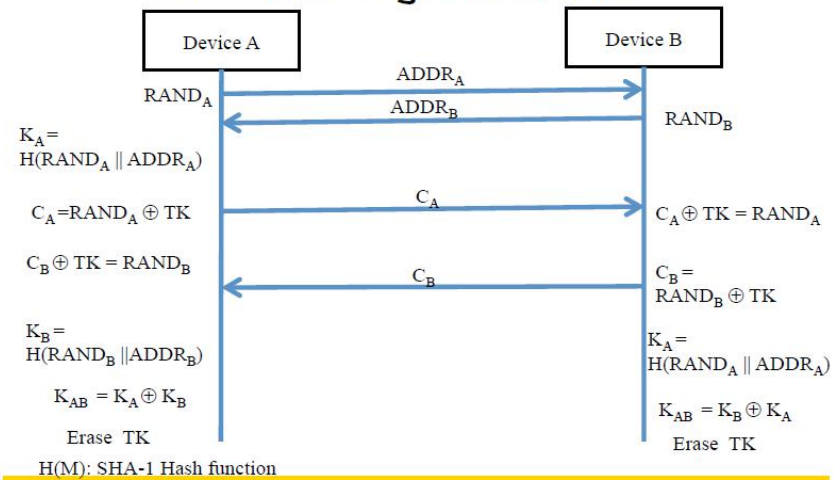
# Authentication Summary




Authentication Process

| Parameter | Length | Secrecy parameter |
|---|---|---|
| Device Address | 48 Bits | Public |
| Random Challenge | 128 Bits | Public |
| Authentication(Auth) Response | 32 Bits | Public |
| Temporary Key | 128 Bits | Secret |

# Session Key Generation




# Is Channel Hopping Secure?

- Channel Hopping (Bluetooth Smart only) – Both communication parties would hop to a different wireless channel per packet in a fixed channel hopping increment.
- Adversary could not achieve data by monitoring one wireless channel only.
- We will study its vulnerability soon.

# Case study: Bluetooth Low Energy (BTLE)

- Introduced in Bluetooth 4.0 (2010)
- New modulation and link layer for low power devices
  - Incompatible with classic Bluetooth devices
  - PHY and link layer different (no channel hopping in classic Bluetooth)
  - High-level protocols reused (L2CAP, ATT)

# BTLE applications

- High end smart phones
- Sports/fitness devices
- Door locks
- Upcoming medical devices (e.g., blood glucose monitor)

# BTLE Protocol review

| GATT (Generic Attribute Profile) – how to discover and provide services based on ATT |
|---|
| ATT (Attribute protocol) – how to discover/read/write attributes on a peer device |
| L2CAP (Logical Link Control and Adaptation Protocol) – packet segmentation and reassemble |
| Link Layer |
| Physical Layer |

- We will focus on Link layer and Physical layer security only.
- Other layers are similar to their counterparts in wired networks.

# Physical Layer

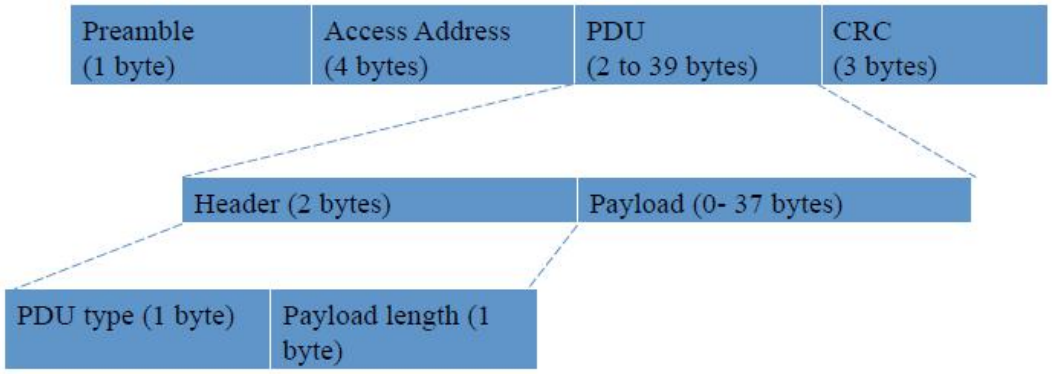
- Physical layer: channels for hopping (40 available channels in 2.4Ghz)
  - Advertising: 3 channels
  - Data: 37 channels

## Channel Hopping

- Hop along 37 data channels
- One data packet per channel
- Next channel = current channel + hop increment (mod 37)
- Time between hops: hop interval, it is the duration when both communication parties stays in one channel. It is equal to one Round Trip Time + channel switch latency.

3→10→17→24→31→1→8→15→…(hop increment = 7)

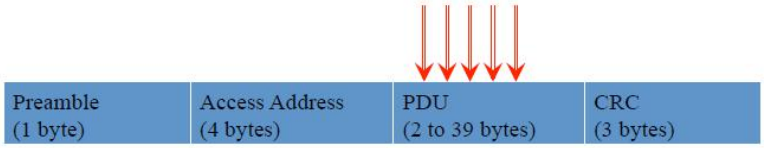
# Link Layer




- Only PDU encrypted which creates security vulnerability
  - packet sniff to break the confidentiality in PDU.

# Encryption and MACs

- Encrypts and MACs PDU section
- AES-CCM algorithm
- AES-CCM is secure but the key exchange protocol is weak!




# Packet Sniff Process

- Configuration
  - Set modulation parameters to match BTLE (e.g., set to the same frequency, 2.4GHz)
  - Tune to proper channel – it needs to know the channel hopping pattern.
    - Hop Increment
    - Hop interval
    - Both can be sniffed from connection packet or recovery in promiscuous mode

# What Information do we need?

| Preamble (1 byte) | Access Address (4 bytes) | PDU (2 to 39 bytes) | CRC (3 bytes) |
|---|---|---|---|

- Access Address (AA)
  - Advertising: Fixed 0x8E89BED6
  - Connection: Actual device address
- Channel Information:
  - Hop interval
  - Hop increment

Where to get this info: Connection packet!

- easy if you get the starting packets

## Promiscuous mode

- What if I missed the connection packets?
  - Capture a number of data packets.
  - Perform the pattern search (promiscuous mode) to recover access addresses, hop interval and hop increment values (easily done!)
- Crack the session key to decrypt PDU

# Recovery of Access Address

| Preamble (1 byte) | Access Address (4 bytes) | PDU (2 to 39 bytes) | CRC (3 bytes) |
|---|---|---|---|

What we know: Preamble (01010101)

What we have: Sea of bits

What we want: Access Address

10001110111101010101 → likely preamble!

10011100000100011001.. -> part of AA

100011001…100011101 →32 bit complete of AA! After that, PDU!

- A preamble is "01010101" but "01010101" is not always a preamble.
- CRC is here to help (for attacker)!
- Attacker could use CRC (after PDU) to verify the access address and PDU.
  - If CRC passes, the access address is correct. Otherwise, the "01010101" is false positive for preamble.

## Recovery of Hop Interval

- Observation: 37 is a prime
- Sit on one data channel and wait for two consecutive packets. Measure the time difference.

$$\Delta t/37 = \text{hop interval}$$

## Recovery of Hop Increment

- Start on data channel 0, jump to data channel 1 when a packet arrives.
- We know hop interval, we can calculate how many channels have been hopped between channel 0 and 1.
  - $\Delta t$/hop interval = channel hops

## Calculate Hop Increment

$$\text{HopIncrement} * \text{channel hops} \equiv 1 \pmod{37}$$

$$\text{HopIncrement} \equiv \text{channel hops}^{-1} \pmod{37}$$

Apply Fermat's little theorem : $a^{p-1} \equiv 1 \pmod{p}$,

$$\text{HopIncrement} \equiv \text{channel hops}^{37-2} \pmod{37}$$

# Sniff summary

- Connections packets
- Promiscuous mode: recovery of
  - Access Address
  - Hop Interval
  - Hop Increment



# Custom Key exchange protocol

- Three pairing methods
  - Just Works™
  - 6-digit PIN
  - 00B
  - "None of these key pairing methods provide protection against a passive eavesdropper" – Bluetooth Core Spec

# Cracking Temporary Key

- Temporary Key (TK) = AES (PIN, AES(PIN, rand XOR p1) XOR p2) ($E_2$ in slide 10)

Green – transmitted in plaintext

Red – wanted to know

PIN: integer between 0 and 999,999

JustWork™ is always 0!

## Cracking the PIN

# Total Time to crack:

# < 1 second

### Subsequent Key crack

PIN → STK
STK → LTK
LTK → Session key!

•Every key is known.
•Attacker can learn about PDU
•Attacker can inject the packets in the networks with the session key!

---

**Wireless Broadcast**

## Datagram TLS (DTLS)

- SSL Designed to run on top of TCP
- Datagram TLS developed later to run over connectionless UDP
  - RFC 4347 for details
- Already supported by several implementations
- Very similar to TLS
  - Needs extra control messages as UDP doesn't provide these like TCP
  - Sequence number in record header to protect from Replay attack
- If very lossy network, may have issues with lot of restranmissons for reliability

# Elliptic Curve (ECC) Scheme

- Key Agreement, aka, Elliptic Curve Deffie-Hellman (ECDH)
  - Allows for establishment of shared secret similar to DH
  - The shared key is then used for symmetric encryption or for further session/temporal key derivation
- Digital Signature: Elliptic Curve Digital Signature Algorithm (ECDSA), allows use of public/private key for signing a message and verification of signature, more efficient than RSA based DSA.

# Elliptic Curve Cryptography

- Elliptic curve cryptography (ECC) is an approach to public-key cryptography based on the algebraic structure of elliptic curves over finite field.
- ECC presents various benefits over RSA such as:
  - fast computation
  - small key size
  - compact signatures.
- For example, to provide equivalent security to 1024-bit RSA, an ECC scheme only needs 160 bits.

## Elliptic Curve Diffie-Hellman Key Exchange

1. Alice and Bob publicly agree on an elliptic curve E over a finite field $Z_p$.
2. Next Alice and Bob choose a public *base point* B on the elliptic curve E.
3. Alice chooses a random integer $1<\alpha<|E|$, computes $P = \alpha B$, and sends P to Bob. *Alice keeps her choice of $\alpha$ secret.*
4. Bob chooses a random integer $1<\beta<|E|$, computes $Q = \beta B$, and sends Q to Alice. *Bob keeps his choice of $\beta$ secret.*

1. Alice and Bob choose E to be the curve $y^2 = x^3+x+6$.
2. Alice and Bob choose the public base point to be B=(2,4).
3. Alice chooses $\alpha = 4$, computes $P = \alpha B = 4(2,4) = (6,2)$, and sends P to Bob. *Alice keeps $\alpha$ secret.*
4. Bob chooses $\beta = 5$, computes $Q = \beta B = 5(2,4) = (1,6)$, and sends Q to Alice. *Bob keeps $\beta$ secret.*

5. Alice computes $K_A$ = $\alpha Q = \alpha(\beta B)$.
6. Bob computes $K_B = \beta P = \beta(\alpha B)$.
7. The shared secret key is $K = K_A = K_B$.

- Even if Eve knows the base point B, or P or Q, she will not be able to figure out $\alpha$ or $\beta$, so K remains secret!

5. Alice computes $K_A = \alpha Q = 4(1,6) = (4,2)$.
6. Bob computes $K_B = \beta P = 5(6,2) = (4,2)$.
7. The shared secret key is K = (4,2).

## Recap: How to use keys?

- Rule of thumb:
- Public Key Cryptography: slow
- Symmetric Cryptography: fast

- Hence, do not encrypt large messages with Public Key Cryptography
- Encrypt a random, fresh symmetric key with Public Key Cryptography
- Use this key and symmetric encryption to encrypt a large message

- For signature, only sign the hash value of messages

- Send the encrypted key, signature, and symmetrically encrypted message to your communication partner

# Challenges for Broadcast Security

- Broadcast applications need security
  - Packet injection or eavesdropping is easy
- Security solutions for point-to-point communication not suitable secure for broadcast
- Broadcast challenges
  - Scale to large audiences
  - Dynamic membership
  - Low overhead (computation & communication)
  - Packet loss
    - How to achieve reliability in broadcasts?

# Scale & Dynamics

- Small groups contain up to ~100 members
- Medium-size groups contain 100-1000 members
- Large groups contain 1000-$10^9$ members – e.g. IoT
- How does scale affect security?
- Dynamic membership: members may join and leave at any time
- How do dynamics affect security?

# Communication Pattern

- Group can be single-source broadcast
  - One-to-many
  - SSM: Single-source multicast, source-specific multicast
- Multiple-source broadcast
  - Some-to-many
- All members broadcast
  - Many-to-many

# Reliable Broadcast Transmission

- How to reliably and scalably disseminate data to large numbers of receivers?
- Challenges
  - Ack implosion problem if receivers return Ack to sender for received packets
  - Nack implosion problem is severe as well
  - For large numbers of receivers, there usually is a fraction of them that do not obtain message
  - Local repair mechanisms (create tree topology and ask upstream parent for packet) faced numerous scalability difficulties
    - Accumulate ack (delayed) and send to parent node on the tree.

# End-to-End approach ?

- Trusted server authenticates each node and distributes the key.
- Use well-known E2E protocols such as (D)TLS, IPSEC, SSL.

**Pros**

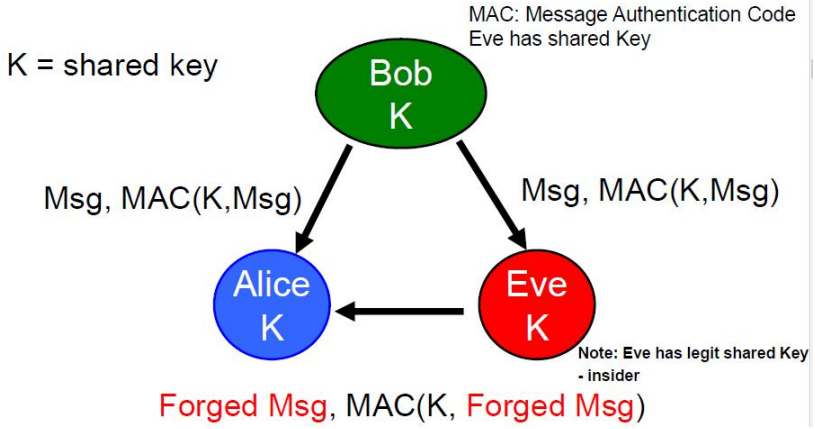
- Higher security level
- Dynamic grouping

**Cons**

- Extremely higher ciphertext overhead due to one by one transfer
- Higher computation requirements

## Shared Key: Easy to Forge






## Asymmetric Key: Digital Signature

- Sign each packet and verify using Asymmetric key
- However, Signatures are expensive esp. for low end processors, e.g., RSA 2048:
  - High generation cost (~1 millisecond), High verification cost (~0.1 millisecond), High communication cost (256 bytes/packet)
- If we use one signature over multiple packets, intolerant to packet loss

# Trivial broadcast key distribution

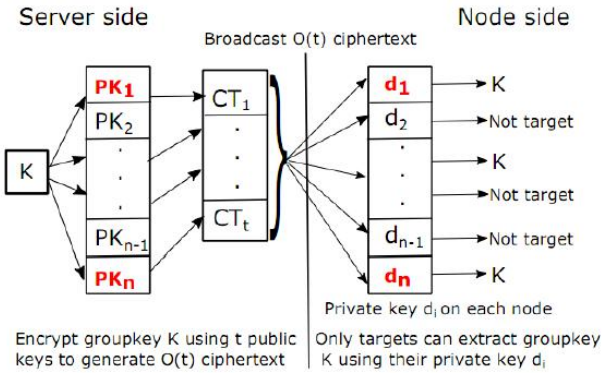



Figure 1: Trivial Broadcast Encryption scheme in an $n$ nodes network targeting $t$ nodes. K: shared group key. PK: Public Key. CT: Ciphertext. d: Decryption using each node's private key.

# Trivial (2)

- Very simple and effective system. Easy to implement.
- The only issue is linear ciphertext O(t) for key distribution.
- Good for applications with smaller number of target nodes.
- Possible solution for non-sensor applications such as Desktop, smart-phone
- Optional reading: Stinson, Douglas R. *Cryptography: theory and practice*. CRC press, 2005.

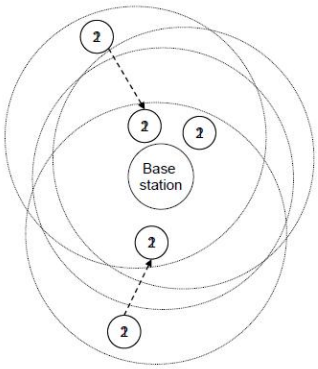
# Broadcast Authentication Protocol

- *k* different keys to authenticate every message with *k* different MAC's
- Each receiver knows *m*-keys and hence verifies m MAC's.
- Key distribution in such a way that no coalition of *w* receivers can forge a packet for a specific receiver
- Assumptions on number of colluding receivers (on the order of *k*)
- Refer to (optional read, lot of work in crypto)R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas. Multicast security: A taxonomy and some efficient constructions. In *INFOCOMM'99*, pages 708–716, March 1999.

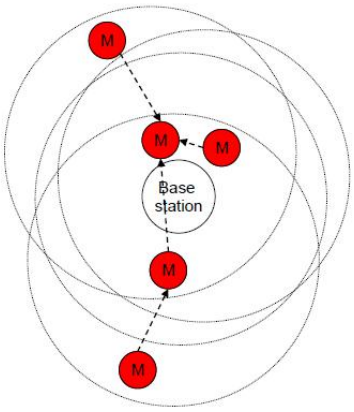
# Hash Chain -basics

- Client generate 1000 hashes for password
  - Suggested by Lamport for password protection
- Server stores $H^{1000}$(password)
- Client willing to authenticate sends $H^{999}$(password)
- Server computes $H^{1000}$(password) = H($H^{999}$(password))
  - Match found, store $H^{999}$(password)for next time
- Eavesdropper can't use $H^{999}$(password) since server expects $H^{998}$(password)

# Case Study: Multi-hop code distribution

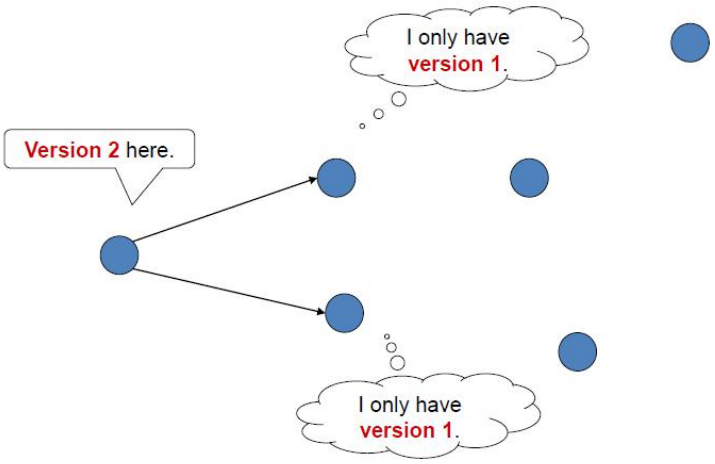



# Attack by a compromised node

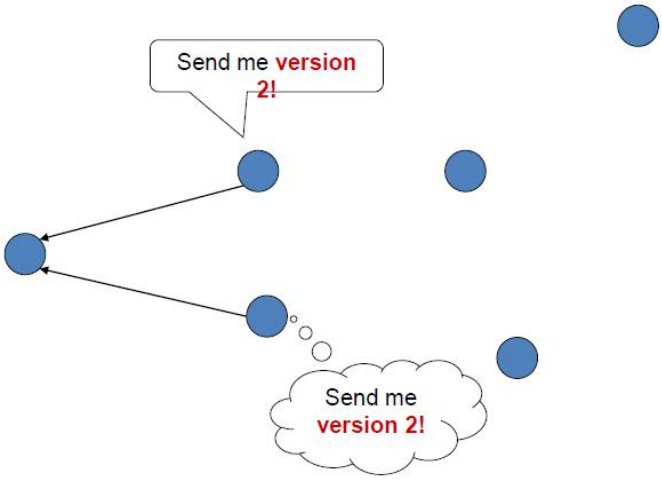


# Epidemic propagation (1)

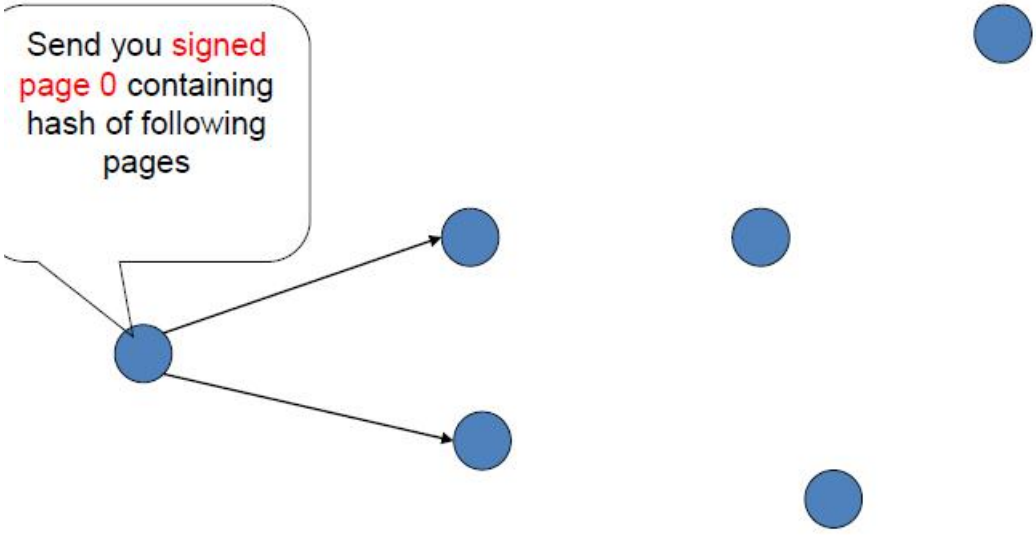
## 1. Nodes periodically advertise



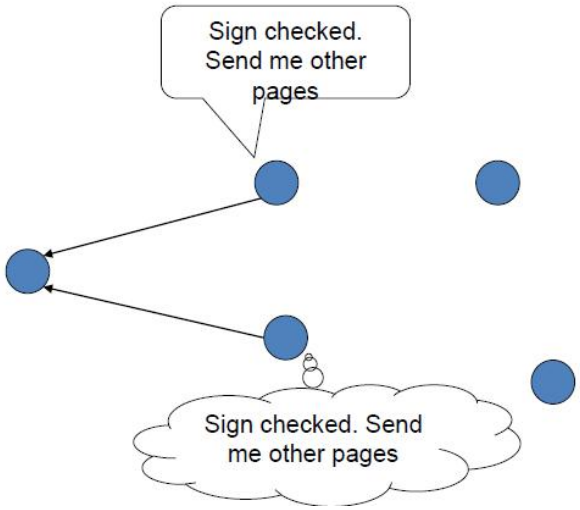

## 2.Neighboring nodes request new version



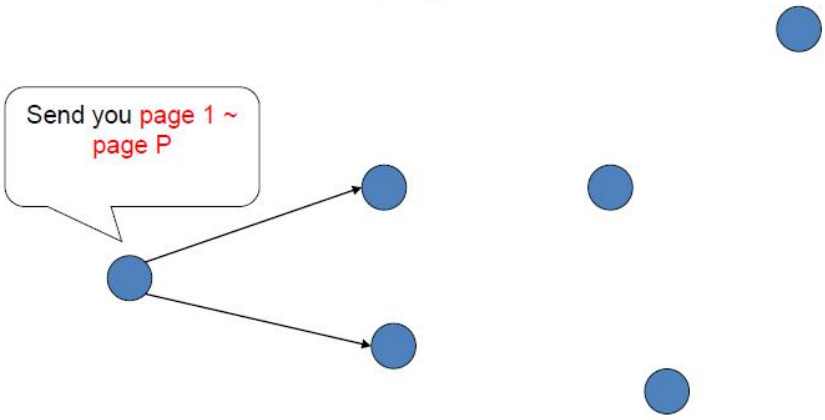

## 3. Server sends signed hash of each page



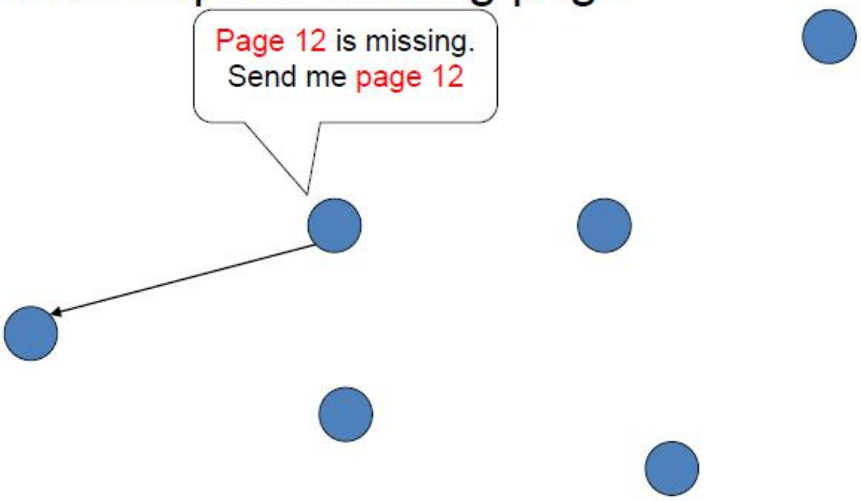
## 4.Neighboring nodes request data



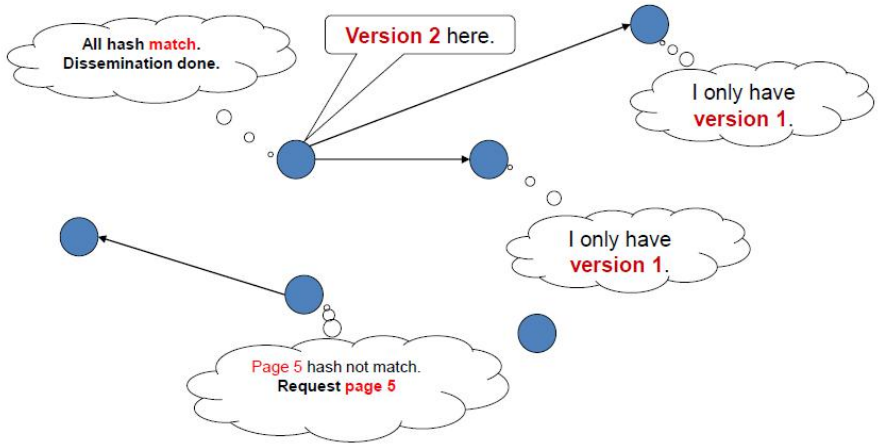

## 5. Transmit each page




## 6.Nodes request missing page



7.Check hash of each page




# Threat Model

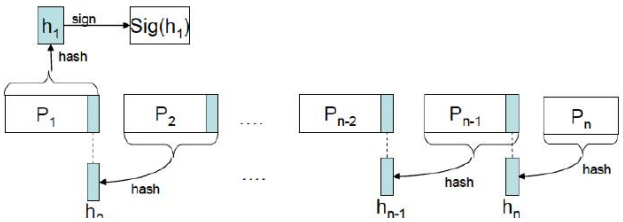
- Adversary has power laptop/computer
- External Attacks: adversary doesn't control any node
  - Eavesdrop, inject forged messages, replay intercepted messages, impersonate valid node, *Wormhole Attack* (fake non-existing links ), *Sybil Attack* (one node presents several identities to defeat fault tolerance)
- Internal Attacks: take control of nodes
  - manipulate nodes until detected, intercept sensitive information even if encrypted, (selectively) drop packets, and launch Sybil attacks
- Adversary can distribute illegal code, drain battery, disconnect network etc.

# Setting: Metrics

- Security metrics
  - Can external adversary forge a message?
  - Can single or several receivers forge a message that at least one other receiver accepts?
- Efficiency metrics
  - Communication overhead
  - Computation overhead
  - Storage overhead
  - Delay for authentication / signature
  - Resilience to packet loss

# Code Dissemination – Hash Chain

- Image fragmented into fixed size segments – called pages
- Calculate hash value ($h_n$) of last page $P_n$ and append this to previous page $P_{n-1}$ , until first page is reached
- Hash value of first page P1 signed with private key of base station (BS)
- Receivers verify this using public key of BS and recursively authenticate each page

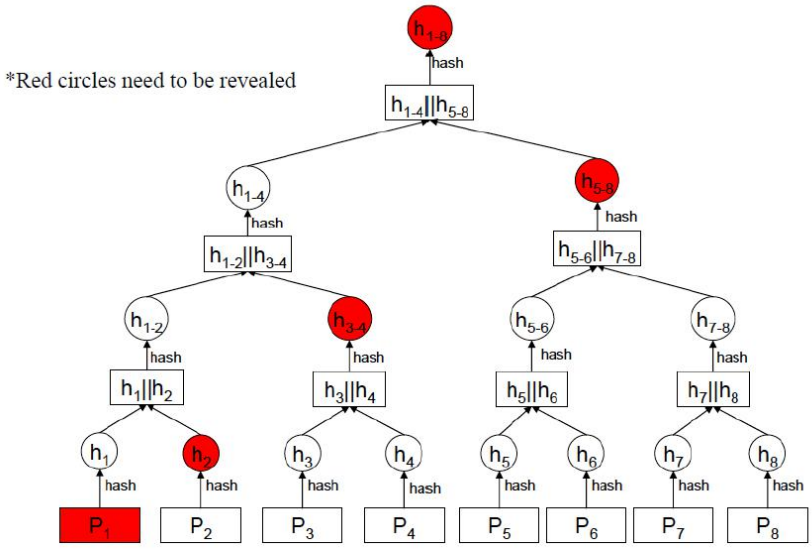



# Merkle Hash Tree

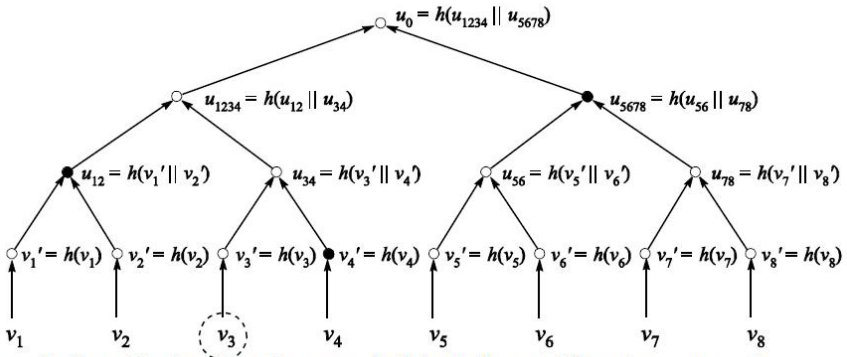
- Hash Chain elements can only be revealed sequentially
  - May not be acceptable for many applications e.g. P2P
- Merkle-trees : allowing for the pre-authentication of a set of values with a single digital signature (on the root $u_0$ of the tree) and for the revelation of those values in *any* order
- when revealing a value $v_i$, reveal all the values assigned to the sibling vertices on the path from $v_i$' to the root
  - One way hash property ensures: this disclosure not sufficient to calculate any other unrevealed $v_j$

# Code Dissemination-MT Example

*Red circles need to be revealed



# Merkle Tree Example




- Authenticate $v_3$, $v_3$ is revealed together with $v_4'$, $u_{12}$, $u_{5678}$)
- verifiers hash the revealed values appropriately and check if the result is $u_0$
  - $u_0 = H( H( u_{12} || H( H(v_3) || v_4' )) || u_{5678} )$

# Code Dissemination - MT

- We could further divide each page into packets
- Take hash of each packet
- Construct a Merkle tree of these hash values and get root for each page
- Recursive hash value of the roots of each page is done using Hash-Chain
- Digital signature for the first root value is sent
- Along with root value, also need sibling vertices as discussed earlier

# Signature Based Attack

- Signature expensive operation on small embedded devices
- Attacker keeps sending forged data
- Node depletes energy in doing signature verification
- These approaches need to know number of data-packets in advance
  - may not be available in many applications

# Small RSA Signature

- Idea
  - Use short-lived small RSA keys (e.g., 384 bit)
  - Periodically send out new public key signed with strong signature
    - 48 byte signature per packet
    - Signature generation ~0.1ms, verification ~10us
- Advantages
  - Relatively low computation overhead
  - No buffering, no verification delay
  - Scalable
- Disadvantages
  - Relatively high communication overhead (> 50 bytes/packet)
  - Need time synchronization
  - Not perfectly robust to packet loss

# Sample Question1

Consider the following pseudo-WEP protocol. The key is 4 bits and the IV is 2 bits. The IV is appended to the end of the key when generating the keystream. Suppose that the shared secret key is 1010. The keystreams for the four possible inputs are as follows:

101000: 00101011010101010011011010100100 . . .
101001: 10100110110010101110100100101101 . . .
101010: 00011010001111000101001010011111 . . .
101011: 11111010000000000101010100010111 . . .

Suppose all messages are 8-bits long. Suppose the ICV (integrity check) is 4-bits long, and is calculated by XOR-ing the first 4 bits of data with the last 4 bits of data. Suppose the pseudo-WEP packet consists of three fields: first the IV field, then the message field, and last the ICV field, with some of these fields encrypted.

We want to send the message m = 10100000 using the IV = 11 and using WEP. What will be the values in the three WEP fields?

# Sample Question1 - Solution

*Answer:*
*Since IV = 11, the key stream is 111110100000 …….*

*Given, m = 10100000*
*Hence, ICV = 1010 XOR 0000 = 1010*
*The three fields will be:*

*IV: 11*
*Encrypted message: 10100000 XOR 11111010 = 01011010*
*Encrypted ICV: 1010 XOR 0000 = 1010 (remember the key bits from 9-12 above are all 0000.*
•

# Sample Question-2

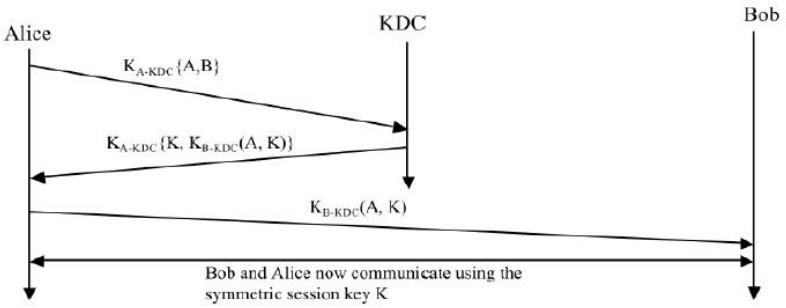
Suppose Alice wants to communicate with Bob using Symmetric Key Cryptograpy using a session key Ks. In this question we use a Key Distribution Centre in place of Public Key Cryptography. KDC shares a unique secret symmetric key with each registered user. For Alice and Bob, denote these keys by $K_{A-KDC}$ and $K_{B-KDC.}$ Design a scheme that uses the KDC to distribute the session key: A message from Alice to the KDC; A message from KDC to Alice; and finally a message from Alice to Bob. The first message is $K_{A-KDC}$ (A,B). Using the notation $K_{A-KDC}$ , $K_{B-KDC}$ , S, A, and B answer the following questions:

A) What is the second message?

B) What is the third message?

(Exam question may be longer a bit more complex)

# Sample Question-2 Solution




First message; Request a session key with Bob from KDC
Second Message: KDC to Alice, sends Key as well as the key encrypted with Bob's shared key with KDC for Bob to verify that it is from a legitimate source.
Third Message: Alice sends this encrypted key to Bob (saying that use this key for session With A (Alice)

Correction: Curly braces should Be parenthesis in first message

# Sample Question 3

- Suppose Bob initiates a TCP connection to Trudy who is pretending to be Alice. During the handshake, Trudy sends Bob Alice's certificate. In what step of the SSL handshake algorithm will Bob discover that he is not communicating with Alice?

# Sample Question 3 - Solution

*Answer:*

After the client will generate a pre-master secret (PMS), it will encrypt it with Alice's public key, and then send the encrypted PMS to Trudy. Trudy will not be able to decrypt the PMS, since she does not have Alice's private key. Thus Trudy will not be able to determine the shared authentication key. She may instead guess one by choosing a random key. During the last step of the handshake, she sends to Bob a MAC of all the handshake messages, using the guessed authentication key. When Bob receives the MAC, the MAC test will fail, and Bob will end the TCP connection.