



Protocol Security

Protocol Design



Why Protocol Security?

- ▶ Expectation on ICT systems: **Dependability**
 - More and more mission-critical tasks are moved to ICT
- ▶ Problem: Bugs, Crashes, Failures, Malfunctions
- ▶ Problem: **Malice**
 - Protection against Malice may also help against bad coincidences



How much security?



Some terminology

- ▶ A **system** is designed with **security objectives** in mind
- ▶ Real systems have **weaknesses**
- ▶ **Vulnerabilities** allow circumvention (or misuse!) of security mechanisms

- ▶ A **threat** is the potential for an **attack**
- ▶ Attacks may create **damage**
- ▶ **Risk** = $p(\text{attack}) \times \text{cost}(\text{damage})$



Security systems

- ▶ Security systems control attacks by:
 - *prevention*
 - *detection*
 - *containment*
- ▶ This is based on an underlying **security policy**
 - Rules and regulations, training of employees
 - Emergency planning, training
 - Management support (including protection of security personnel)



Who are the attackers?

- ▶ **Insiders** (lazy, frustrated, criminal)
 - Possibly implicated in **Social Engineering**
- ▶ **„Hackers“** (Crackers), „script kiddies“
 - Pure curiosity, Fun/Suspense/Addiction, Craving for recognition!
- ▶ **Professional Attackers** (espionage, secret services)
- ▶ **Organized Crime**
 - E.g., blackmail
 - E.g., damaging a competitor



Security Objectives

- ▶ Confidentiality, access control (read), Privacy
 - Special case: Anonymity
- ▶ Integrity/Authenticity, access control (write)
- ▶ Accountability/Non-repudiability
- ▶ Availability



Confidentiality, access control (read), Privacy

- ▶ **secrecy**: restricting (read) access to authorized principals
- ▶ **confidentiality**: (often used in the sense of secrecy)
obligation to keep secret
- ▶ **privacy**: right to secrecy of personal information

- ▶ Special case: The fact that communication occurred at all is often also a subject of confidentiality (vs. **traffic analysis**)
- ▶ Special case:
Anonymity: Principal can act without giving away **identity**
 - Possibly giving away a **pseudonym**



Integrity/Authenticity, access control (write)

- ▶ **Integrity** of data: protection against **unauthorized** and **unnoticed** modification
(cf. integrity in databases)
- ▶ **Authenticity**: Information is **integrity-protected** and **fresh**; clearly associated to the **identity** of a principal



Accountability/Non-repudiability

- ▶ **Accountability**: An action can be reliably associated with the identity of the principal responsible for the action
- ▶ **Non-repudiability**: An action cannot be denied after the fact
Necessary for:
 - Digital contracts
 - Digital interaction with government authorities



Availability

- ▶ **Availability:** protect the system against unauthorized impairment of function
 - vs. Denial of Service (DoS) attacks
- ▶ Availability + Correctness:
dependability: soundness; **reliability** in providing the service



Where are the weaknesses?

- ▶ **Bad Design**
 - E.g., missing security mechanisms, bad security model
- ▶ **Bad Implementation**
 - E.g., buffer overflows, avenues for circumvention
- ▶ **Bad Administration**
 - E.g., leaving accounts with standard password, open ports in firewalls, using inappropriate systems and tools
- ▶ **Bad Management**
 - E.g., leaving the security policy less than well-defined, not investing in training, no funds for security audits, no management support for the organizational cost of security measures



Design principles for secure systems (1)

- ▶ **Principle of Economy of Mechanism**

The protection mechanism should have a simple and small design.

- ▶ **Principle of Fail-safe Defaults**

The protection mechanism should deny access by default, and grant access only when explicit permission exists.

- ▶ **Principle of Complete Mediation**

The protection mechanism should check every access to every object.

[Saltzer/Schroeder 1975]



Design principles for secure systems (2)

- ▶ **Principle of Open Design**

The protection mechanism should not depend on attackers being ignorant of its design to succeed (no **security by obscurity**).

It may however be based on the attacker's ignorance of specific information such as passwords or cipher keys.

- ▶ **Principle of Separation of Privilege**

The protection mechanism should grant access only based on more than one piece of information.



Design principles for secure systems (3)

▶ Principle of Least Privilege

The protection mechanism should force every process to operate with the minimum privileges needed to perform its task.

▶ Principle of Least Common Mechanism

The protection mechanism should be shared as little as possible among users.

▶ Principle of Psychological Acceptability

The protection mechanism should be easy to use (at least as easy as not using it).



Design principles for secure systems (4)

▶ Principle of Defense in Depth

There should be multiple layers of defense before a high-value target is compromised. (No Maginot lines.)

▶ Principle of Securing the Weakest Link

The protection mechanism should not have weak spots that allow circumventing the well-secured parts. (Security often is a chain.)

▶ Principle of Reluctance to Trust

The protection mechanism should not give unwarranted trust to any mechanism or entity. (Healthy skepticism.)

(Beyond the 8 principles listed by Saltzer/Schroeder)



Examples: Layer 1 attacks

- ▶ Ethernet Repeaters, most kinds of communication lines:
 - Eavesdropping (attacking confidentiality)
 - Data Modification, Injection
 - (usually simpler on higher layers)

- ▶ Countermeasure: Quantum cryptography
 - Observation changes phenomena
 - Eavesdropping attack can be reliably detected
 - Low bitrate: mainly useful for transferring keying material



Examples: Layer 2 attacks

- ▶ Ethernet Switches: Poisoning the Switch database
 - E.g., make the switch send traffic to all ports

- ▶ ARP Spoofing
 - Eavesdropping (attacking confidentiality), data modification/suppression
 - Tools: Dsniff, Ettercap

- ▶ Spoofing MAC Adresses
 - E.g., to circumvent WLAN access control
 - Tools: ifconfig ... ether ...



Examples: Layer 3 attacks

- ▶ Router: Poisoning Routing Protocols
 - Traffic is diverted
 - Eavesdropping (attacking confidentiality)
 - Traffic suppression (creating black holes so victim cannot be heard)
- ▶ Spoofing IP addresses
 - E.g., to circumvent NFS access control
 - Injection of data (e.g., for Session Hijacking)
- ▶ Loose Source Routing
 - Packets are returned via reversed source route
 - Circumvents TCP Handshake
 - ➔ Loose Source Route is heavily filtered throughout the Internet



Finding victims: Scanning

- ▶ Reconnaissance: Finding potentially vulnerable Hosts
- ▶ IPv4 address space is densely populated
 - Of $\sim 4.3E9$ IPv4 addresses, $\sim 3.7E9$ can be used as unicast addresses; of these $\sim 2.5E9$ are allocated (66 %)
 - Of these, $\sim 1.7E9$ are routed globally (44 % of the usable, 68 % of the allocated address space)
 - $> 0.3E9$ of these have a web server (netcraft.com), which is nearly 10 %!
- ▶ IPv6 makes scanning much harder
 - $4E33$ addresses are allocated (0.01 % of the currently usable space)
 - Enumerating these at 1 Gbit/s takes $\sim 4E19$ years
 - However, there are other ways to collect IPv6 addresses, e.g.
 - DNS analysis
 - Snooping traffic



Internet Background Radiation

- ▶ Worms such as SQL-Slammer are always active somewhere
- ▶ There is also backscatter from random spoofed source addresses
- ▶ „Background radiation“: ~ 0.1–4 Bytes/s/IP-Address
- ▶ Connecting an unpatched Windows-System to the Internet?
 - Infections within minutes (seconds?)
 - Usually crashes completely after ~ 30 minutes
- ▶ Add the intentionally targeted attacks
- ▶ Corporate networks may not need full Internet connectivity
 - Firewalls → next segment



Examples: Layer 4 attacks

- ▶ RST-Attacks
 - Aborting a TCP connection between victim hosts
 - Can seriously damage Routing System (BGP) → DoS
- ▶ SYN-Flooding
 - Create state
 - Overload prevents the creation of normal connections (DoS)



Examples: Layer 7 attacks

- ▶ DNS Spoofing
 - Poison the Caches of DNS Servers
- ▶ Email Spoofing
- ▶ Web Spoofing, Phishing
- ▶ Attacking programs: Buffer Overflows etc.



Commonalities

- ▶ On-Path attackers can eavesdrop
 - Certain active attacks can divert the path to make the attacker “on-path”
 - Countermeasure: Encryption (Cryptography)
- ▶ Identity assertions (e.g., source addresses) can be faked
 - Countermeasure: Authentication
 - Must be resistant against eavesdropping and replay
 - Cryptographic authentication



The Internet threat model

- ▶ Assumption: The end-systems are not compromised
 - There are ways to minimize damage even in this case, e.g., perfect forward secrecy
- ▶ However, the communications channel is completely compromised, i.e., attacker can:
- ▶ Read any PDU
- ▶ Undetectably remove, modify, inject any PDU
 - Including PDUs that appear to be from a “trusted” machine



Types of attacks

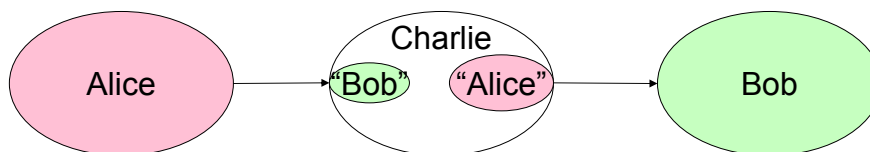
- ▶ Passive attacks:
 - Attacker only reads packets (“sniffing”)
 - Extremely easy on wireless
 - Relatively easy on shared media such as Ethernet
 - Can only really be excluded by quantum cryptography
- ▶ Active attacks:
 - Attacker also injects new packets into the network
 - Source address can be spoofed
 - Egress/ingress filtering can make this harder
 - Blind attacks: can only write, not read
 - Replay attacks: inject copy of previous good packet (“launch rocket now”)

Combinations

- ▶ **Passive followed by active attack:**
 - Password sniffing (passive) + login using sniffed password (active)
 - Can be supported by an offline attack, e.g. dictionary attack
 - If sniffed information can be used offline to determine whether guessed password is correct
- ▶ **Active attack to facilitate passive attack:**
 - Subvert forwarding/routing system to divert traffic via attacker
 - Quite easy at layer 2 (tools: dsniff, ettercap)
 - Subverting routing at layer 3 may be harder
 - Compromised router/switch can be used as tool

Man-in-the-middle (middleperson) attack

- ▶ **Special form of active attack:**
- ▶ **Man-in-the-middle creates the illusion for each communicating partner to be the other communicating partner:**
 - Messages can be copied and modified



- ▶ **Countermeasure: Cryptography (Authentication/Encryption)**



On-path vs. off-path attacks

- ▶ On-path attacker can easily eavesdrop, spoof, suppress, inject
- ▶ Off-path attacker typically is limited to blind attacks
 - Unless topology can be subverted to convert off-path into on-path situation
- ▶ Many protocols protect well against off-path attackers, not so well against on-path
 - E.g., TCP random sequence numbers are worthless if overheard by on-path attackers
- ▶ (Note that real Internet paths are often asymmetric.)



Special case: link-local attacks

- ▶ Link-local peers may enjoy special trust (e.g., home network)
- ▶ Packets with TTL 1 will only reach link-local peers
- ▶ Packets with TTL 255 can only have been originated by link-local peers
- ▶ Warning: Some tunneling systems don't decrement TTL



Key Management

- ▶ Keys “wear off”
 - Each usage increases amount of material available for cryptanalysis
 - The longer (in time) a key is in use, the more time an attacker has for cryptanalysis
 - Some modes of operation only allow limited number of uses before IV repeats
- ▶ Rekeying
 - After some time / some amount of data exchanged, rerun key management
 - **Key derivation:** Use “master key” to derive the actual keys in use
 - Needs cryptographically secure derivation function
 - Per-application keys: compromise in one application does not affect other application



Case Study: IEEE 802.11 WEP

- ▶ “Wired Equivalent Privacy”: Encryption designed under serious fear of export control problems
- ▶ Key too short (40 bits, this one remedied in products)
- ▶ Bad crypto usage (24-bit IV, RC4 problems)
 - Product flaws often made IV reuse even more likely
- ▶ No replay protection
- ▶ Ridiculous integrity check (CRC32 allows bit flipping attacks)
- ▶ The really bad problem:
 - There is only one key for each WLAN
 - The long-term key is directly used as encryption key
 - Once cracked, there is no security left





Case Study: IEEE 802.11i (“WPA”)

- ▶ 802.11i: Completely redesigned security algorithms
- ▶ Pairwise master key (PMK)
 - Derived from secure authentication protocol (e.g., EAP-TLS, EAP-TTLS)
 - PMK is not used directly for encryption/authentication of data
- ▶ PMK can alternatively be per-WLAN shared secret (“Pre-shared key”, WAP-PSK)
 - Intended for SOHO use (no EAP authentication server available)
 - Well-defined Password-based Key Derivation Function (PBKDF2, RFC2898) to convert passphrase into fixed-size key (**usability!**)
 - Unfortunately, still vulnerable to passive offline dictionary attack
 - But passphrase can be long and hard to guess, thwarting dictionary attacks
 - I.e., need to choose passphrase wisely

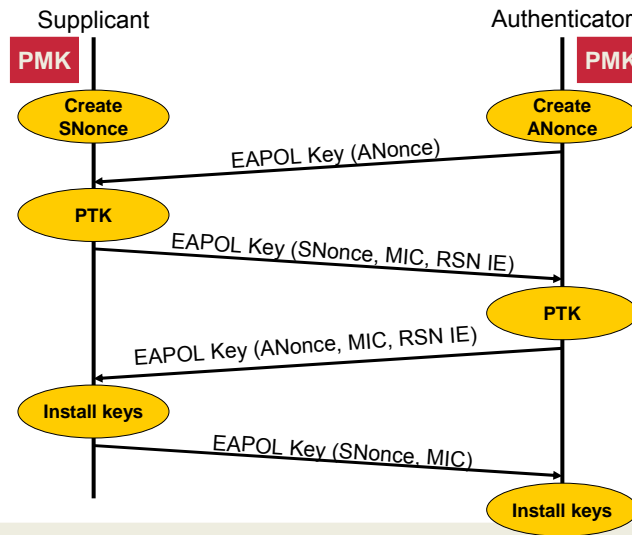


WPA: 4-Way-Handshake and PTK

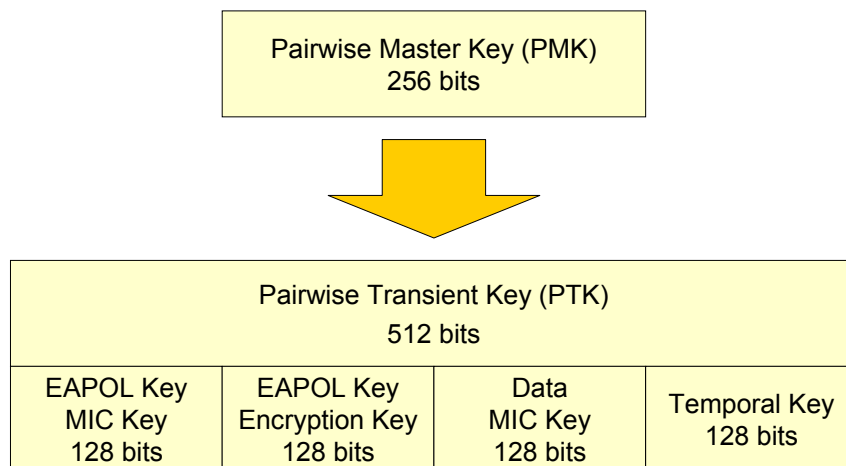
- ▶ Do not use PMK for actual data transfer
- ▶ Instead:
create Pairwise Transient Key PTK (512 bits) from the PMK and two Nonces
 - ANonce (*authenticator nonce*) and SNonce (*supplicant nonce*) ensure freshness of PTK
 - Principle: **Joint Key Control** (both parties contribute to key)
- ▶ This is then divided up into 4 parts of 128 bit each:
 - Encryption key, Integrity protection key
 - EAPOL-Key Encryption, EAPOL-Key Integrity
- ▶ I.e., a part of the PTK is used for protecting rekeying

- ▶ The four-way handshake also establishes that both Station and AP know PMK
 - Principle: **Mutual Authentication**

4 Way Handshake und PTK



4 Way Handshake und PTK





Group Keys

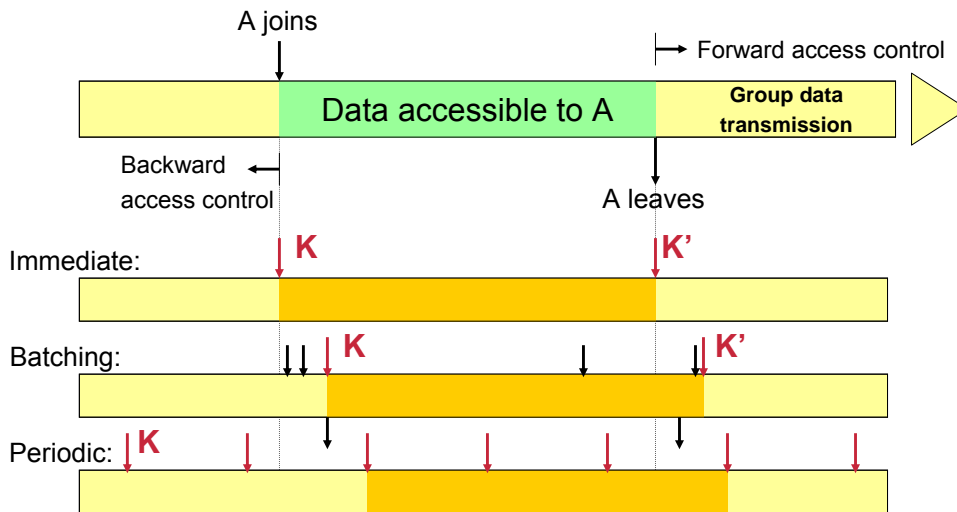
- ▶ So far, all keys are pairwise (except PSK)
- ▶ Problem: Broadcasts (AP to Station) cannot use pairwise key
 - (Exception: Broadcast packets from the Stations are unicast to APs first)
 - For unicast Station→AP, the normal PTK is used
- ▶ Separate Group Transient Key (GTK)
- ▶ Sent from AP to each Station
 - via pairwise security association, once this has been established
- ▶ Needs to be recreated after each disassociation!
 - The old WEP Key-ID field is used to indicate a key serial
 - Allows seamless transition from old to new GTK



Generalizing the Terminology: Multicast Data Confidentiality

- ▶ GTK == use a shared session key in the group:
Traffic Encryption Key (TEK)
- ▶ To be deployed with a symmetric encryption algorithm
- ▶ Straightforward
- ▶ In addition:
 - Initial key distribution
 - Rekeying due to membership changes
- ▶ PTK == one or more **Key Encryption Keys (KEK)**

Data Confidentiality and Re-Keying



Group Authentication

- ▶ Apply shared group key also to authentication
- ▶ Calculate hashed message authentication code (HMAC)
 - Hash over the message + key + nonce (e.g. timestamp)
 - E.g. Message Digest 5 (MD5, RFC 1321), better: SHA1 (RFC 3174), SHA256/384/512 (RFC 4634)
- ▶ Allows to identify the originator of a message as one of the group
 - But: does not provide source authentication
 - And does not support integrity protection
 - Message may have been altered by another group member
- ▶ Different for point-to-point communications
 - There are only two peers sharing a secret



Source Authentication (1)

- ▶ Prove the origination of a message / packet
- ▶ Must work for multicasting
- ▶ Digital signatures?
 - Public-key cryptography too expensive
 - Would require PKI
- ▶ Possibly operate on blocks of packets
 - Hash over a group of packets, then sign
 - Application-specific authentication support
 - E.g. file transfer: Calculate signatures only once over the entire contents
 - Entire transmission is lost if only a single packet is faked
 - Delays verification of contents!



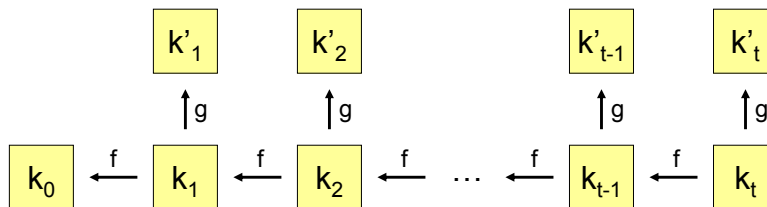
Source Authentication (2)

- ▶ Authenticating individual packets
 - Tree hashing / hash chaining
 - Hash a sequence of packets
 - e.g. Packet P1 validates the hash of P2, P2 that of P3, etc.
 - Only one packet (e.g. P1) is signed per run of packets
- ▶ Issues with packet losses: verification may get impossible
 - Multi-chaining: include a hash in several other packets
 - Still may lead to extra packet drops of unauthenticated packets
- ▶ MAC-based authentication of unreliable streams: TESLA
Timed Efficient Stream Loss-tolerant Authentication

TESLA (1)

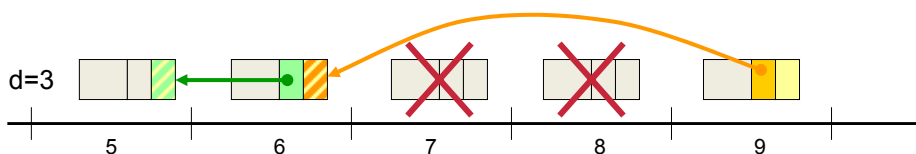
Basic idea: Hash key chain

- Select an initial key
- Then calculate derived keys using a one-way function f
- Generate keys k_0, \dots, k_t – starting with k_t as initial random key
 $k_{t-1} = f(k_t)$
- Use another hash function to derive k'_j from k_j : $k'_j = g(k_j)$
- Use keys in backwards order, starting with k_0



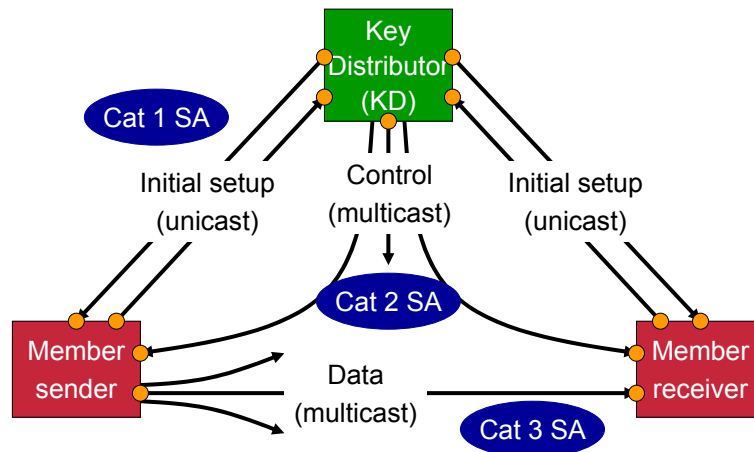
TESLA (2)

- ▶ Requirement: rough time synchronization of senders & receivers
- ▶ Subdivide time axis into t intervals
 - All data packets per interval $i = [1, \dots, t]$ are authenticated with k'_i
 - Choose a disclosure interval d (equals authentication / processing delay)
- ▶ Sender transmits a digitally signed packet to initialize
 - Include “commitment to key chain” by means of signed k_0
- ▶ Sender transmits data packet P_j in interval i containing
 - Data D_j , the revealed key k_{j-d} of interval $j-d$, auth MAC using k'_j





Group Security Association (GSA)



Group Management

- ▶ Initial setup of a Category 1 SA to the KD
 - (Several KDs may operate in a distributed fashion)
 - Point for access control policy enforcement
 - Authenticate the new group member
 - Verify its authorization to participate in the group
 - Configure member
 - Bootstrap Category 2 SA
 - Initialize Category 3 SA(s)

- ▶ Group management involves rekeying
 - Via push mechanisms using Category 2 SA
 - Via pull mechanisms through Category 1 SA



Group Key Management

- ▶ Provide a shared group key to all members: TEK
- ▶ Update group key during the group's lifetime
 - Periodically to “defeat” cryptoanalysis
 - For membership changes
- ▶ Group key management architectures
 - E.g. IKAM
 - Hierarchical approach to key management and distribution
- ▶ Group key distribution protocols
 - GKMP, GSAKMP (derived from ISAKMP), GDOI
 - MIKEY (Multimedia Internet Keying; used for RTP)



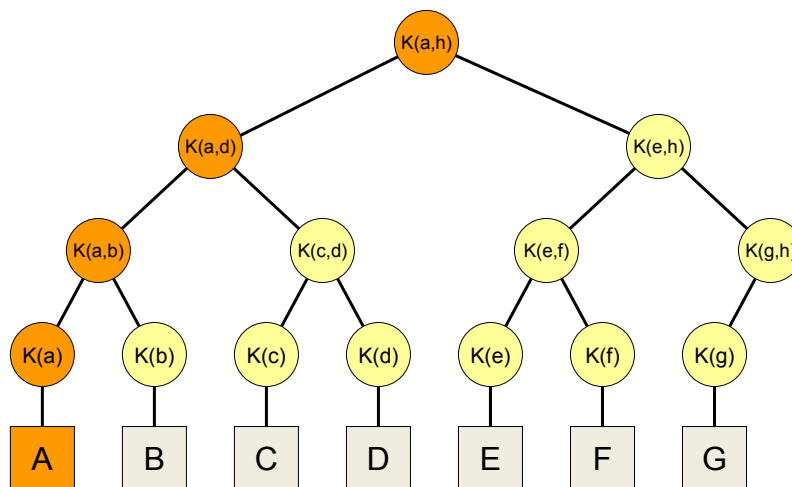
Group Key Management Algorithms

- ▶ Initialization and re-keying
- ▶ Re-keying: immediate, periodic, batching
- ▶ Simplest variant for group changes
 - Re-key each group member individually using Cat 1 SA
 - $O(n)$ for rekeying
 - Does not really scale to large groups
- ▶ Periodic re-keying: use a different group key from Cat 2 SA
 - Helps for stable membership
- ▶ Use hierarchical schemes to achieve better scalability

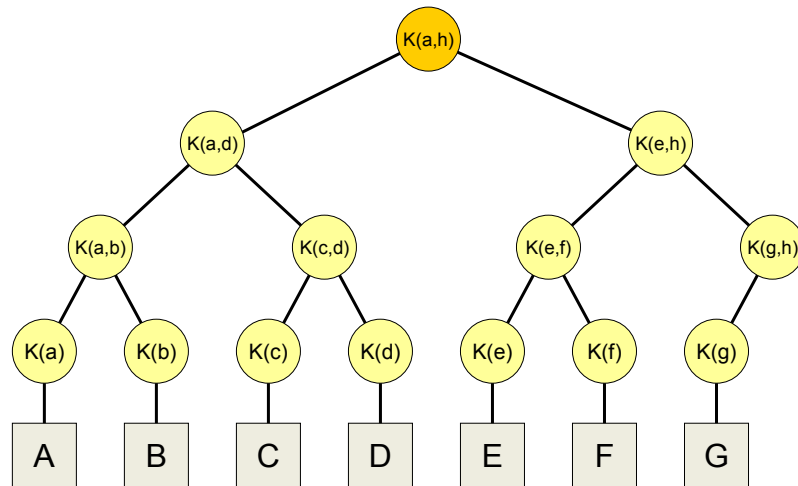
Example: Logical Key Trees (LKH)

- ▶ Create a (balanced) binary tree
 - As many leafs as group members (each leaf represents a member)
 - Adjusted dynamically by adding nodes (possibly splitting existing ones) and removing nodes
- ▶ Each node (including leafs) represents a KEK
- ▶ KEKs are used to distribute TEKs and new KEK when membership changes
- ▶ A group member A knows all the keys (KEKs) on the path from its corresponding leaf node up to the root
- ▶ Rekeying is done by distributing new keys (TEKs, KEKs) using the KEKs that are known to as many members possible
- ▶ Complexity $O(2 \log n)$ for join and leave group operations

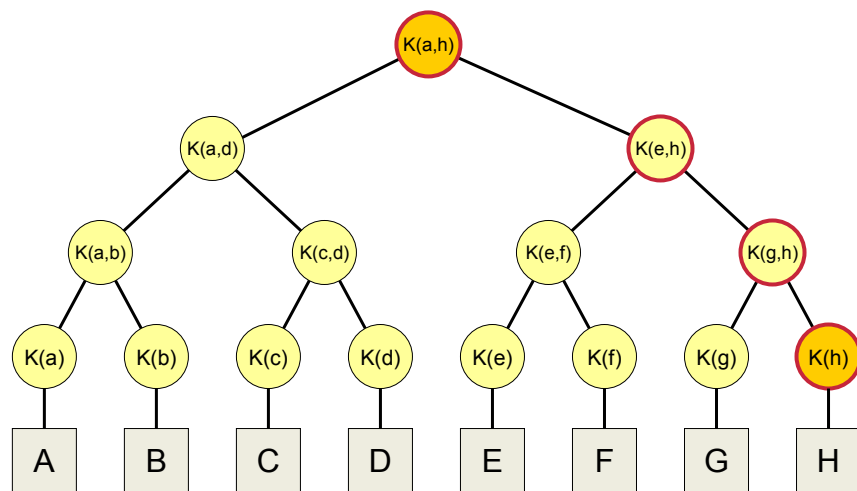
LKH Example



LKH Example: Periodic Re-keying

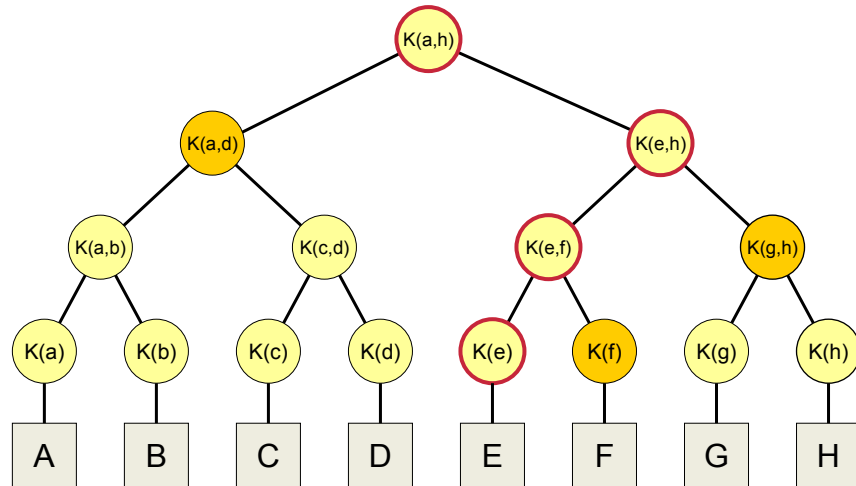


LKH Example: H joining





LKH Example: E leaving



Multicast Security Review

- ▶ No surprise: Adding Multicast makes life harder
 - Multicast Key Management = Security + Multicast
 - In practice, needs to interact with membership management
- ▶ LKH: Adding (even artificial) structure to a group can reduce effort required for state management algorithms significantly
- ▶ Scalable, efficient source authentication is really hard
 - TESLA is a nice “out of the box” idea with a limited field of application



Security: Take-away message

- ▶ Study security best practices
 - Key management usually is the complex part
 - Most security algorithms have a **limited field of applicability**
 - Often, security mechanisms need to be combined to hold water
 - But, in combinations, one algorithm can be used to attack another in surprising ways
- ▶ Reuse existing protocols, frameworks, algorithms as much as possible
 - But make sure you are using them **within** their field of applicability!
 - Communication security vs. object security
- ▶ Most important:
Submit security protocols to early review (open design!)