Proactive Key Distribution to support fast and secure roaming

Arunesh Mishra, Minho Shin, William Arbaugh

University of Maryland

College Park

Insun Lee, Kyunghun Jang

Samsung Electronics

Goals of this Talk

- Introduce the different methods for key predistribution (the good and the bad).
- Introduce a back-end protocol that is independent of key management and hand-shakes.
- Information slides on example implementations for key distribution (not covered in talk)

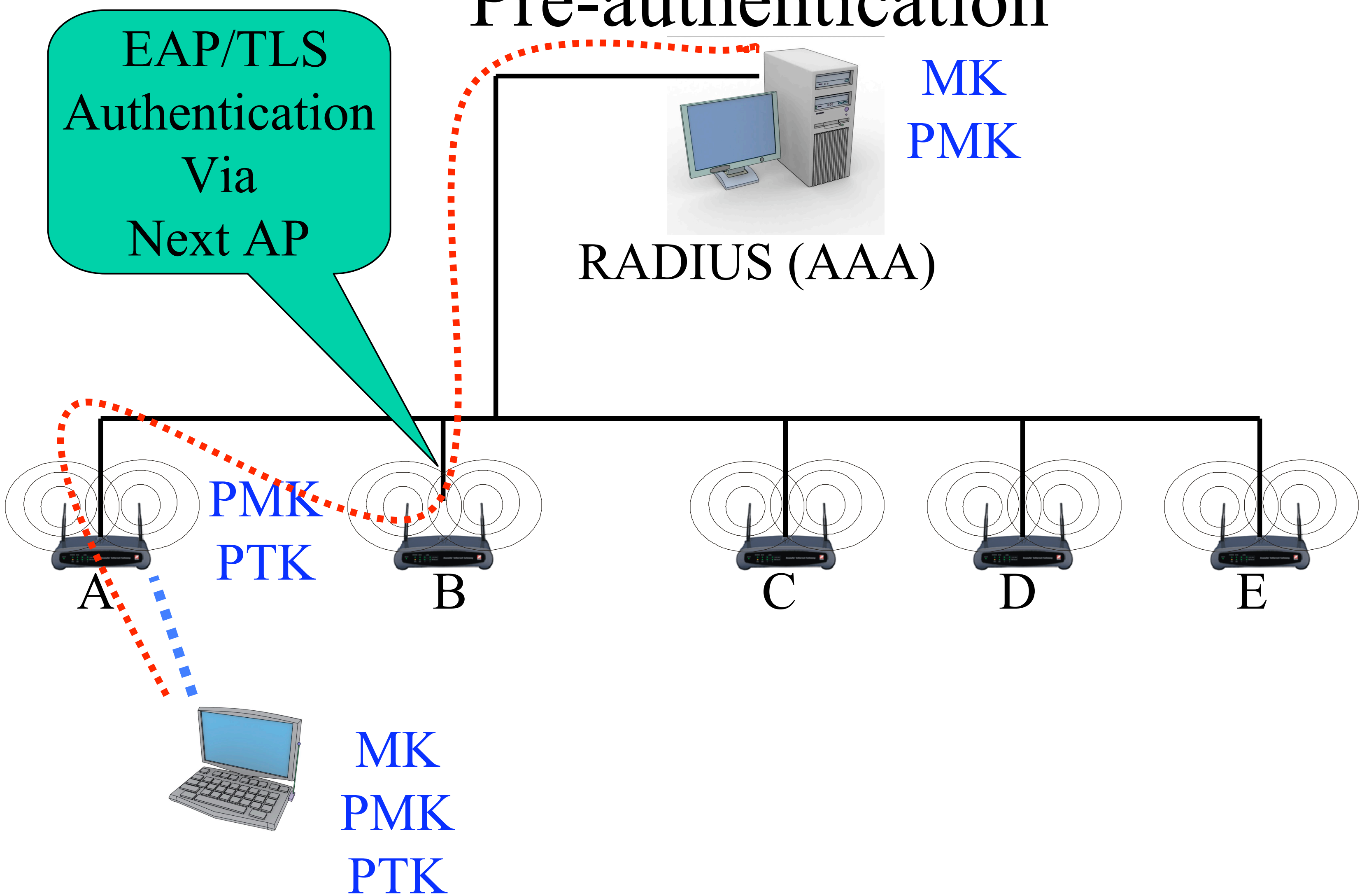
TGi MUST Support Fast Roaming

- Otherwise non-standard and non-vetted solutions will evolve….creating potential “brand” problems.
- Transparent roaming was one cause of exponential growth in the cellular market.
- Interworking is around the corner.

Backend Requirement

- Protocol MUST be standardized within IETF
  - This requires that key material NOT leave the AS.
  - This means that the protocol should fit within current and future AAA practice.

Pre-authentication



Problems with Pre-Auth

- Expensive in terms of computational power for client, and time (Full EAP-TLS can take seconds depending on load at RADIUS Server). TLS-Resume will make things faster, but other problems persist.
- Requires well designed and overlapping coverage areas
- Can not extend beyond LAN
- No opportunity for Interworking

Goals

- Permit fast roaming without reducing overall security
- Fast roaming occurs when the total cost of Layer 1-3 hand-off times is less than 50ms (Ideally 35ms).

TGi Fast Roaming Goals

- Handoff to next AP SHOULD NOT require a complete EAP/TLS re-authentication.
- Compromise of one AP MUST NOT compromise past or future key material, i.e. *perfect forward secrecy, and with stand known key attacks*.

TGi Trust Assumptions

- AAA Server is trusted
- AP to which STA is associated is trusted. All other AP's are untrusted.

Only Three Ways to meet TGi Goals

- Exponentiation support for asymmetric cryptographic operations at AP, or
- Trusted Third Party, i.e. Authentication/Roaming Server
- Use IAPP with proactive caching

Three methods for key distribution

- Static roam keys
  - Does not provide PFS
  - Simple implementation for back-end although storage requirements at the AP's can be large! One key for every STA in the LAN, but can be combined with proactive key distribution.
- IAPP with proactive caching
  - Communications from the next association is compromised if STA associates to a compromised AP.
- Proactive key distribution
  - Provides PFS and protection from known key attacks
  - Slightly more complex.

Static Roam keys sketch

- AS pushes a unique seed for key derivation, e.g. PMK, to each AP which the STA knows how to derive without further communications. The PTK is then derived via some form of a hand-shake.
  - Past communications are subject to compromise if the AP becomes compromised.
  - Large memory requirement for AP unless combined with a proactive distribution means.

IAPP

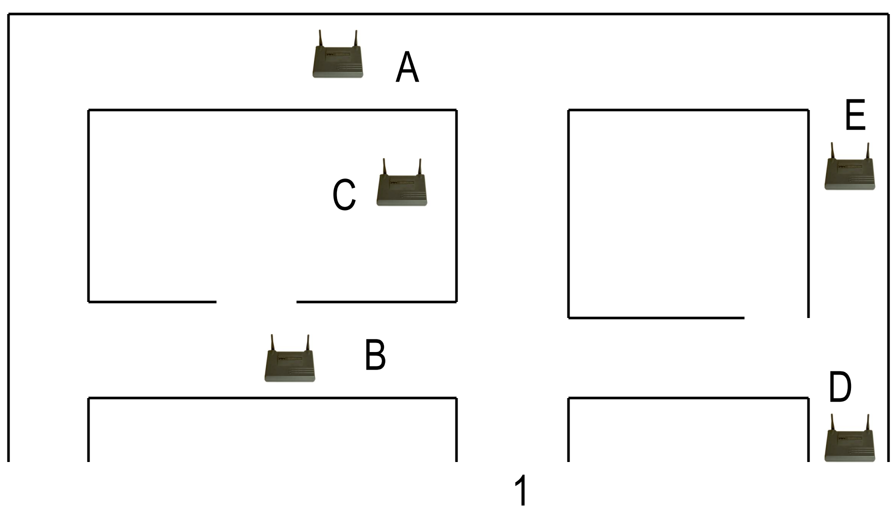
- The AP derives the next PMK (for each neighbor AP) via a hash chain and a roam key(RK) provided by AS such as:
  - $PMK_{next} = PRF(RK, PMK_{current}, STA_{mac}, \text{next } AP_{mac})$
  - PMK_{next} is sent to next AP via IAPP caching (TGf)
  - PTK is derived via some handshake
- Compromised AP only compromises current and next PTK.

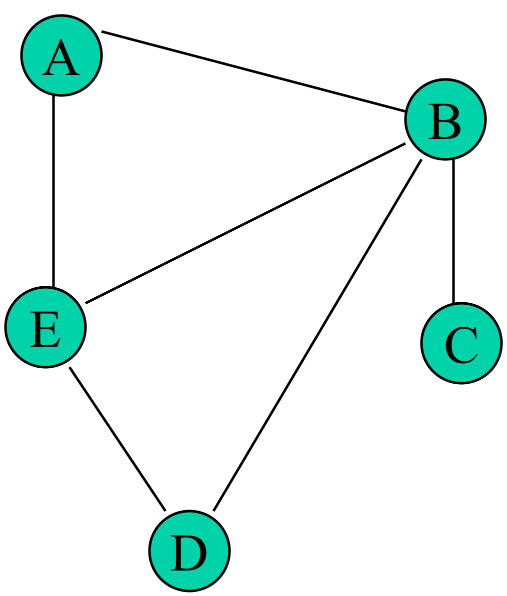
Proactive Key Distribution (TGi)

- Extend Neighbor Graphs and Proactive Caching (IEEE 11-02-758r1.ppt) to support key distribution by the AS
- Eliminates problems with sharing key material amongst multiple APs
  - Easily extended to support WAN roaming
  - Extendable to support Interworking

Neighbor Definition and Graph

- Two APs *i* and *j* are neighbors iff
  - There exists a path of motion between *i* and *j* such that it is possible for a mobile STA to perform a *reassociation*
  - Captures the '*potential next AP*' relationship
  - Distributed data-structure i.e. each AP or AS/RS can maintain a dynamic list of neighbors





AP Neighborhood Graph – Automated Learning

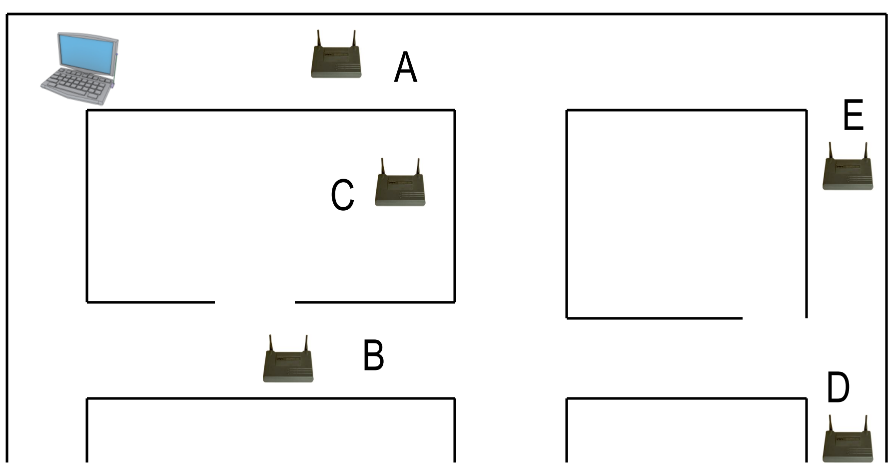
- Construction
  - Manual configuration for each AP/RS or,
  - AP/RS can learn:
    - If STA *c* sends *Reassociate Request* to AP *i*, with old-ap = AP *j* :
    - Create new neighbors *(i,j)* (i.e. an entry in AP *i*, for *j* and vice versa)
    - Learning costs only one '*high latency handoff*' per edge in the graph.
    - Enables mobility of APs, can be extended to wireless networks with an ad-hoc backbone infrastructure.
    - Dynamic implementation using LRU replacement permits invalid and stale entries to time out.

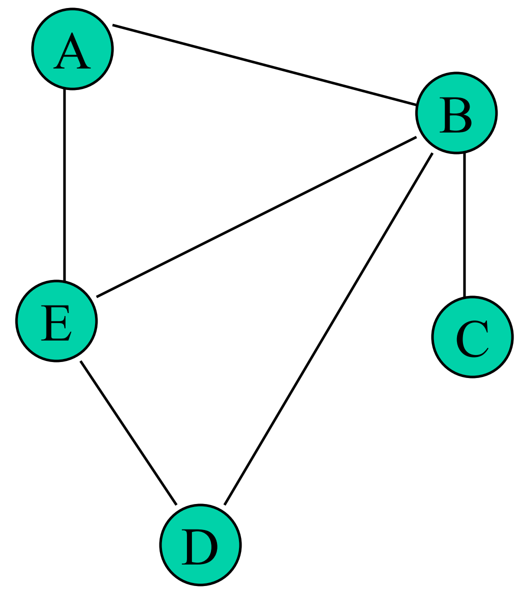
Graph Synchronization

- The graph's state at the accounting server is updated by:
  - Accounting-Request messages from the current AP (draft-congdon-radius-8021x)

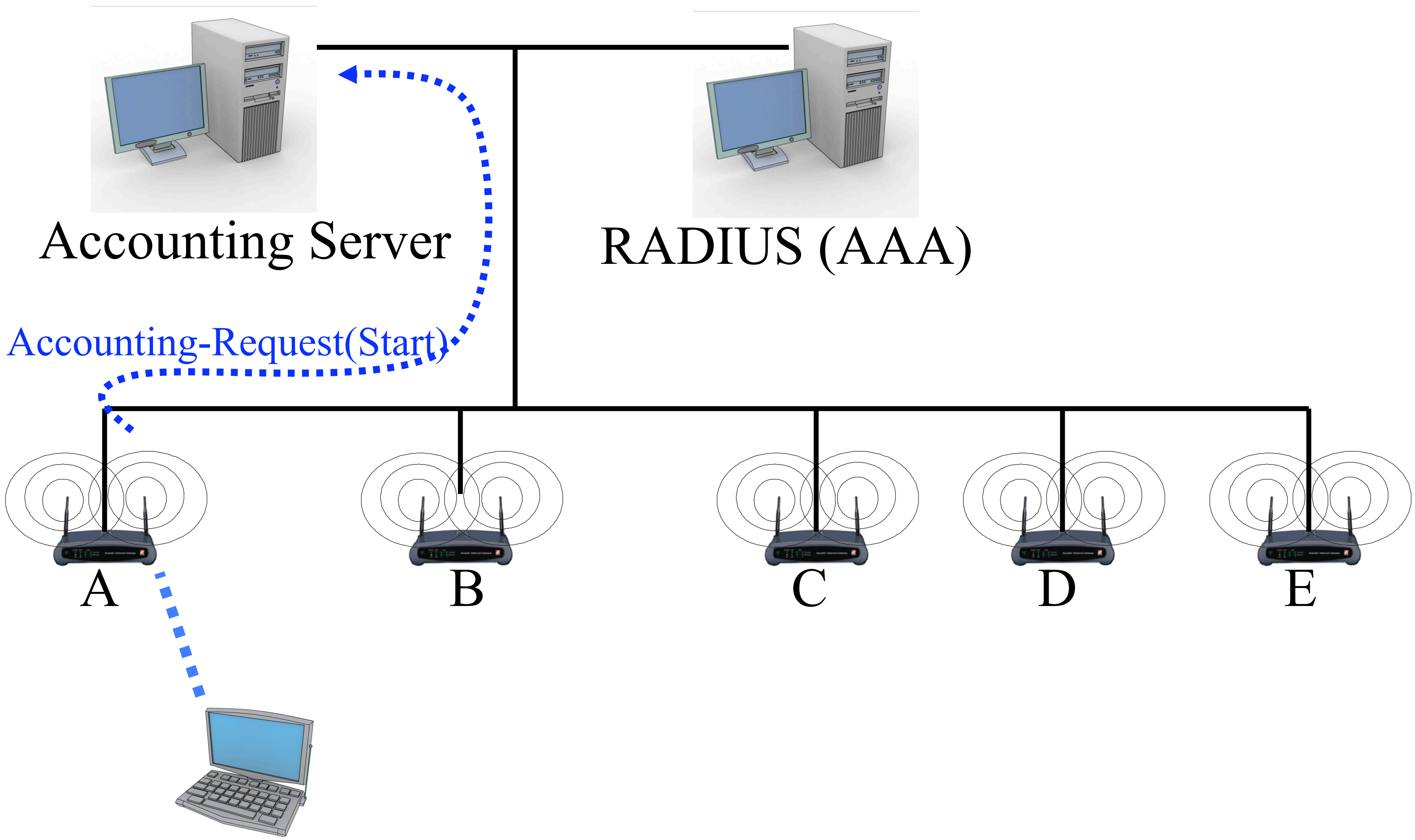
Roaming Example

- Given the following infrastructure with associated neighbor graph with STA about to associate to AP A.

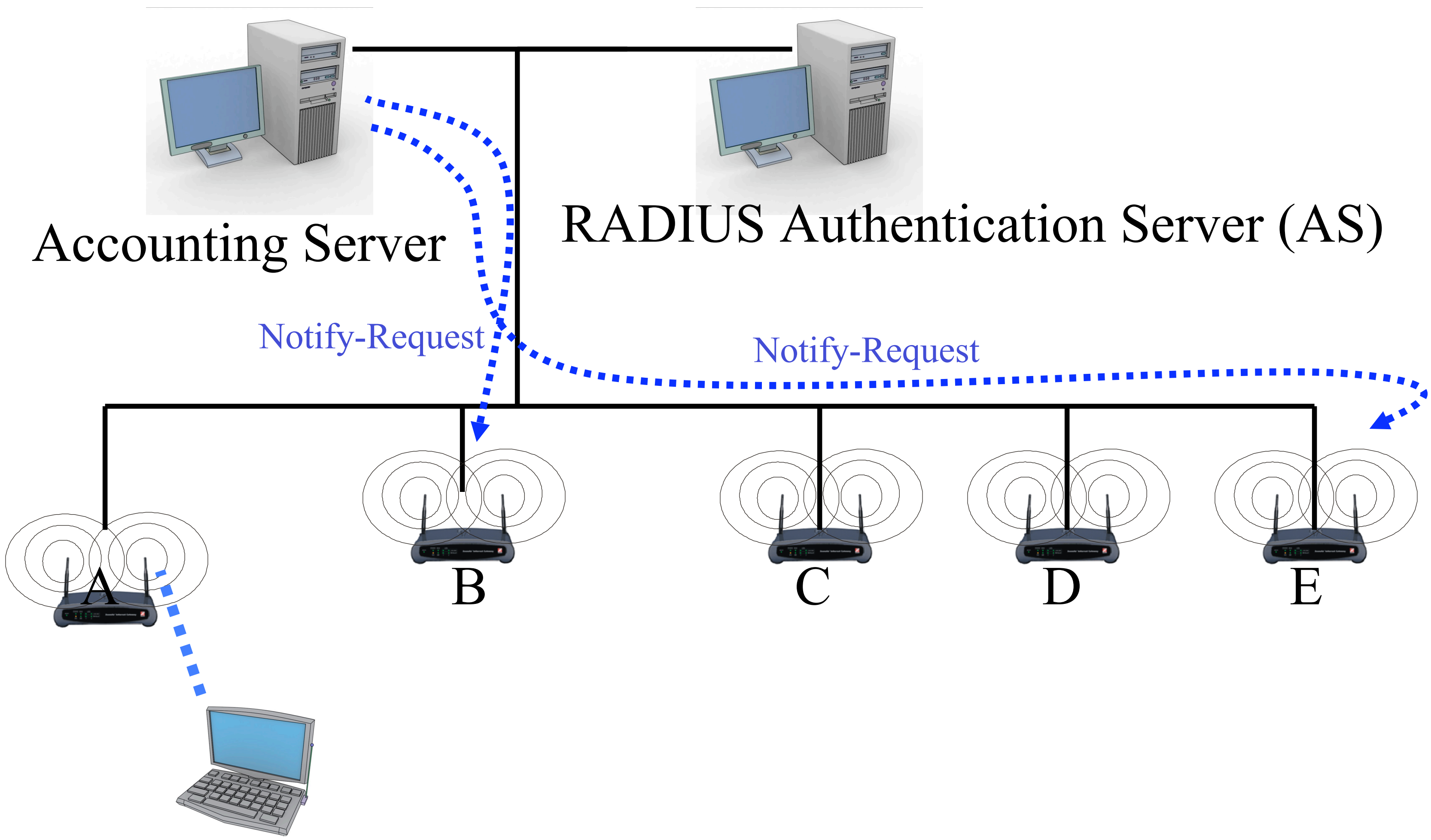




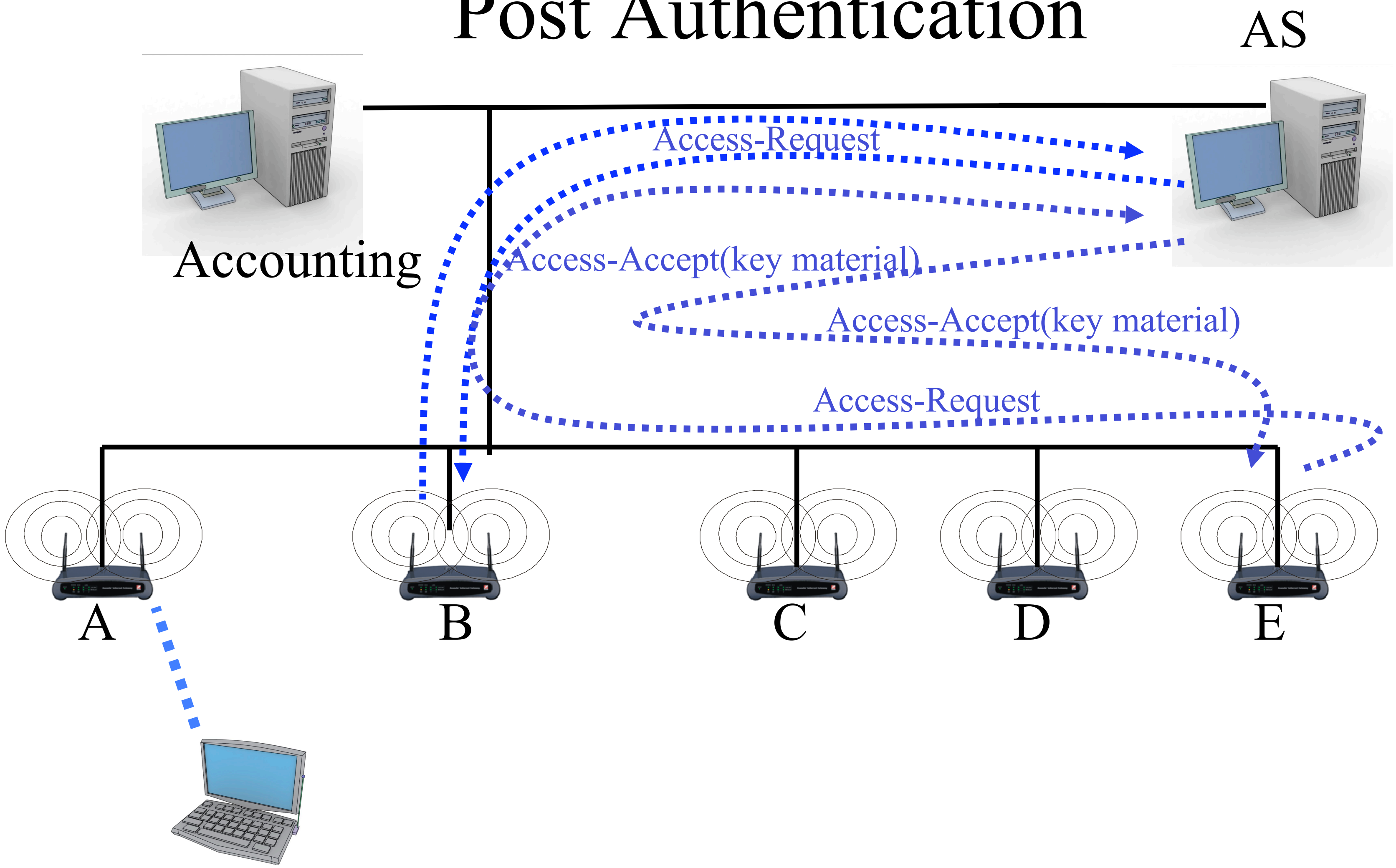
Post Authentication and 4-handshake



Proactive Key Distribution



Proactive Key Distribution Post Authentication



AP Actions on Notify Request

- Dynamic Keys, i.e. PMK changes per roam.
  - AP MUST send an ACCESS-REQUEST to AS
- Static Key, i.e. PMK is unique per AP but never changes.
  - Nothing unless authorization is required.

Maximum STA Velocity

For the Notify and PMK install to occur in time, we need:

2 RTT + handshake < D/v

Where:

D = coverage diameter

v = STA velocity

RTT = round-trip time from AP to AAA server, including processing.

Assuming D=100 ft, handshake = 10 ms, and RTT = 100ms, we get:

v = 100 ft/ (200ms + 10 ms) ~ 500 ft/sec = **Mach 0.5!!**

Conclusions

- Provided an overview of various options
- Provided a protocol that:
  - Can support high speed roaming, meets IETF requirements (draft-arbaugh-radius-handoff-00.txt),
  - Is independent of the type of key management/derivation used (static or dynamic), i.e. can IEEE11-03-008r0-I or the method in the information slides of this presentation.
  - Is independent of the type of hand-shake used.

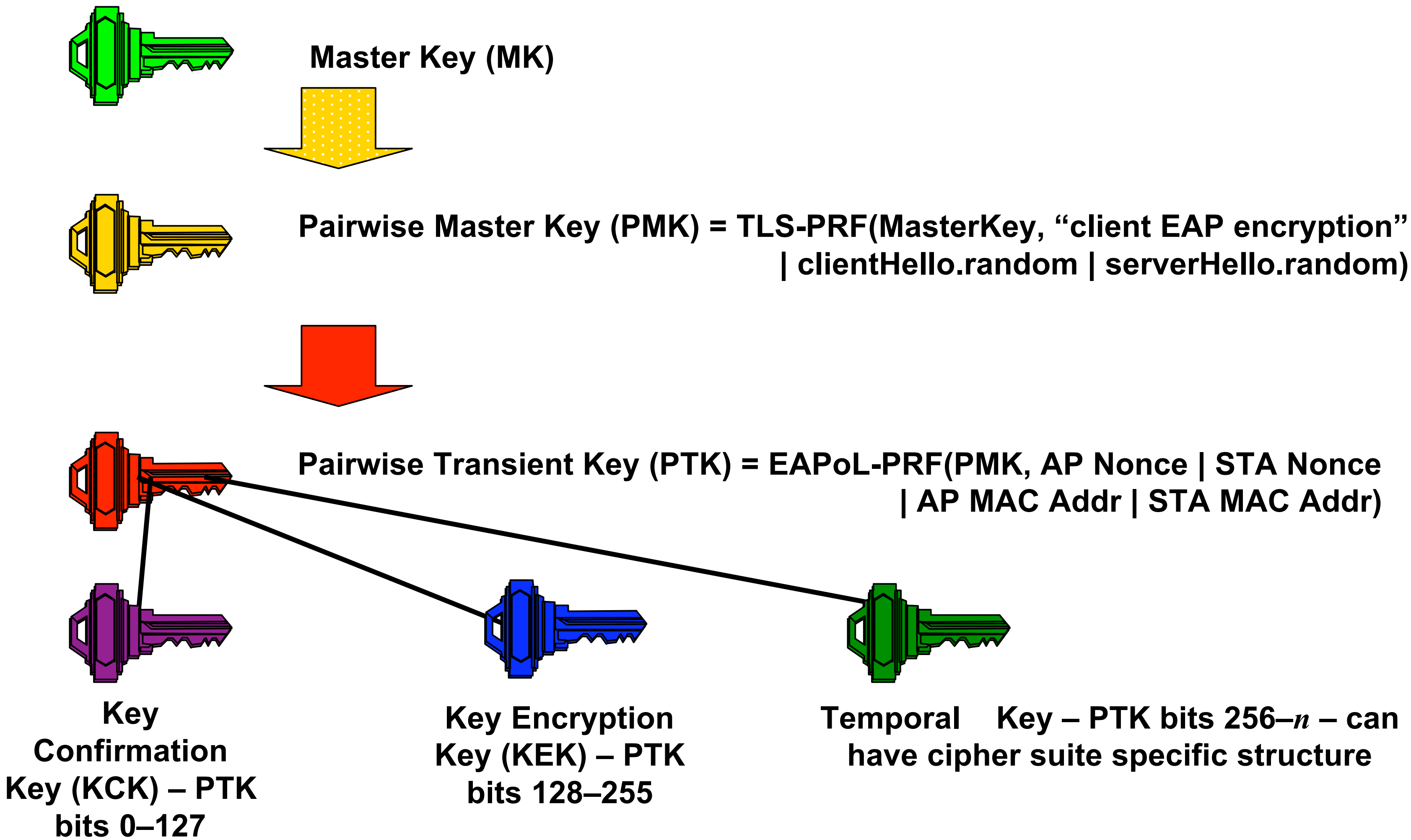
Acknowledgements

- Bernard Aboba assisted with the best way to integrate Proactive key distribution with RADIUS.
- The maximum STA velocity calculation is from Bernard Aboba.
- Jesse Walker pointed out potential synchronization problems in an earlier version of proactive key distribution.
- Nancy Cam-Winget raised several concerns which caused the creation of the IAPP distribution method.

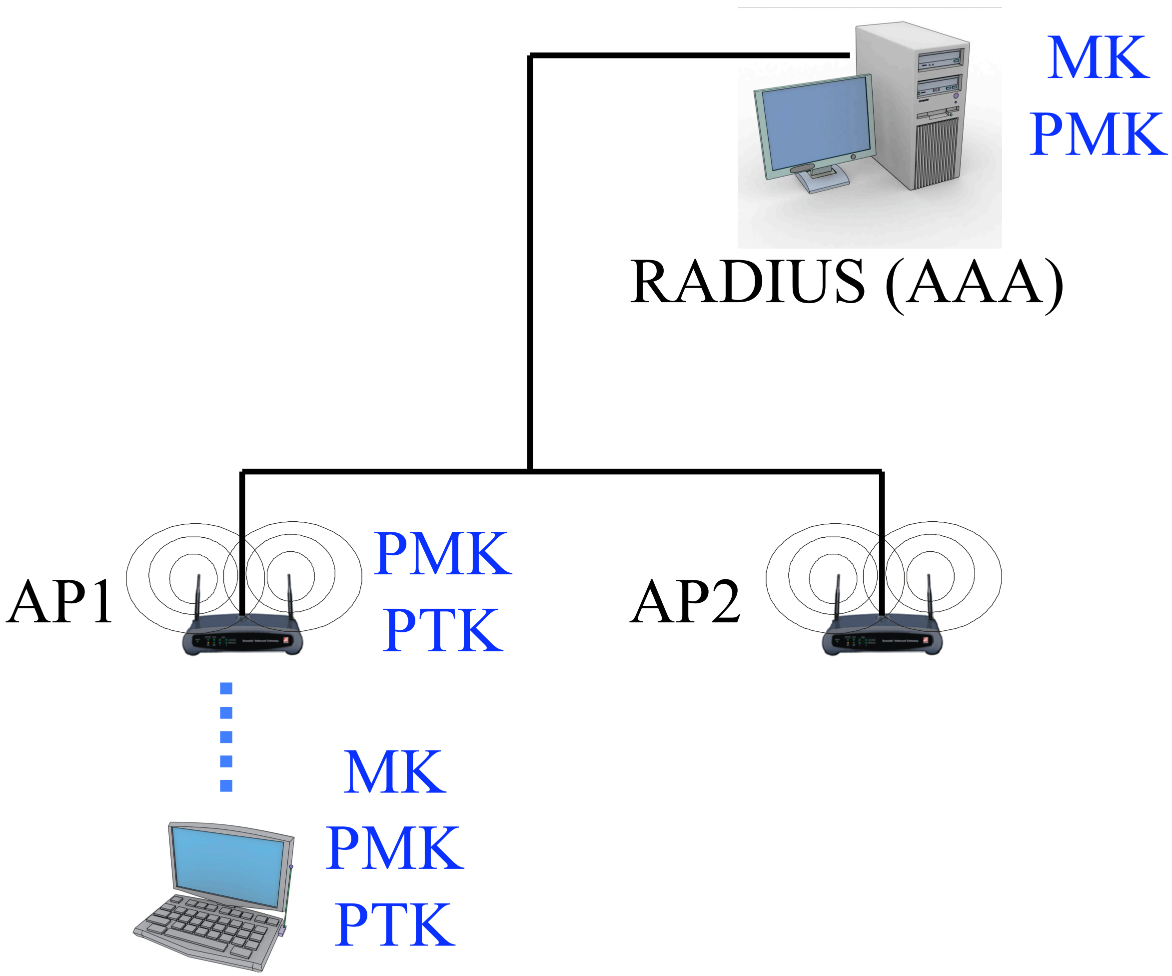
Informational Slides

- How to do predistribution of keys via IAPP and with an accounting server.

TGi Pairwise Key Hierarchy Review



Key Locations



IAPP

- The AP derives the next PMK (for each neighbor AP) via a hash chain such as:
  - $PMK_{next} = PRF(RK, PMK_{current}, STA_{mac}, \text{next } AP_{mac})$
  - PMK_{next} is sent to next AP via IAPP caching (TGf)
  - PTK is derived via some handshake
- Compromised AP only compromises current and next PTK.

IAPP Pairwise Key Hierarchy Review

MK

↓

PMK_{curr}

↙ ↘

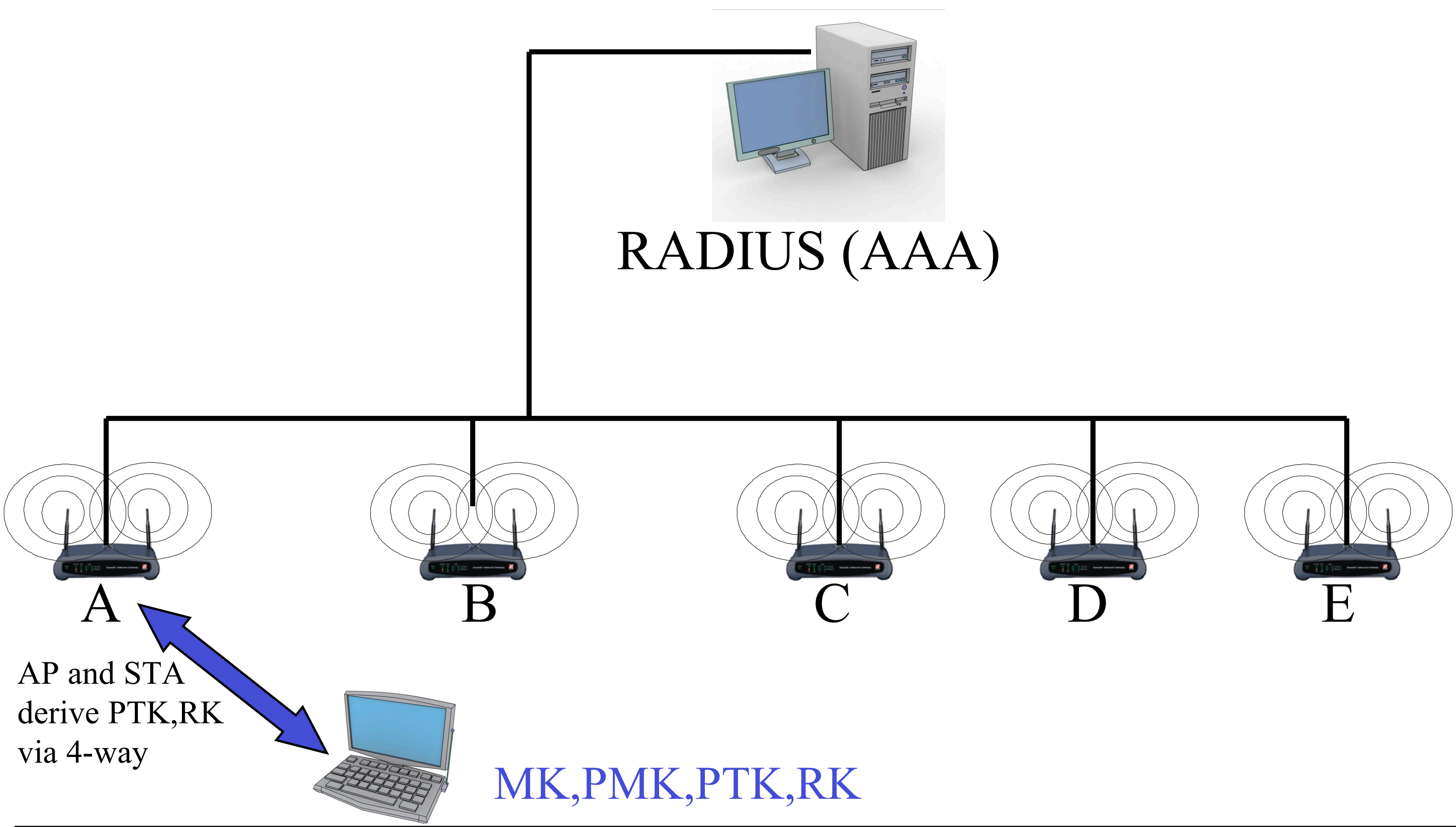
PTK

RK=PRF(PMK, “Roam Key”, AP_{nonce},STA_{nonce})

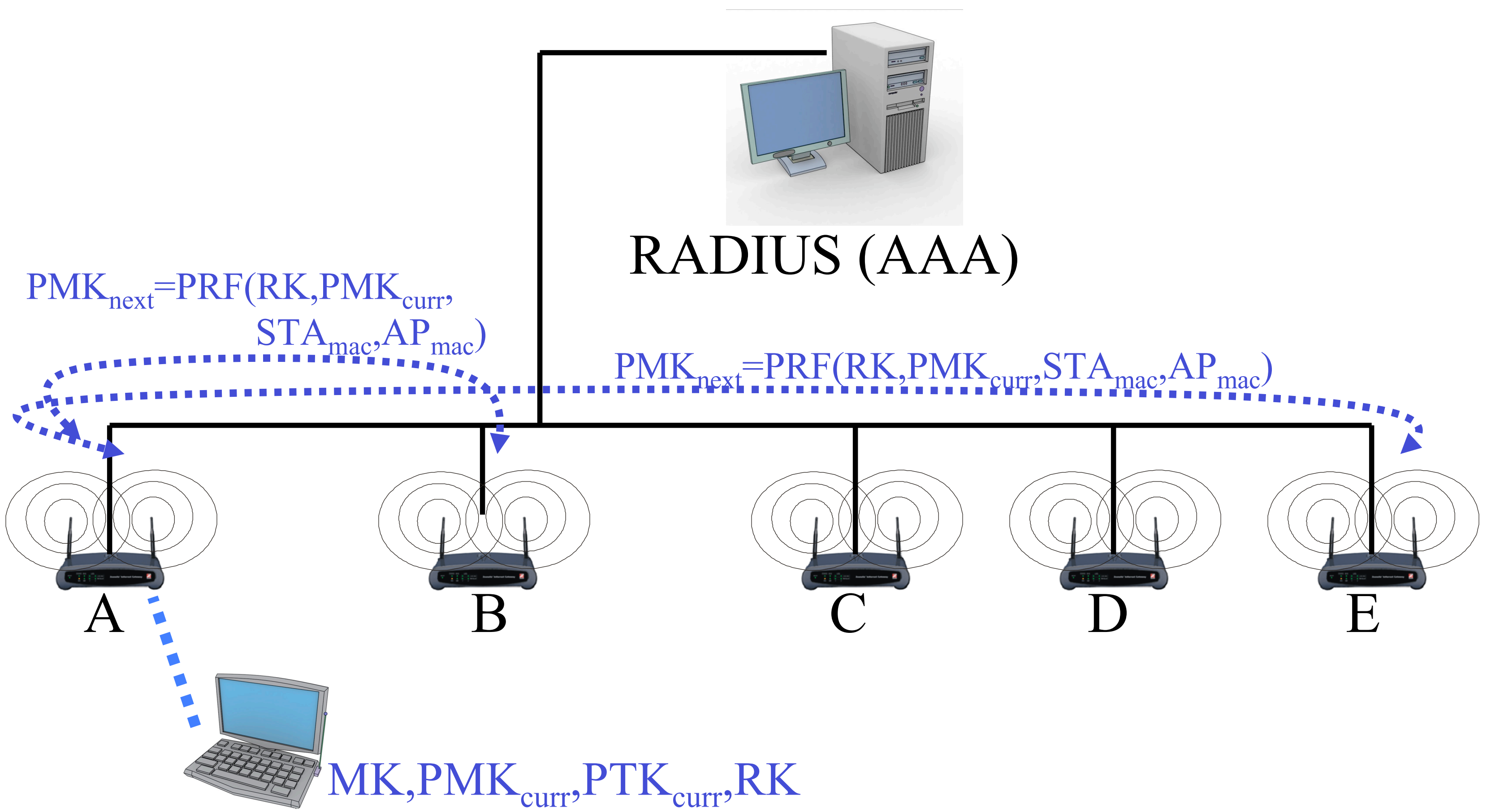
↓

PMK_{next}=PRF(RK,PMK_{curr},STA_{mac},AP_{mac})

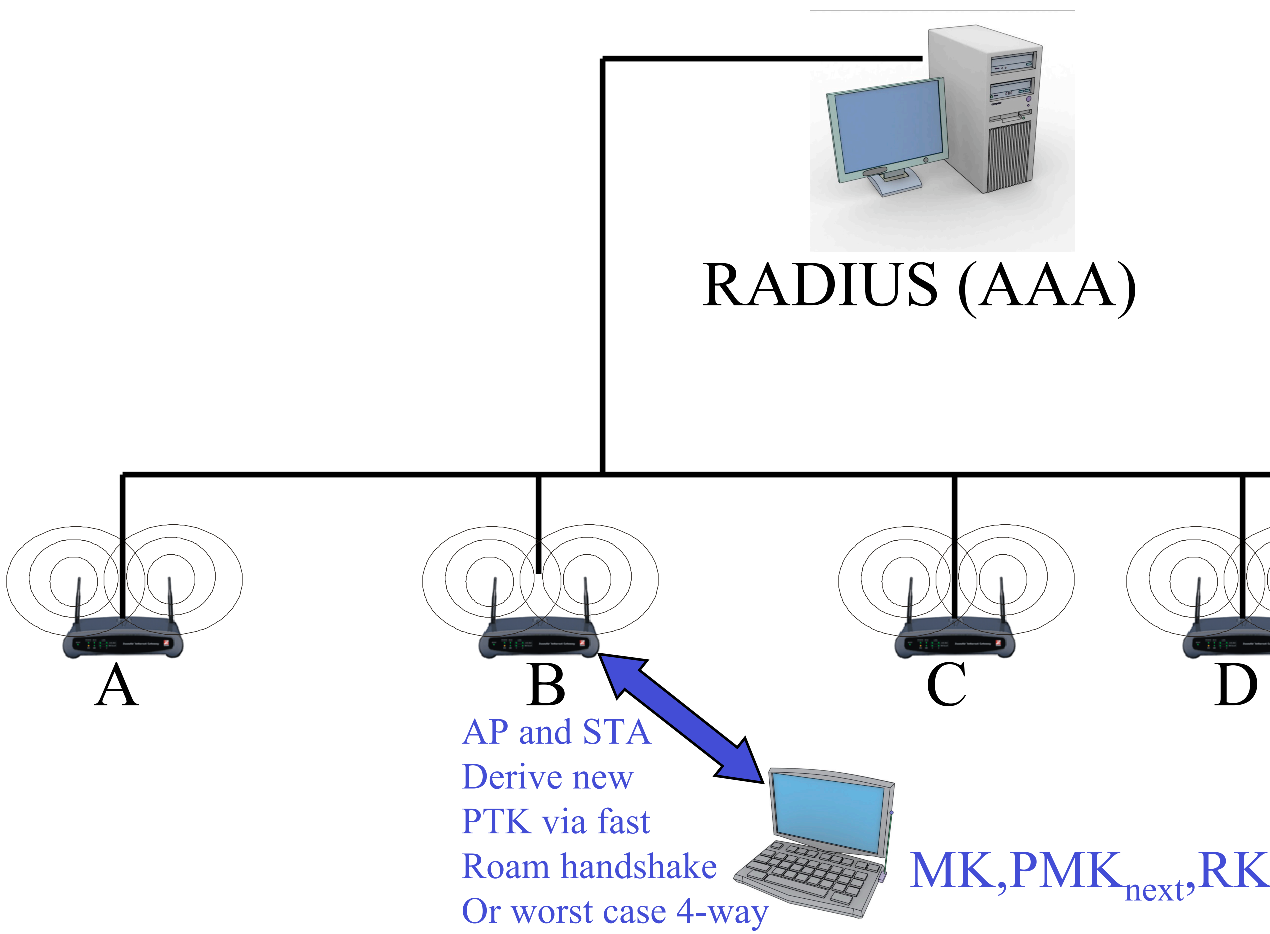
IAPP Example



IAPP Caching of Next PMK to Neighbors



Reassociation



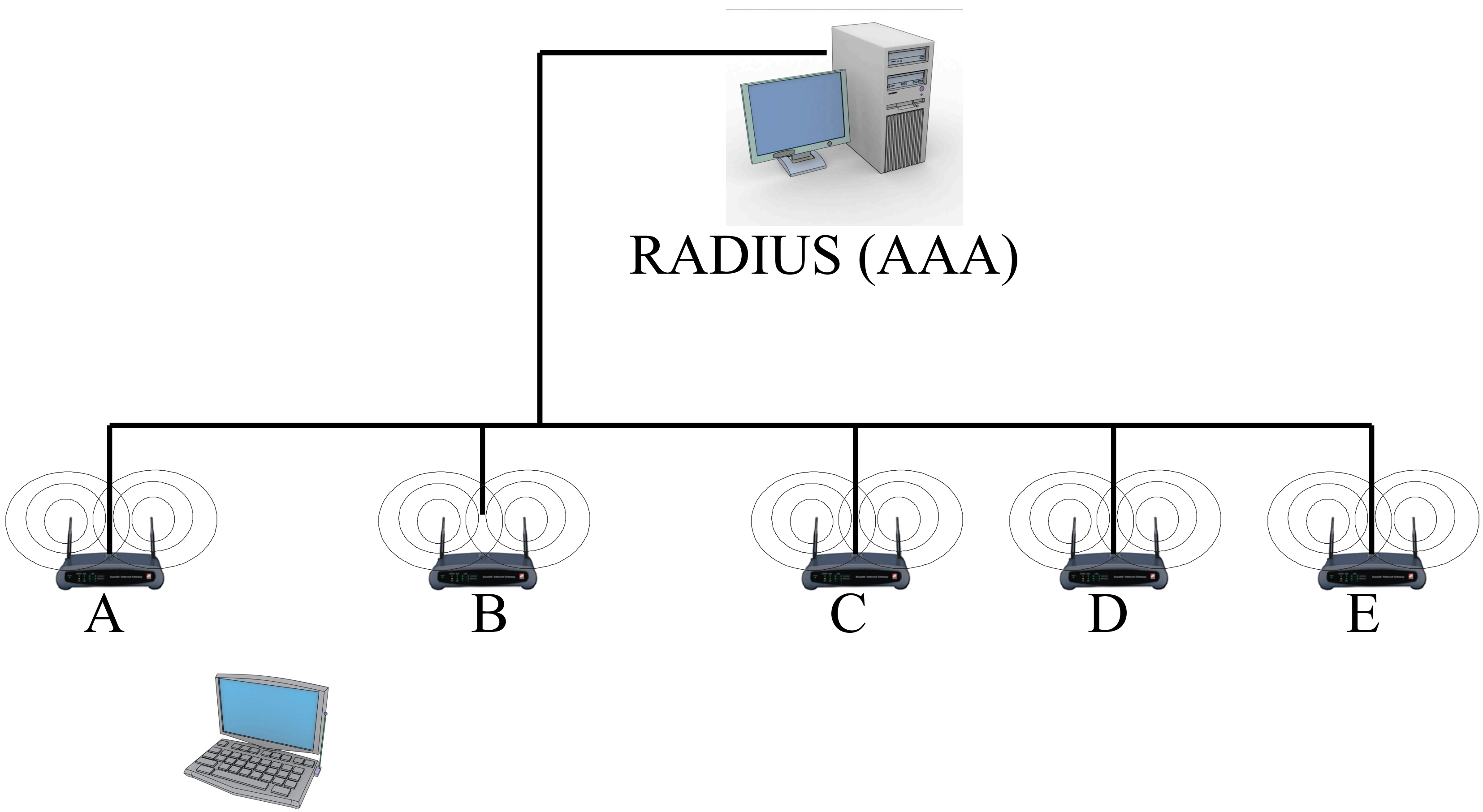
Changes Needed

- TGi
  - Use of RSN IE reserved bit
  - Derivation method for RK
- IETF
  - None

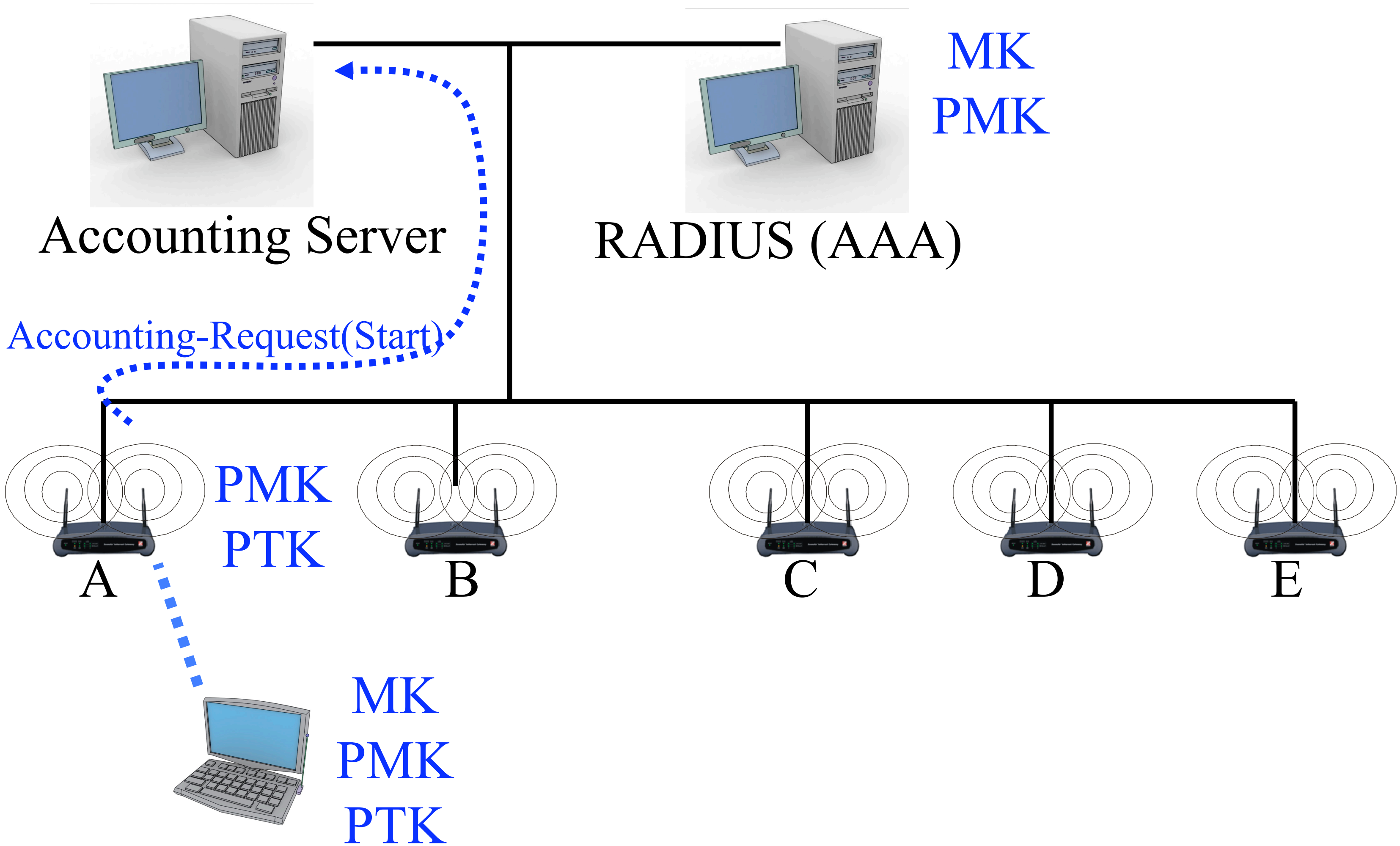
Proactive Key Distribution (TGi)

- Extend Neighbor Graphs and Proactive Caching (IEEE 11-02-758r1.ppt) to support key distribution by the AS/RS
- Eliminates problems with sharing key material amongst multiple APs
  - Easily extended to support WAN roaming
  - Extendable to support Interworking

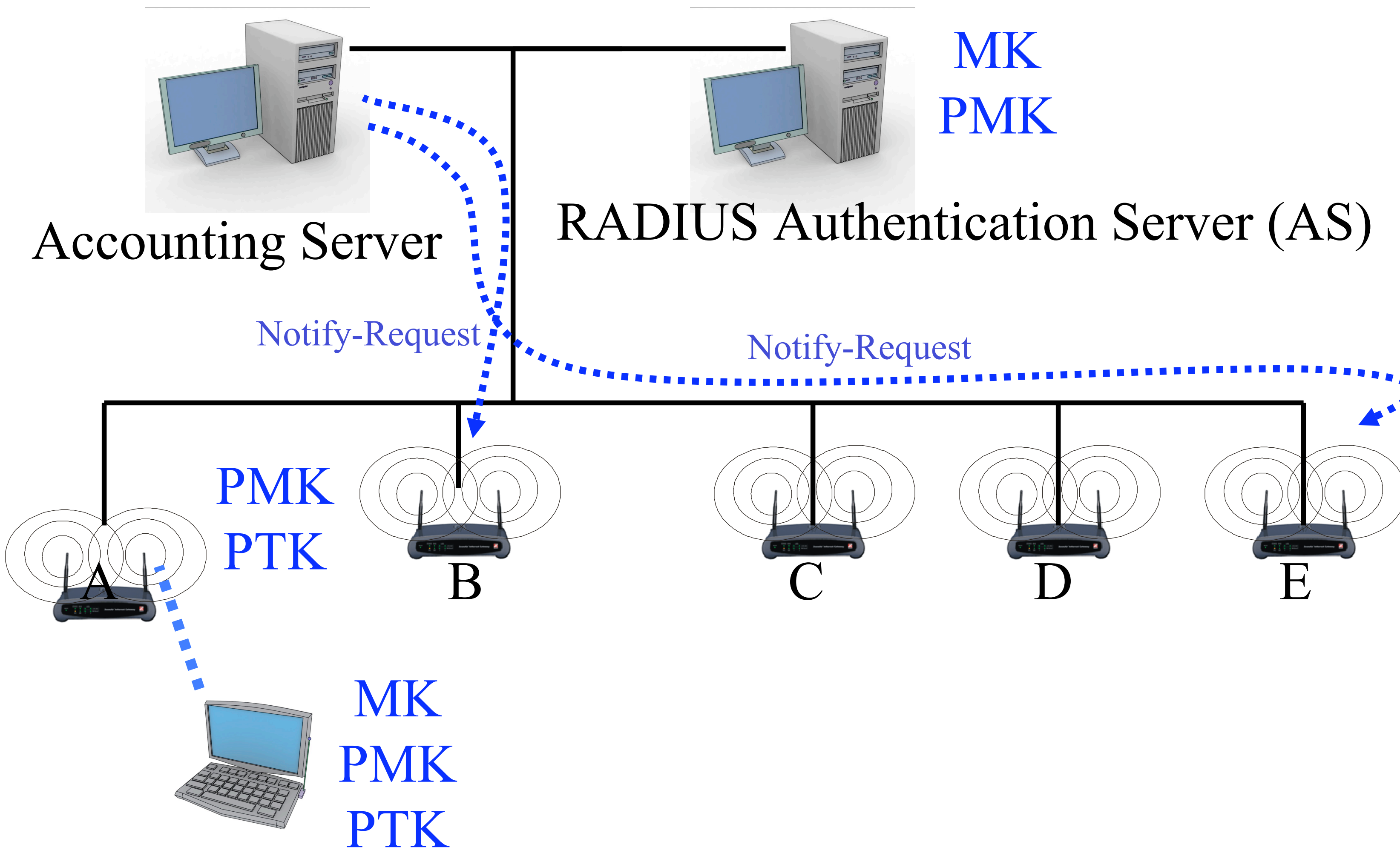
Pre Association



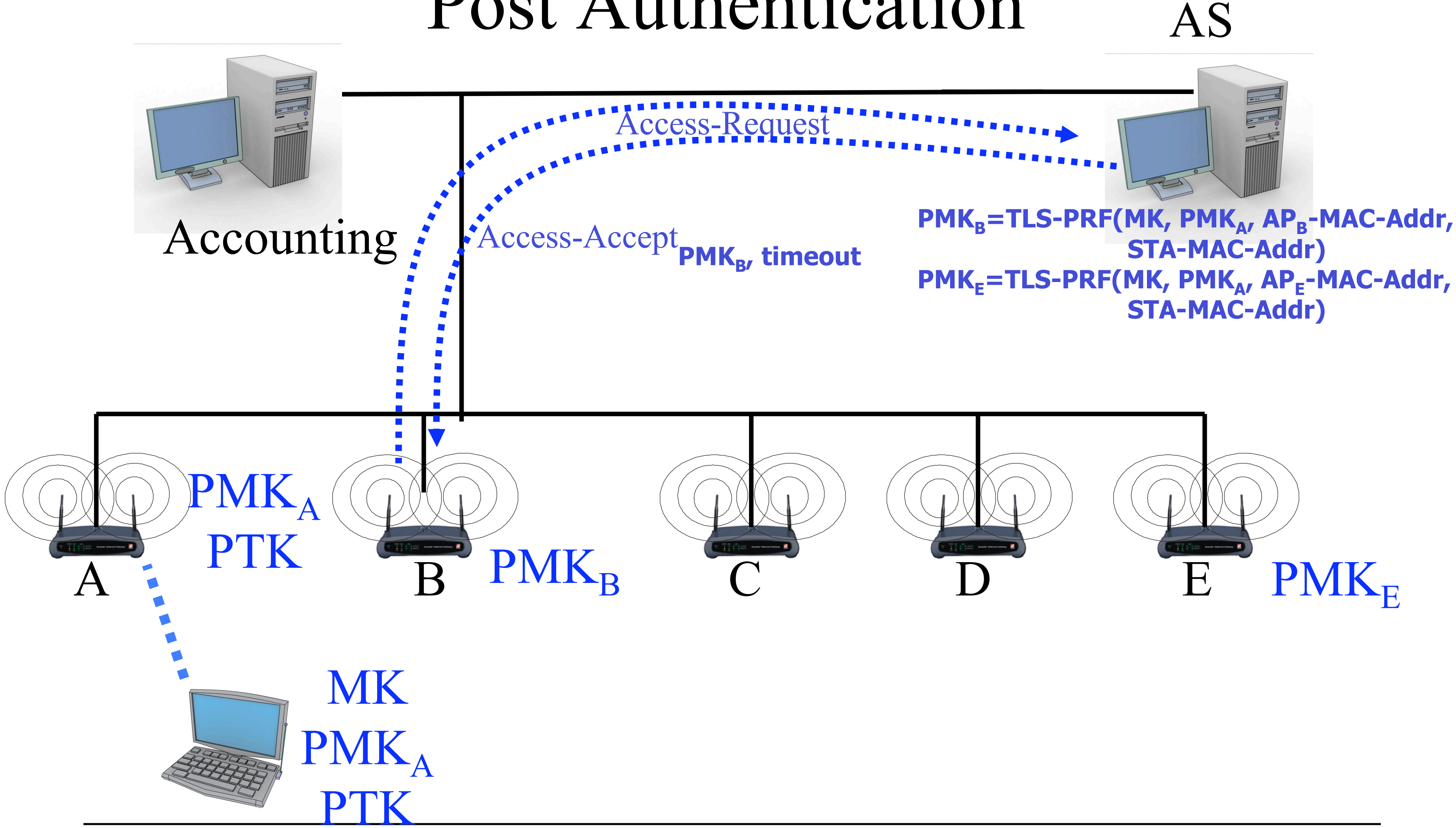
Post Authentication and 4-handshake



Proactive Key Distribution



Proactive Key Distribution Post Authentication



PMK Generation

- Each AP is given a unique PMK per roam, or generation.
- The PMK for the AP for that generation becomes that generation's PMK.

Generations

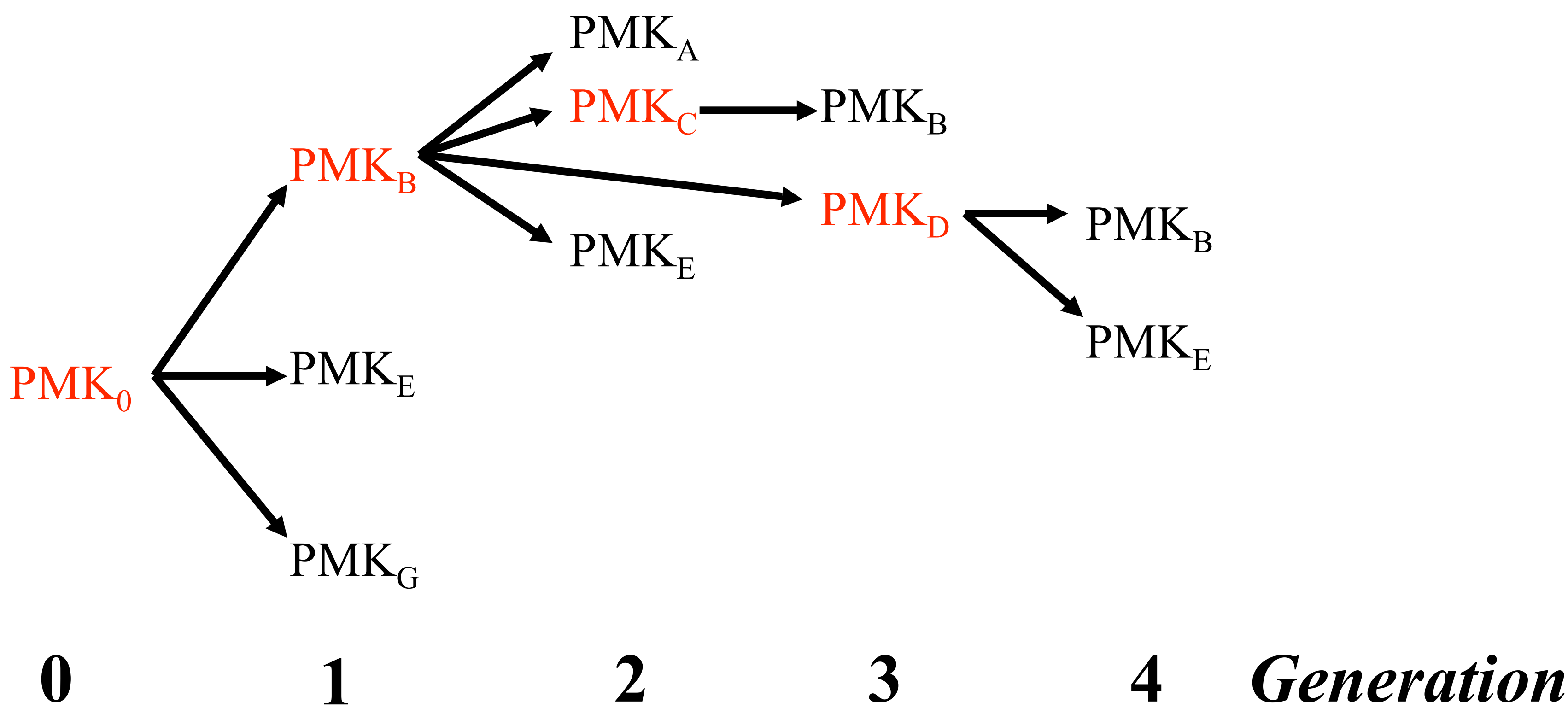
- Generation:
  - PMK_0 = TLS-PRF(MK,"client EAP encryption", client-Hello.random, serverHello.random)
  - PMK_{1-B} = TLS-PRF(MK, PMK_0,AP_B-MAC-Addr, STA-MAC-Addr)
  - PMK_{1-E} = TLS-PRF(MK, PMK_0,AP_E-MAC-Addr, STA-MAC-Addr)
- STA roams to AP_B:
  - $PMK_{1-B} \Rightarrow PMK_0$

PMK Synchronization

- STA maintains PMK for each generation (can be combined with timeouts to require only a "window" of PMK's).
- First message of 4-way handshake (or derivative) from AP indicates the generation PMK held at the AP.
- STA derives the correct PMK based on the provided generation.
- Maintains security even if synchronization is lost, i.e. latest generation PMK hasn't arrived at AP yet.
- Also maintains security (and speed) even if roam to a non-neighbor (but was at one time before timeout).

Synchronization Example

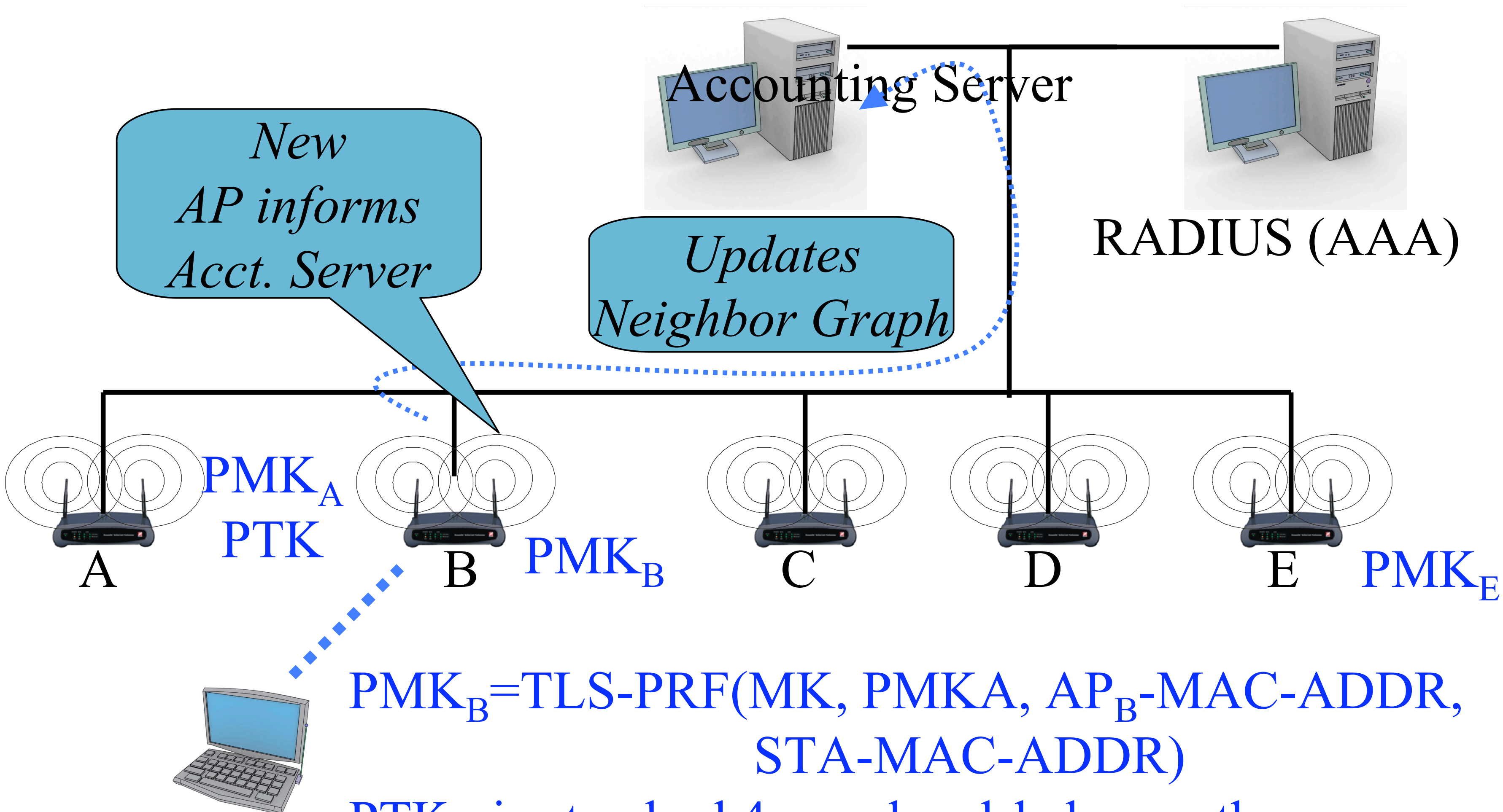
STA roam pattern: A→B →C ≈ D



How do the AP and STA know that Fast Roaming is supported?

- STA asserts that it supports fast roaming by setting a bit (use of the current reserved bits) in the RSN information field element in the REASSOCIATION-REQUEST.
- AP asserts the same bit in the REASSOCIATION-RESPONSE *if and only if* the AP supports fast roaming and it is provisioned with a derived PMK for the STA.
- If either bits are unset, then a full reauthentication MUST be done.

Proactive Key Distribution
STA Roam to AP B



PMK_B=TLS-PRF(MK, PMKA, AP_B-MAC-ADDR, STA-MAC-ADDR)

PTK via standard 4-way handshake or other means

Roaming Functions Handled at Accounting Server

- Reduces exposure of MK, i.e. the MK remains only at the AS and STA
- Reduces load on AS
- Takes advantage of the already defined accouting process

Changes Needed

- TGi
  - Use of RSN IE reserved bit
  - Addition of "Generation" field to Message 1 of 4-way handshake
- IETF
  - Define two new RADIUS messages to install derived PMK's at AP's (draft-arbaugh-radius-handoff-00.txt)
  - Perhaps one or two new RADIUS attributes

Open Source Availability

- The University of Maryland and Samsung Electronics will provide an open source implementation under both the GPL and *BSD style licenses shortly.

External Review

- This proposal will be submitted to an academic conference for peer review soon.