IEEE 802.11i Overview

Nancy Cam-Winget, Cisco Systems

Tim Moore, Microsoft

Dorothy Stanley, Agere Systems

Jesse Walker, Intel Corporation

Presentation Objectives

- Communicate what IEEE 802.11i is
- Communicate how 802.11i works
- Receive feedback on the above

Agenda

- **Conceptual Framework**
- **Architecture**
- **Security Capabilities Discovery**
- **Authentication**
- **Key Management**
- **Data Transfer**
- **Other Features**

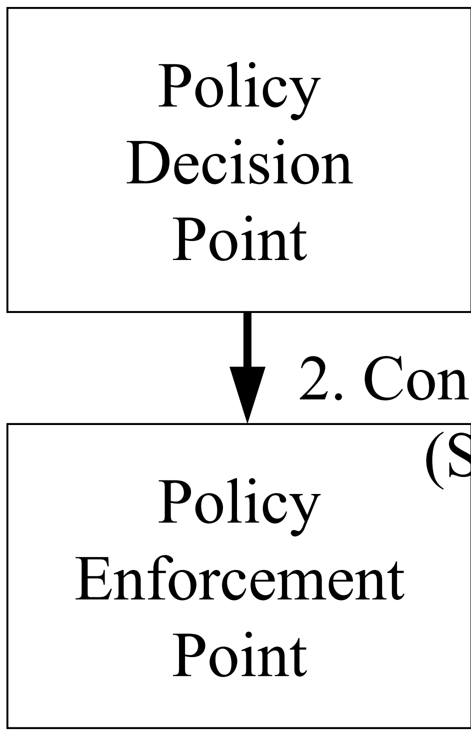
Terminology

- Authentication Server (AS)
- Access Point (AP)
- Station (STA)
- Master Key (MK)
- Pairwise Master Key (PMK)

Generic Policy Model

- Policy Decision Point (PDP) = Logical component making policy decisions
- Policy Enforcement Point (PEP) = Logical component enforcing policy decisions
- Session Decision Token (SDT) = data structure representing a policy decision
- Session Enforcement Token (SET) = data structure used to enforce policy decision

Model Operation



1. Issue Session Decision Token (SDT)

2. Construct Session Enforcement Token (SET) from SDT and distribute

3. Use SET to enforce policy decision

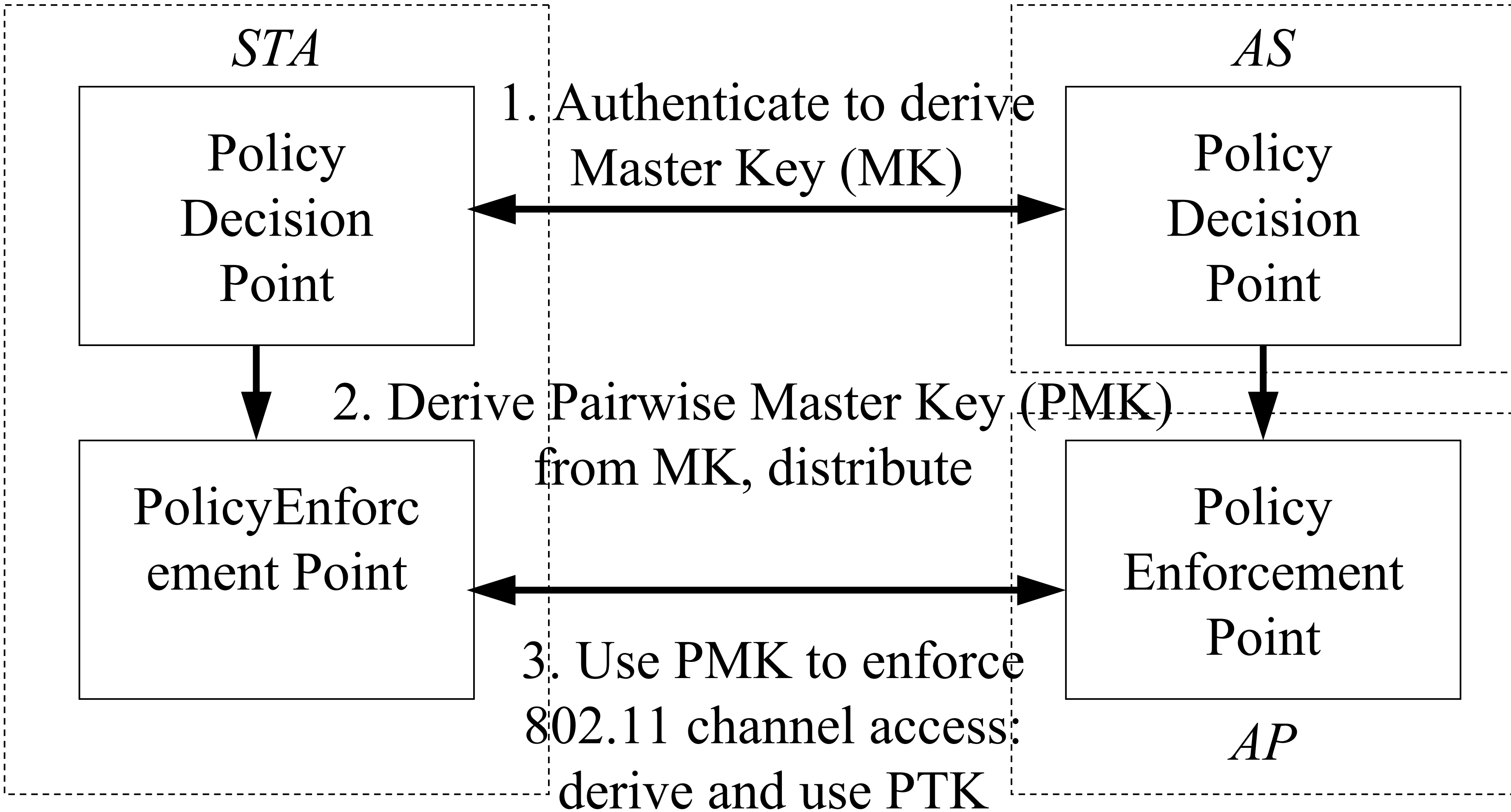
Application to 802.11i (1)

- Two Policy Decision Points: STA and AS
- Policy Decision: allow 802.11 access?
- Policy decision decided by authentication
- 802.11 Policy Decision Token called ***Master Key*** (MK)
  - MK = ***symmetric key*** representing ***Station's*** (STA) and ***Authentication Server's*** (AS) decision during this ***session***
  - Only STA and AS can possess MK
    - MK possession demonstrates authorization to make decision

Application to 802.11i (2)

- Two Policy Enforcement Points: STA and AP
- 802.11 Policy Enforcement Token called ***Pairwise Master Key*** (PMK)
  - PMK is a ***fresh symmetric key*** controlling ***STA's*** and ***Access Point's*** (AP) access to 802.11 channel during this ***session***
  - Only STA and AS can manufacture PMK
    - PMK derived from MK
    - AS distributes PMK to AP
  - PMK possession demonstrates authorization to access 802.11 channel during this session

Application to 802.11i (3)



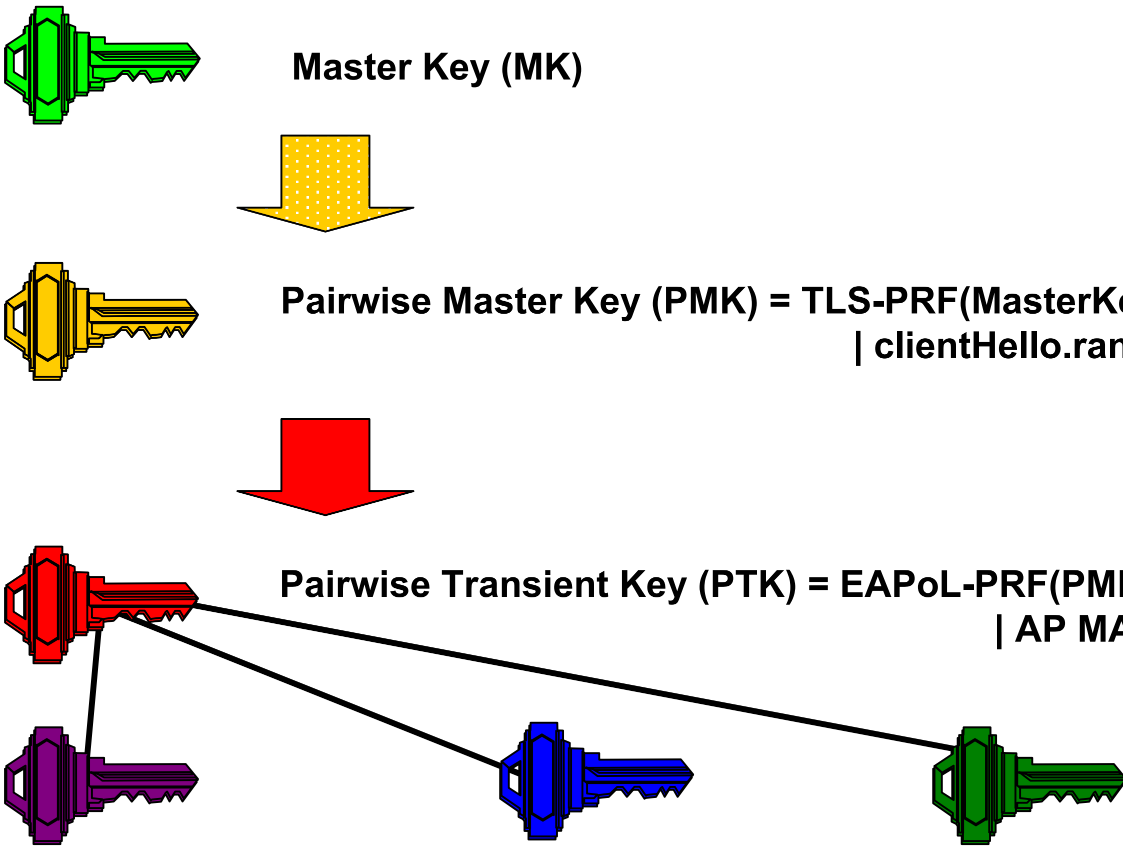
Observations

- Both AP and STA must make same authentication decision
  - Or no communication via 802.11 channel
- MK ≠ PMK
  - Or AP could make access control decisions instead of AS
- PMK is bound to *this* STA and *this* AP
  - Or another party can masquerade as either
- MK is fresh and bound to *this* session between STA and *AS*
  - Or MK from another session could represent decision for *this* session
- PMK is fresh and bound to *this* session between STA and *AP*
  - Or old PMK could be used to authorize communications on *this* session
- When AP ≠ AS, need to ***assume*** AS will ***not***
  - Masquerade as STA or AP
  - Reveal PMK to any party but AP

Architectural Components

- Key hierarchy
  - Pairwise Keys, Group Keys
- EAP/802.1X/RADIUS
- Operational Phases
  - Discovery, Authentication, Key Management, Data transfer

Pairwise Key Hierarchy



Master Key (MK)

Pairwise Master Key (PMK) = TLS-PRF(MasterKey, “client EAP encryption” | clientHello.random | serverHello.random)

Pairwise Transient Key (PTK) = EAPoL-PRF(PMK, AP Nonce | STA Nonce | AP MAC Addr | STA MAC Addr)

Key Confirmation Key (KCK) – PTK bits 0–127

Key Encryption Key (KEK) – PTK bits 128–255

Temporal Key – PTK bits 256–*n* – can have cipher suite specific structure

Pairwise Keys

- Master Key – represents positive access decision
- Pairwise Master Key – represents authorization to access 802.11 medium
- Pairwise Transient Key – Collection of operational keys:
  - Key Confirmation Key (KCK) – used to bind PMK to the AP, STA; used to prove possession of the PMK
  - Key Encryption Key (KEK) – used to distribute Group Transient Key (GTK)
  - Temporal Key (TK) – used to secure data traffic

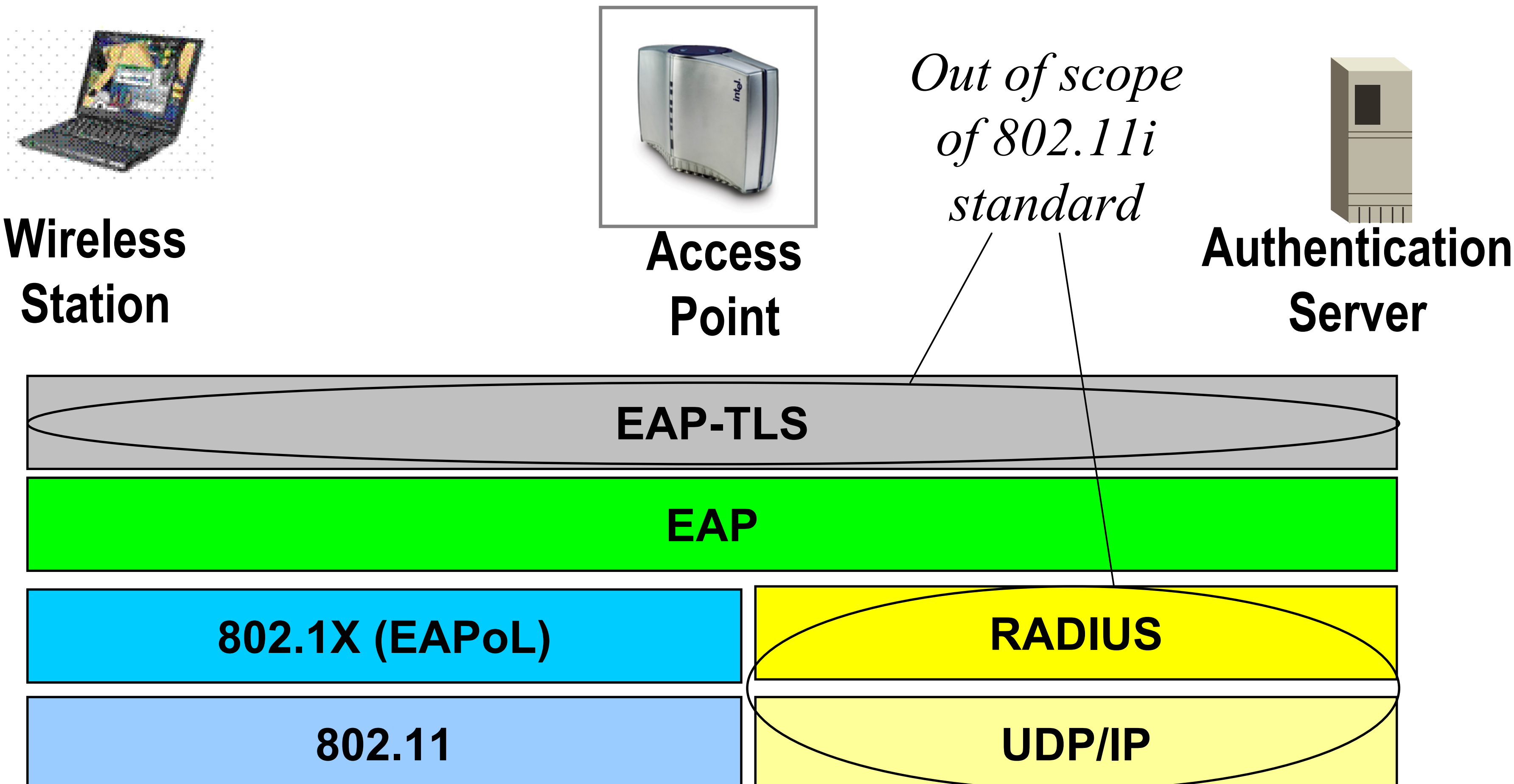
Group Keys

- Group Transient Key (GTK) – An operational key:
  - Temporal Key – used to “secure” multicast/broadcast data traffic
- 802.11i specification defines a “Group key hierarchy”
  - Entirely gratuitous: impossible to distinguish GTK from a randomly generated key

More Terminology

- 802.1X or EAPoL
- EAP
- TLS
- EAP-TLS
- RADIUS

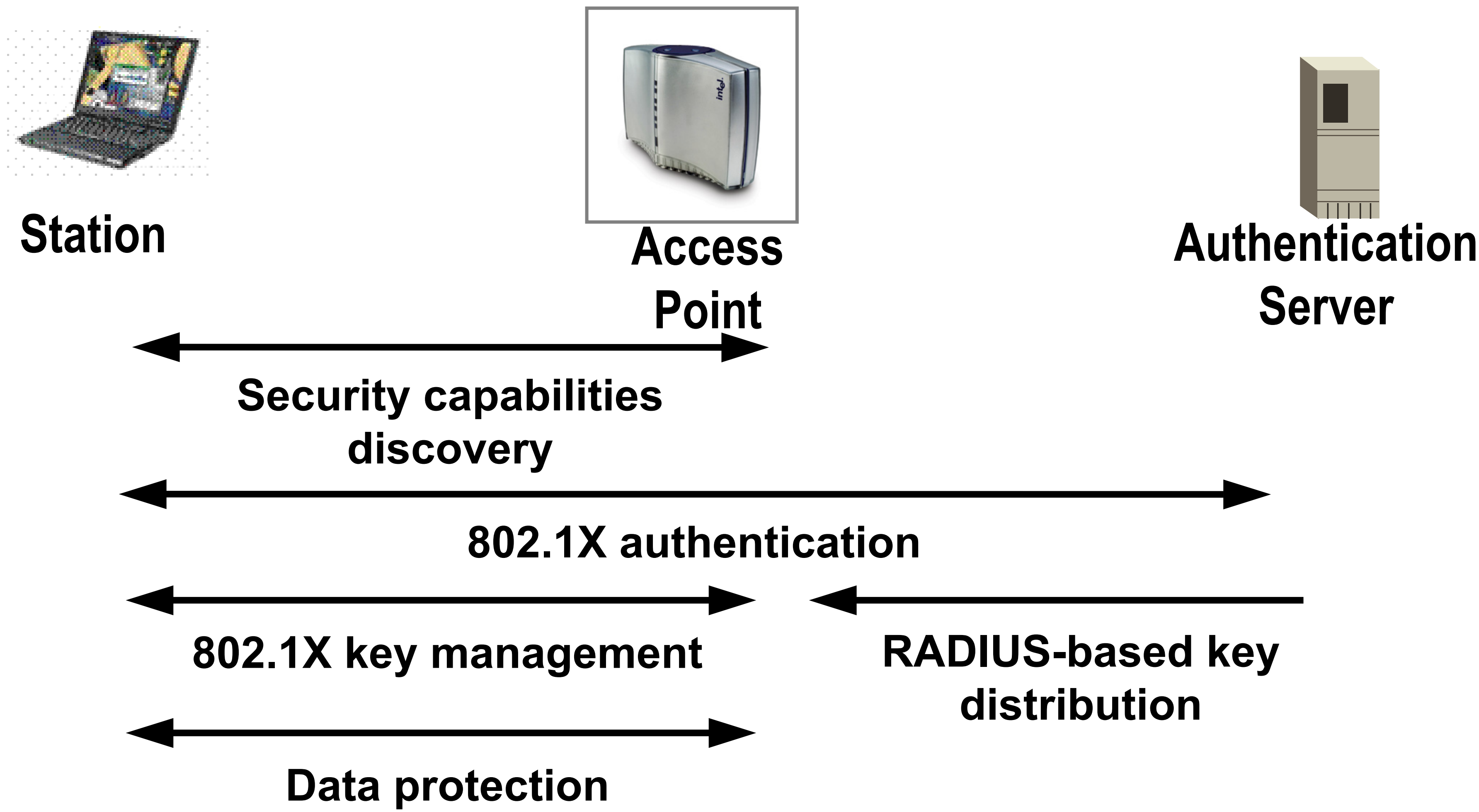
Authentication and Key Management Architecture (1)



Authentication and Key Management Architecture (2)

- EAP is end-to-end transport for authentication between STA, AS
- 802.1X is transport for EAP over 802 LANs
- AP proxies EAP between 802.1X and backend protocol between AP and AS
- Backend protocol outside 802.11 scope
  - But RADIUS is the *de facto* transport for EAP over IP networks
- Concrete EAP authentication method outside 802.11 scope
  - But EAP-TLS is the *de facto* authentication protocol, because the others don’t work

802.11 Operational Phases



Purpose of each phase (1)

- Discovery
  - Determine promising parties with whom to communicate
  - AP advertises network security capabilities to STAs
- 802.1X authentication
  - Centralize network admission policy decisions at the AS
  - STA determines whether it does indeed want to communicate
  - Mutually authenticate STA and AS
  - Generate Master Key as a side effect of authentication
  - Generate PMK as an access authorization token

Purpose of each phase (2)

- RADIUS-based key distribution
  - AS moves (not copies) PMK to STA's AP
- 802.1X key management
  - Bind PMK to STA and AP
  - Confirm both AP and STA possess PMK
  - Generate fresh PTK
  - Prove each peer is live
  - Synchronize PTK use
  - Distribute GTK

Discovery Overview

- AP advertises capabilities in Beacon, Probe Response
  - SSID in Beacon, Probe provides hint for right authentication credentials
    - Performance optimization only; no security value
  - RSN Information Element advertises
    - All enabled authentication suites
    - All enabled unicast cipher suites
    - Multicast cipher suite
- STA selects authentication suite and unicast cipher suite in Association Request

Discovery

Station

Access Point

Probe Request →

← **Probe Response + RSN IE (AP supports CCMP Mcast, CCMP Ucast, 802.1X Auth)**

802.11 Open System Auth →

← **802.11 Open Auth (success)**

Association Req + RSN IE (STA requests CCMP Mcast, CCMP Ucast, 802.1X Auth) →

← **Association Response (success)**

Discovery Process Commentary

- Conformant STA declines to associate if its own policy does not overlap with AP's policy
- Conformant AP rejects STAs that do not select from offered suites
- 802.11 Open System Authentication retained for backward compatibility– no security value
- No protection during this phase – capabilities validated during key management
- Capabilities advertised in an ***RSN Information Element*** (RSN IE)

The RSN IE

Element ID = 48	Element Length	Version = 1	Group key suite	Paiwise suite count	Pairwise suite list	Auth Suite List	Auth suite list	Capabilities
1 octet	1 octet	2 octets	4 octets	2 octets	4xPW count	2 octets	4xAuth count	2 octets

- Element Length – the size of element in octets.
- Version 1 means
  – Supports 802.1X key management per 802.11i
  – Supports CCMP

Suite Selectors

OUI - 3 octets	Type - 1 octet

- Constituent of
  - Authentication suite list – authentication and key management methods
  - Pairwise cipher suite list – crypto used for key distribution, unicast
  - Group cipher suite list – crypto used for multicast/broadcast

Some Suite Selector

Authentication and Key Management Suites

- 00:00:00:1 – 802.1X authentication and key management
- 00:00:00:2 – no authentication, 802.1X key management
- Vendor Specific

Pairwise or Group Cipher Suites

- 00:00:00:1 – WEP
- 00:00:00:2 – TKIP
- 00:00:00:3 – WRAP
- 00:00:00:4 – CCMP
- 00:00:00:5 – WEP-104
- Vendor Specific

Capabilities

Preauthentication	Group key unicast	# replay counters	Reseved
b0	b1	b2/b3	b4-b15

- Preauthentication – 1 means supported
- Group key unicast – for WEP only
- # replay counters – for QoS support
- Reserved – set to 0 on transmit, ignored on receive

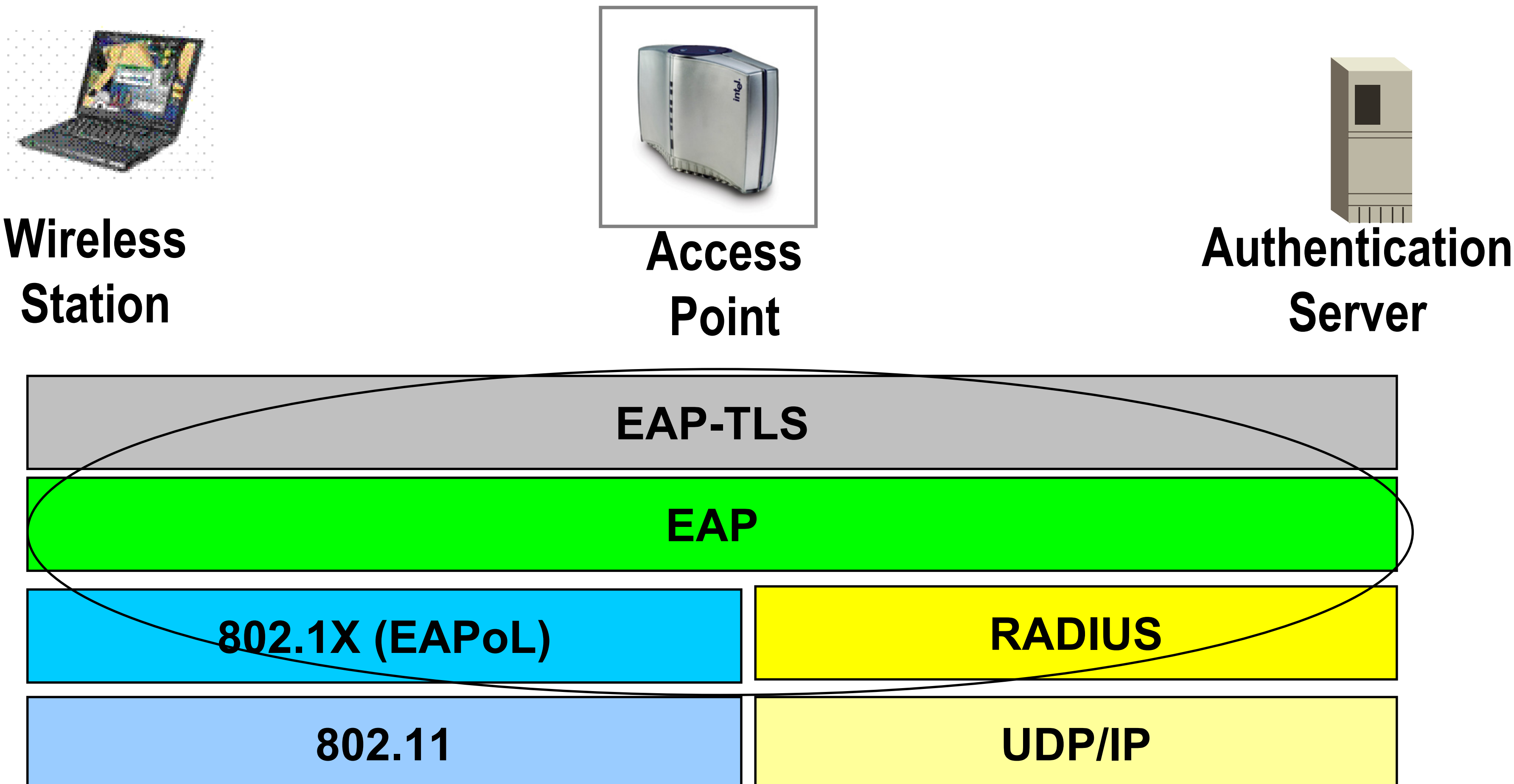
Discovery Summary

- At the end of discovery
  - STA knows
    - The alleged SSID of the network
    - The alleged authentication and cipher suites of the network
    - These allow STA to locate correct credentials, instead of trial use of credentials for every network
  - The AP knows which of its authentication and cipher suites the STA allegedly chose
  - A STA and an AP have established an 802.11 channel
  - The associated STA and AP are ready authenticate

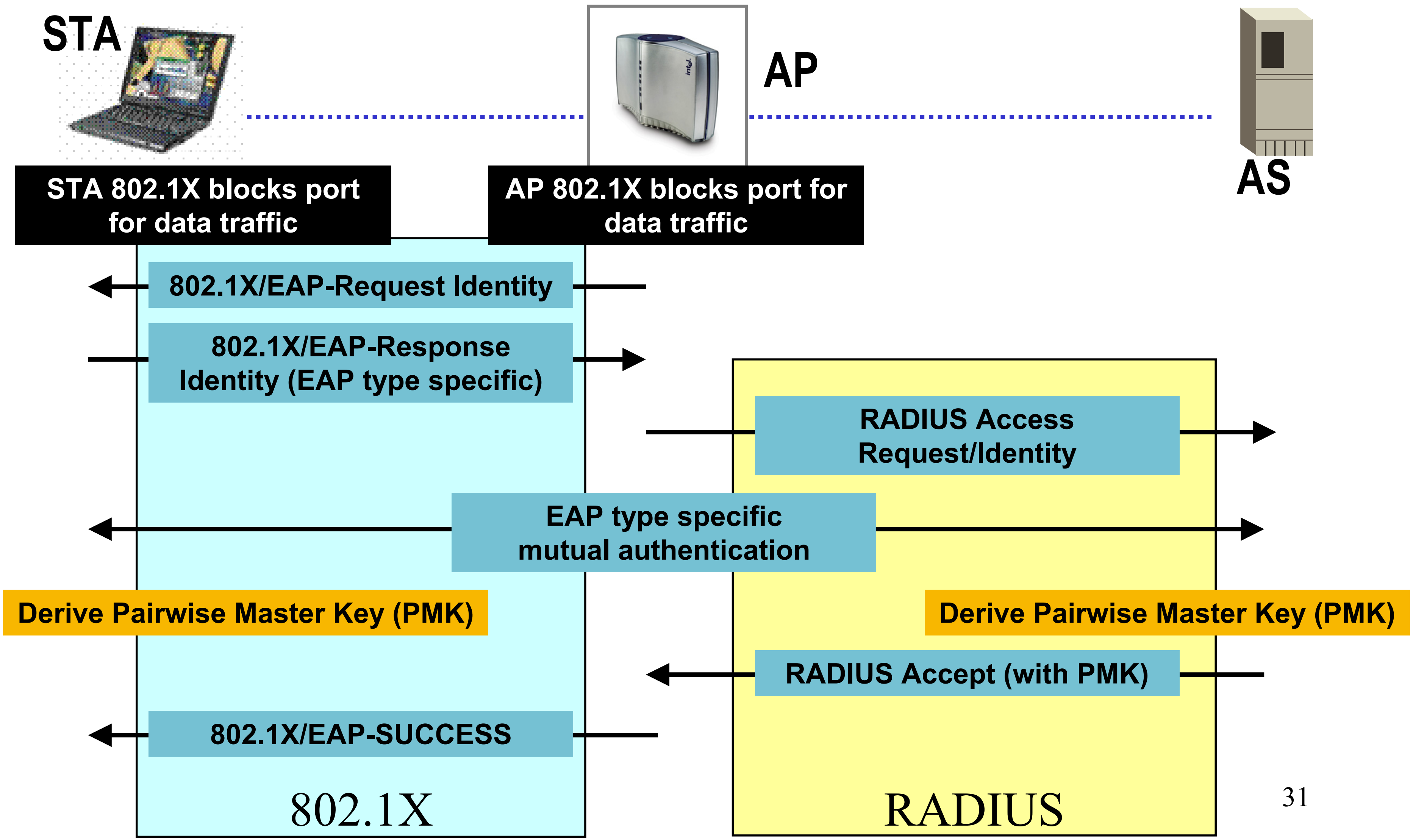
Authentication Requirements

- Establish a session between AS and STA
- Establish a mutually authenticated session key shared by AS and STA
  - Session $\Rightarrow$ key is fresh
  - Mutually authenticated $\Rightarrow$ bound only to AS and STA
- Defend against eavesdropping, man-in-the-middle attacks, forgeries, replay, dictionary attacks against either party
  - Cannot expose non-public portions of credentials
- Identity protection *not* a goal
  - Can't hide the MAC address

Authentication Components



Authentication Overview



Digging Deeper: EAP (1)

- EAP = ***Extensible Authentication Protocol***
  - Defined in RFC 2284
  - Being revised due to implementation experience and poor specification (rfc2284bis)
- Developed for PPP, but 802.1X extends EAP to 802 LANs
- Design goal: allow "easy" addition of new authentication methods
  - AP need not know about new authentication method
  - Affords great flexibility
- EAP is a ***transport*** optimized for authentication, ***not*** an authentication method itself
  - Relies on "concrete" methods plugged into it for authentication

Digging Deeper: EAP (2)

- Eases manageability by centralizing
  - Authentication decisions
  - Authorization decisions
- Well matched economically to 802.11:
  - Minimizes AP cost by moving expensive authentication to AS
  - AP unaware of the authentication protocol
- EAP supports "chained" authentications naturally
  - First do mutual authentication of devices, then user authentication, etc…
  - … so well suited to multi-factor authentication

Digging Deeper: EAP (3)

- AS initiates all transactions
  - Request/Response protocol
  - STA can’t recover from AS or AP problems
  - This affords AS with limited DoS attack protection
- AS tells the STA the authentication protocol to use
  - STA must decline if asked to use weak methods it can’s support
- AS sends EAP-Success to STA if authentication succeeds
  - STA breaks off if AS authentication fails
- AS breaks off communication if authentication fails

Digging Deeper: EAP (4)

- EAP provides no cryptographic protections
  - No defense against forged EAP-Success
  - Relies on concrete method to detect all attacks
  - No cryptographic binding of method to EAP
- No strong notion of AS-STA binding
  - "Mutual" authentication and binding must be inherited from concrete method
- Legacy 802.1X has no strong notion of a session
  - EAP's notion of session problematic, very weak, implicit
  - Relies on session notion within concrete method
  - Key identity problematic
  - 802.11i fixes *some* of this (see key management discussion below)

802.1X

- Defined in IEEE STD 802.1X-2001
- Simple
  - Simple transport for EAP messages
  - Runs over all 802 LANs
  - Allow/deny port filtering rules
- Inherits EAP architecture
  - Authentication server/AP (aka "Authenticator")/STA (aka "Supplicant")

RADIUS (1)

- RADIUS is ***not*** part of 802.11i; back-end protocol is ***out of scope***
  - But RADIUS is the *de facto* back-end protocol
- RADIUS defined in RFC 2138
- Request/response protocol initiated by AP
  - Encapsulates EAP messages as a RADIUS attribute
  - Response can convey station-specific parameters to the AP as well
- 4 messages
  - **Access-Request** – for AP → AS messages
  - **Access-Challenge** – for AS → AP messages forwarded to STA
  - **Access-Accept** – for AS → AP messages indicating authentication success
  - **Access-Reject** – for AS → AP message indicating authentication failure

RADIUS (2)

- RADIUS data origin authenticity
  - AP receives weak data origin authenticity protection
    - Relies on ***static*** key AP shares with AS
    - AP inserts a random challenge into each RADIUS request
    - AS returns MD5(response data | challenge | key) with response
  - No cryptographic protection to the AS
    - AS relies on security of the AP-AS channel for protection
  - Trivial attack strategy:
    - Interject forged requests into the correct place in the request stream
    - RADIUS server will generate valid response

RADIUS (3)

- RADIUS key wrapping defined in RFC 2548
  - Non-standard cross between 1-time pad scheme and MD5 in "CBC" mode

    digest1 $\leftarrow$ MD5(secret | response data | salt), ciphertext1 $\leftarrow$ plaintext1 $\oplus$ digest1
    digest2 $\leftarrow$ MD5(secret | ciphertext1), ciphertext2 $\leftarrow$ plaintext2 $\oplus$ digest2
    digest3 $\leftarrow$ MD5(secret | ciphertext2), ciphertext3 $\leftarrow$ plaintext2 $\oplus$ digest3
    …
  - Uses ***static*** key AP shares with AS
  - No explicit binding of key to AP, STA
  - Great deployment care and vigilance needed to prevent key publication!!

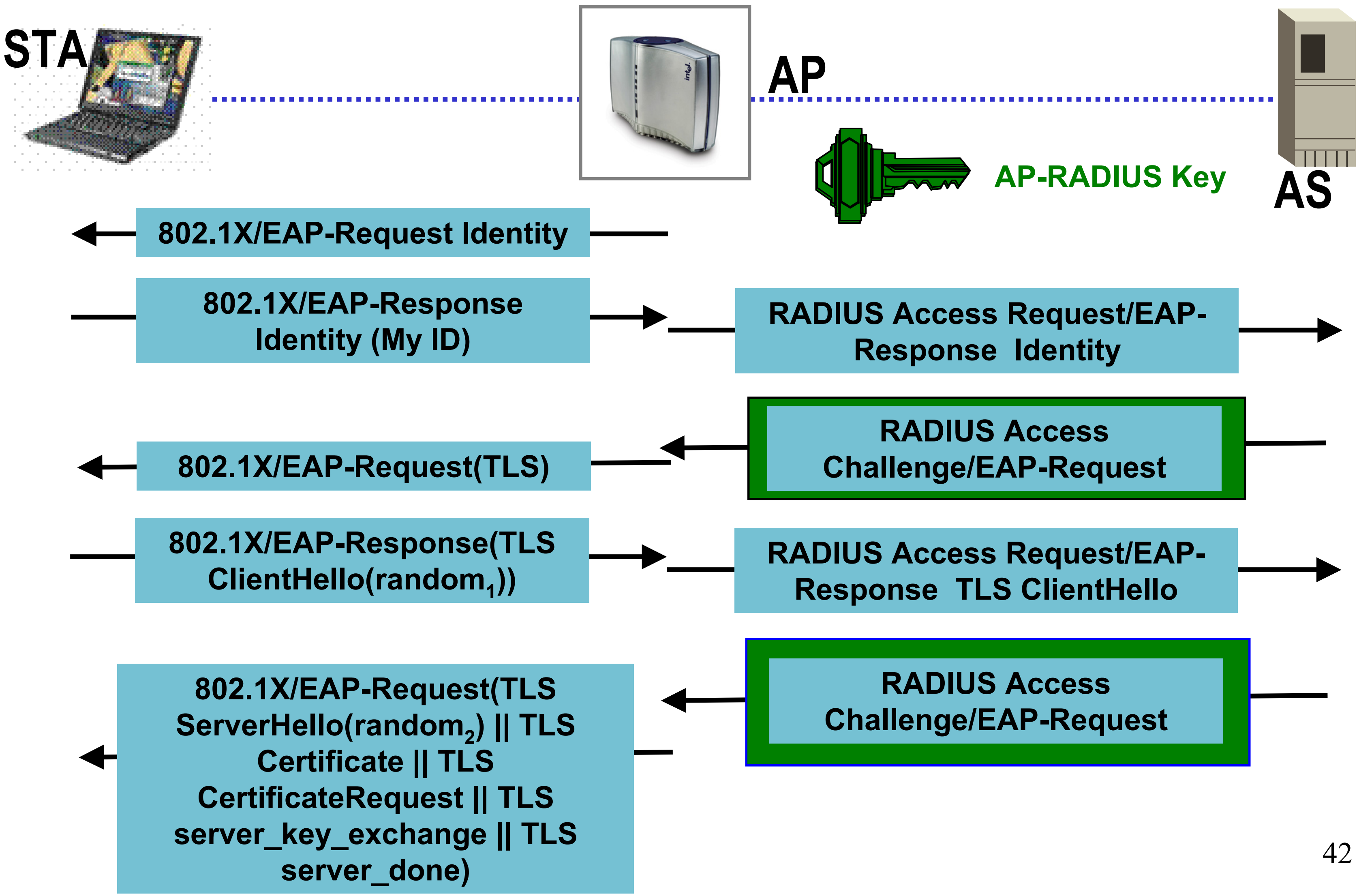
Is Glass Half Full/Empty?

- Reasons to hope
  - Vendors working diligently to replace RADIUS with DIAMETER
  - DIAMETER can use CMS (RFC 3369) to distribute keys
  - G. Chesson, T. Hardjono, R. Housley, N. Ferguson, R. Moskowitz, and J. Walker have sketched a better architecture
- Reasons to despair
  - DIAMETER community misapprehends keying as a data transport instead of a binding problem – not solving the right problem!!
    - And vendors want to use IPsec instead of CMS
    - How to do DIAMETER key management if using CMS?
  - Work on the better architecture has stalled

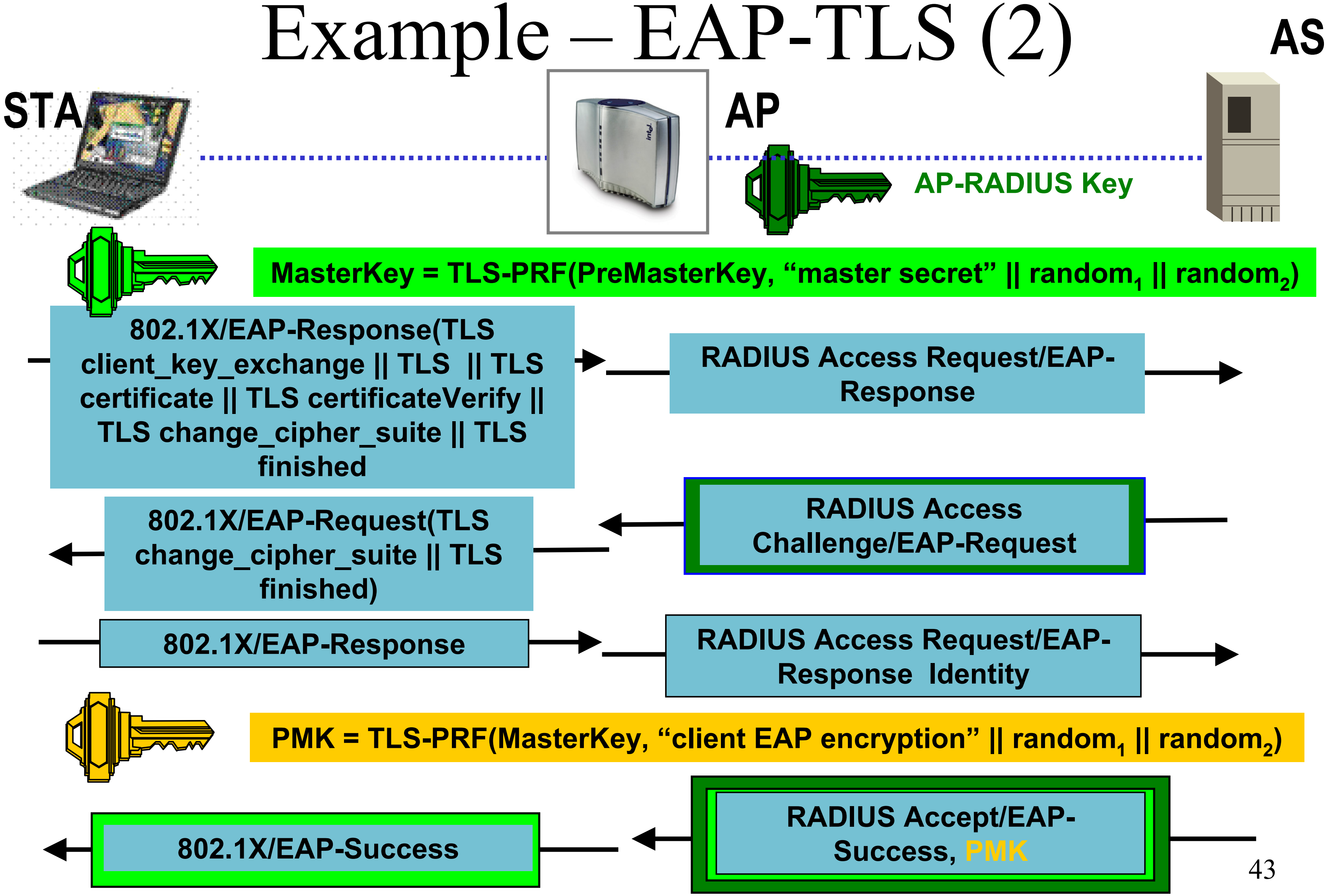
Digging Deeper: EAP-TLS

- EAP-TLS is ***not*** part of 802.11i; neither is any other specific authentication method
- But EAP-TLS is the *de facto* 802.11i authentication method
  - Can meet all 802.11i requirements
  - Other widely deployed methods do not
- EAP-TLS = TLS Handshake over EAP
  - EAP-TLS defined by RFC 2716
  - TLS defined by RFC 2246
- Always requires provisioning AS certificate on the STA
- Mutual authentication requires provisioning STA certificates

Example –EAP-TLS (1)



Example – EAP-TLS (2)



Why is Cert Provisioning Required for EAP-TLS?

- Using public CA instead of CA known to root only legitimate APs is insecure:
  - Malicious rogue AP cheap
  - Unlike e-commerce server, rogue AP controls STA’s view of network topology
    - Can block any and all traffic to and from STA
    - Can’t get to the CRL server
  - Analog of reverse DNS lookup on AS not possible until after session established

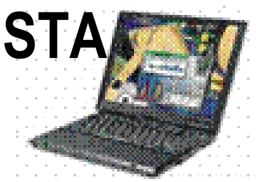
Authentication Summary

- At the end of authentication
  - The AS and STA have established a session if concrete EAP method does
  - The AS and STA possess a mutually authenticated Master Key if concrete EAP method does
    - Master Key represents decision to grant access based on authentication
  - STA and AS have derived PMK
    - PMK is an authorization token to enforce access control decision
  - AS has distributed PMK to an AP (hopefully, to the STA's AP)

802.1X Key Management

- Original 802.1X key management hopelessly broken, so redesigned by 802.11i
- New model:
  - Given a PMK, AP and AS use it to
    - Derive a fresh PTK
  - AP uses KCK and KEK portions of PTK to distribute Group Transient Key (GTK)
- Limitations:
  - No explicit binding to earlier association, authentication
    - Relies on temporality, PMK freshness for security
  - Keys are only as good as back-end allows

Key Management Overview

STA 

 AP

AS 

Step 1: Use RADIUS to push PMK from AS to AP

Step 2: Use PMK and 4-Way Handshake to derive, bind, and verify PTK

Step 3: Use Group Key Handshake to send GTK from AP to STA

Step 1: Push PMK to AP

- RADIUS: we’ve seen it is all already…

EAPoL Key Message

Descriptor Type – 1 octet	
Key Information – 2 octets	**Key Length – 2 octets**
Replay Counter – 8 octets	
Nonce – 32 octets	
IV – 16 octets	
RSC – 8 octets	
Key ID – 8 octets	
MIC – 16 octets	
Data Length – 2 octets	**Data – n octets**

EAPoL Key Message Fields (1)

- Descriptor – value = 254, means 802.11i Key Message
- Key Information – see below
- Replay counter – used to sequence GTK updates, detect replayed STA requests
- Nonce – used to establish liveness, key freshness
- IV – when used, to make key wrapping scheme probabilistic
- RSC – where to start the replay sequence counter (required for broadcast/multicast)

EAPoL Key Message Fields (2)

- Key ID – reserved for a real key naming scheme, if ever invented
- MIC – Message Integrity Code, to prove data origin authenticity
- Data length – number of octets of data transported by this Key Message
- Data – When used, the data to be transported
  - RSN IEs from discovery
    - STA's RSN IE in 4-Way Handshake Message 2
    - AP's RSN IE in 4-Way Handshake Message 3
  - GTK in Group Key Handshake Message 1

Key Information (1)

3 bits Version	1 bit Key Type	2 bits Key Index	1 bit Install	1 bit Ack	1 bit MIC	1 bit Secure	1 bit Error	1 bit Reque st	4 bits Reserved

- Version
  - 1: HMAC-MD5 MIC, RC4 Key Wrap
  - 2: HMAC-SHA1 MIC, NIST AES Key Wrap
- Type
  - 0: Group Key
  - 1: Pairwise Key
- Index
  - 0 if Type = 1 (Pairwise)
  - 1 or 2 if Type = 0 (Group)

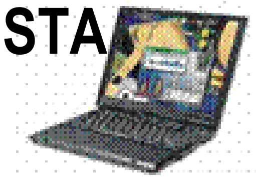
Key Information (2)

- Install: Set only by AP
  - 0: Don’t use PTK to protect data link yet
  - 1: Begin using PTK to protect data link
- Ack: Set only by AP
  - 0: Don’t reply to this message
  - 1: Reply to this message
- MIC:
  - 0: MIC not present in this message
  - 1: MIC present in this message

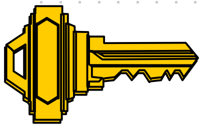
Key Information (3)

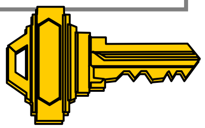
- Secure: Set only by AP
  - 0: Initialization not yet complete
  - 1: Initialization complete
- Error: Set only by STA, to report TKIP MIC errors
- Request: Set only by STA, to request new key
- Reserved: set to 0 on transmit, ignored on receive

Step 2: 4-Way Handshake

STA


AP



PMK

PMK


Pick Random ANonce

EAPoL-Key(Reply Required, Unicast, ANonce) ←

Pick Random SNonce, Derive PTK = EAPoL-PRF(PMK, ANonce | SNonce | AP MAC Addr | STA MAC Addr)

EAPoL-Key(Unicast, SNonce, MIC, STA RSN IE) →

Derive PTK

EAPoL-Key(Reply Required, Install PTK, Unicast, ANonce, MIC, AP RSN IE) ←

EAPoL-Key(Unicast, MIC) →

Install TK

Install TK

4-Way Handshake Discussion (1)

- Assumes: PMK is known ***only*** by STA and AP
  - So architecture ***requires*** a further assumption that AS is a trusted 3rd party
- PTK derived, not transported
  - Guarantees PTK is fresh if ANonce or SNonce is fresh
  - Guarantees Messages 2, 4 are live if ANonce is fresh and unpredictable,
  - Guarantees Message 3 is live if SNonce is fresh and unpredictable
  - PTK derivation binds PTK to STA, AP

4-Way Handshake Discussion (2)

- Message 2 tells AP
  - There is no man-in-the-middle
  - STA possesses PTK
- Message 3 tells STA
  - There is no man-in-the-middle
  - AP possesses PTK
- Message 4 serves no cryptographic purpose
  - Used only because 802.1X state machine wants it

4-Way Handshake Discussion (3)

- Sequence number field used by 4-way handshake only to filter late packets
- Recall PTK ::= KCK | KEK |TK
  - KCK used to authenticate Messages 2, 3, and 4
  - KEK unused by 4-way handshake
  - TKs installed after Message 4
- The discovery RSN IE exchange from alteration protected by the MIC in Messages 2 and 3

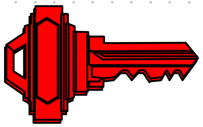
4-Way Handshake Discussion (4)

- Asserting Install bit in Message 3 synchronizes Temporal Key use (data link protections)

STA

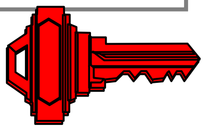
Step3: Group Key Handshake

AP



PTK

PTK



Pick Random GNonce,
Pick Random GTK

Encrypt GTK with KEK

EAPoL-Key(All Keys Installed, ACK, Group Rx, Key Id, Group , RSC, GNonce, MIC, GTK)

Decrypt GTK

EAPoL-Key(Group, MIC)

unblocked data traffic

unblocked data traffic

Group Key Discussion (1)

- GTK encrypted using the KEK portion of PTK
- Group Key Handshake message authenticity protected by KCK portion of PTK
- Group Key Handshake replay protected by EAPoL Replay Counter
- Starting Replay Sequence Counter (RSC) included to minimize replay to STAs joining the group late

Group Key Discussion (2)

- AP ping-pongs GTK between Key ID 1 and 2
  - Send new GTK on new key ID to all associated STAs
  - Then start using new key ID
- GNonce is gratuitous; it plays no useful role in the protocol
  - Putting GNonce into the Beacon would be useful: provide as a hint that group key has changed

One Last Detail

EAPoL-PRF(K, A, B, Len)

$R \leftarrow$ ""

for $i \leftarrow 0$ **to** (Len+159)/160 **do**

$R \leftarrow R$ | HMAC-SHA1(K, A | B | i)

return Truncate-to-len(R, Len)

Example for CCMP:

PTK ← EAPoL-PRF(PMK, "Pairwise key expansion", AP-Addr | STA-Addr | ANonce | SNonce, 384)

Why HMAC-SHA1?

- Because we couldn't think of anything better
- Because that's what IKE and Son-of-IKE use

Key Management Summary

- 4-Way Handshake
  - Establishes a fresh pairwise key bound to STA and AP for this session
  - Proves liveness of peers
  - Demonstrates there is no man-in-the-middle between PTK holders if there was no man-in-the-middle holding the PMK
  - Synchronizes pairwise key use
- Group Key Handshake provisions group key to all STAs

Data Transfer Overview

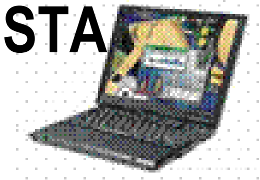
- 802.11i defines 3 protocols to protect data transfer
  - CCMP
  - WRAP
  - TKIP – for legacy devices only
- Three protocols instead of one due to politics

Data Transfer Requirements

- Never send or receive unprotected packets
- Message origin authenticity — prevent forgeries
- Sequence packets — detect replays
- Avoid rekeying — 48 bit packet sequence number
- Eliminate per-packet key – don’t misuse encryption
- Protect source and destination addresses
- Use one strong cryptographic primitive for both confidentiality and integrity
- Interoperate with proposed quality of service (QoS) enhancements (IEEE 802.11 TGe)

Filtering

STA



AP

Association Request

Begin filtering non-802.1X data MPDUs

Association Response

Begin filtering non-802.1X data MPDUs

EAP type specific mutual authentication

4-Way Handshake

Group Key Handshake

Allow data MPDUs protected by pairwise, group keys

Allow data MPDUs protected by pairwise, group keys

Filtering Rules

- If no pairwise key,
  - Do not transmit unicast data MPDU (except 802.1X)
  - Discard received unicast data MPDU (except 802.1X)
- If no group key
  - Do not transmit multicast data MPDU
  - Discard received multicast data MPDU

Replay Mechanisms

- 48-bit IV used for replay detection
  - First four bits of IV indicate QoS traffic class
  - Remaining 44 bits used as counter
  - Decryption/integrity check fail if traffic class bits are altered
  - Sender uses single counter space, but receiver needs one for each traffic class
- AES with CCM or OCB authenticated encryption
  - CCM is mandatory, and OCB is optional
  - Header authentication
  - Payload authentication and confidentiality

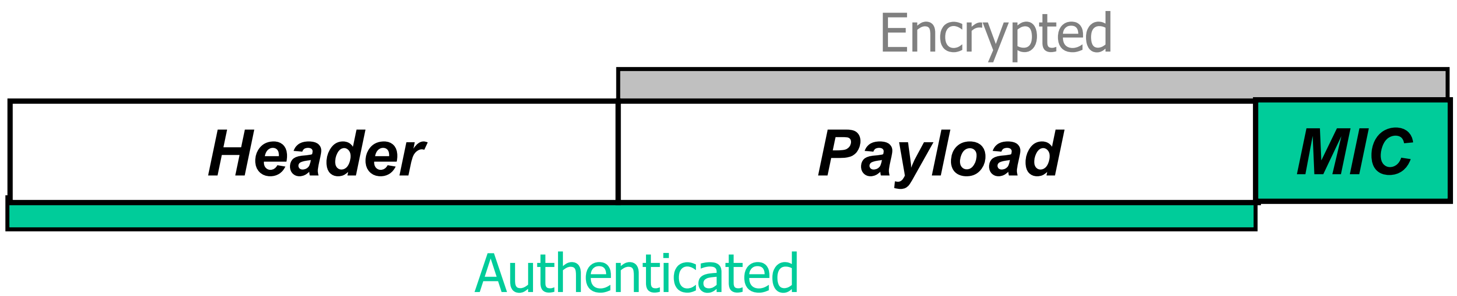
CCMP

- Mandatory to implement: the long-term solution
- Based on ***AES in CCM mode***
  - CCM = Counter Mode Encryption with CBC-MAC Data Origin Authenticity
  - AES overhead requires new AP hardware
  - AES overhead may require new STA hardware for hand-held devices, but not PCs
- An all new protocol with few concessions to WEP
- Protects MPDUs = fragments of 802.2 frames

Counter Mode with CBC-MAC

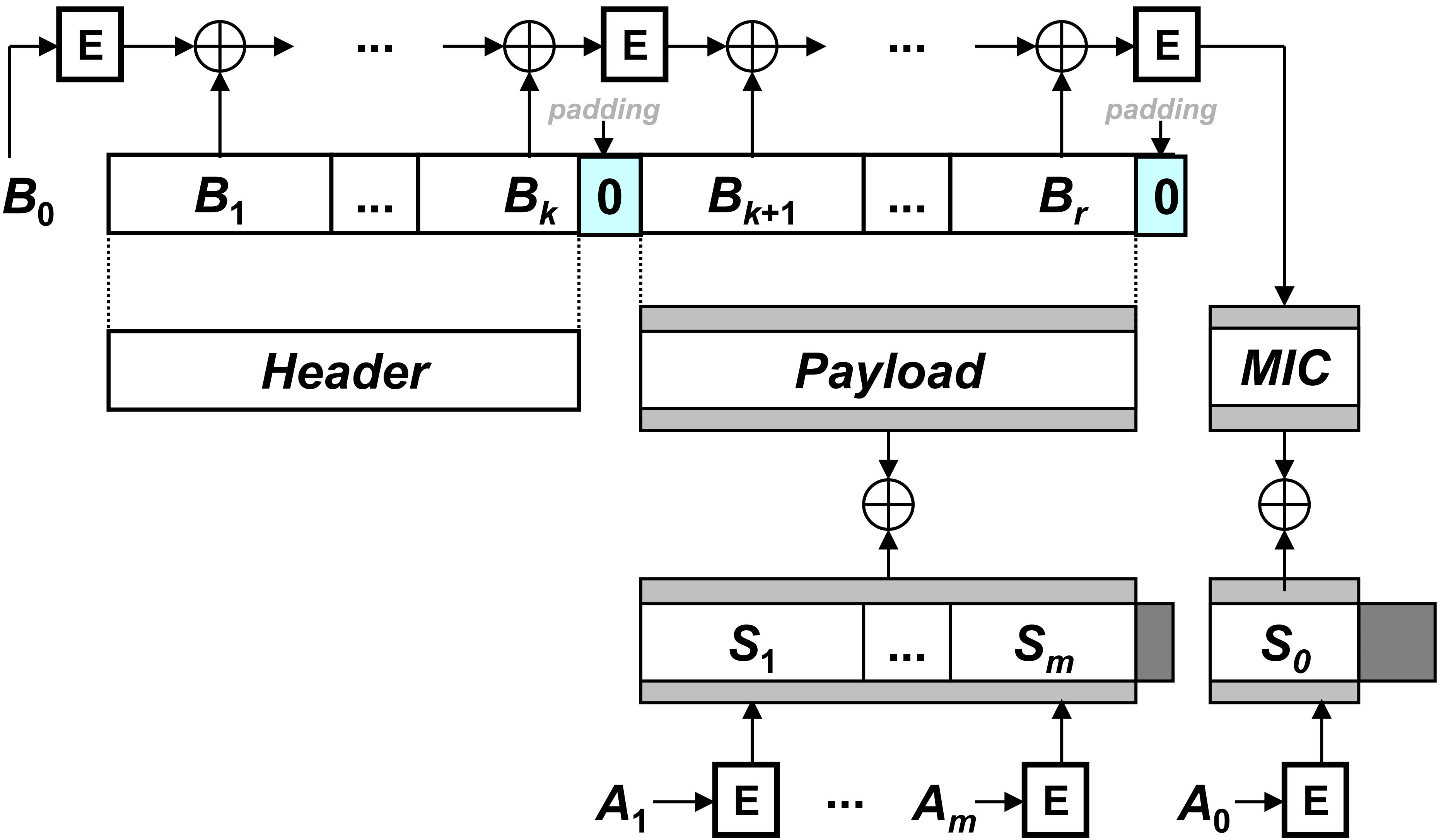
- Authenticated Encryption combining Counter (CTR) mode and CBC-MAC, using a single key
  - Assumes 128 bit block cipher – IEEE 802.11i uses AES
- Designed for IEEE 802.11i
  - By D. Whiting, N. Ferguson, and R. Housley
  - Intended only for packet environment
  - No attempt to accommodate streams

CCM Mode Overview



- Use CBC-MAC to compute a MIC on the plaintext header, length of the plaintext header, and the payload
- Use CTR mode to encrypt the payload
  - Counter values 1, 2, 3, …
- Use CTR mode to encrypt the MIC
  - Counter value 0

Data Transfer



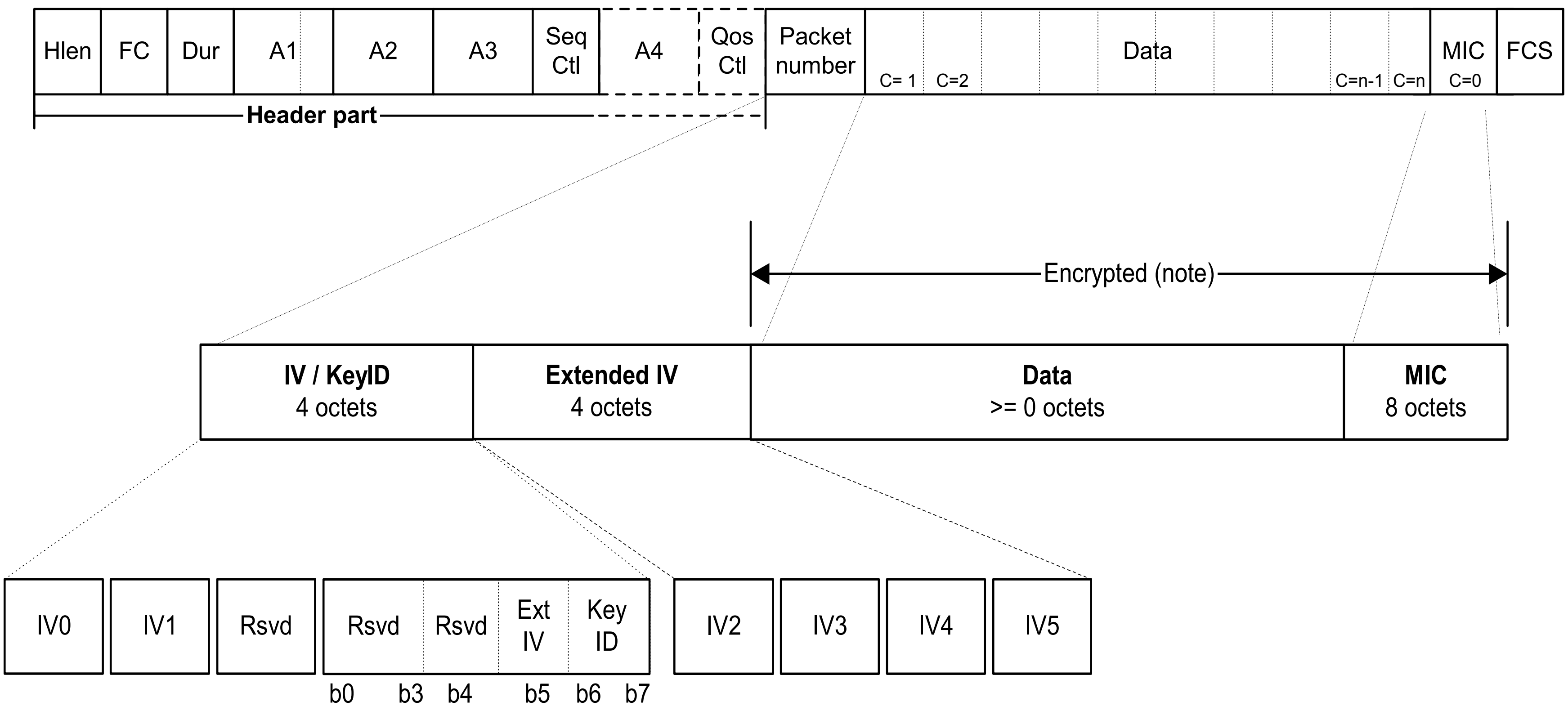
CCM Properties

- CTR + CBC-MAC (CCM) based on a block cipher
- CCM provides authenticity and privacy
  - A CBC-MAC of the plaintext is appended to the plaintext to form an *encoded* plaintext
  - The encoded plaintext is encrypted in CTR mode
- CCM is packet oriented
- CCM can leave any number of initial blocks of the plaintext unencrypted
- CCM has a security level as good as other proposed combined modes of operation, including OCB
  - In particular, CCM is provably secure

CCM Usage by CCMP

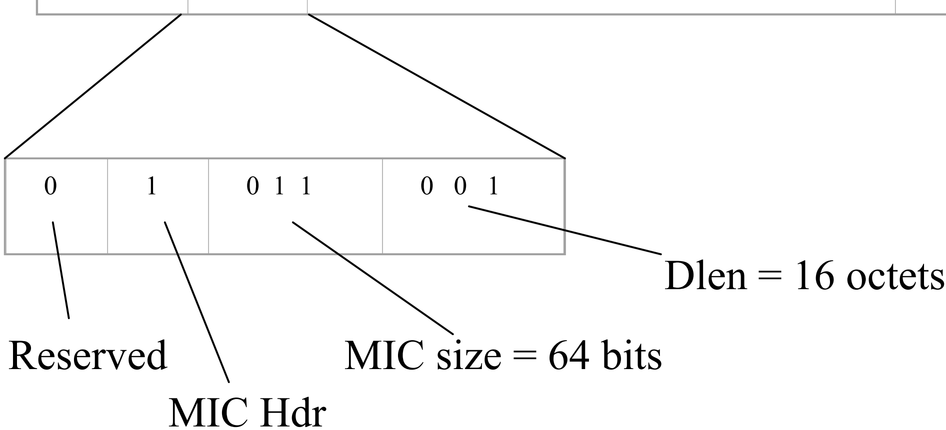
- Temporal key = PKT bits 256-383, GTK 0-127 bits
  - Same 128-bit Temporal key used by both AP and STA
  - CBC-MAC IV, CTR constructions make this kosher
- Key configured by 802.1X
  - CCMP requires a fresh key, or security guarantees voided
- CCMP uses CCM to
  - Encrypt packet data payload
  - Protect packet selected header fields from modification

CCMP MPDU Format



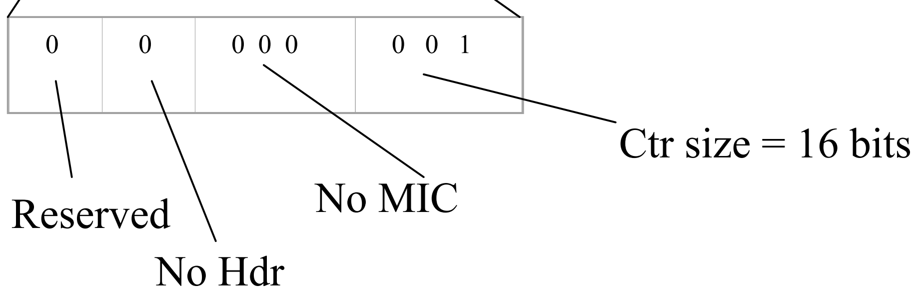
CCMP CBC-MAC IV

Octet Index:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Content	Flag	Transmit Address						Packet Sequence Number (IV)						DLen	



CCMP CTR

Octet Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Content	Flag	Transmit Address						Packet Sequence Number (IV)							Ctr	



Long-term Solution Summary

- Builds on the lessons learned from IEEE 802.10 and IPsec packet protocol designs
  - Relies on proper use of strong cryptographic primitives
- Strong security against all known attacks
- Requires new hardware

WRAP

- The original AES-based proposal for 802.11i
  - Based on AES in OCB mode
- Replaced by CCMP when IPR issues could not be overcome
  - 3 different parties have filed for patents
- Retained in draft because some vendors have implemented WRAP hardware

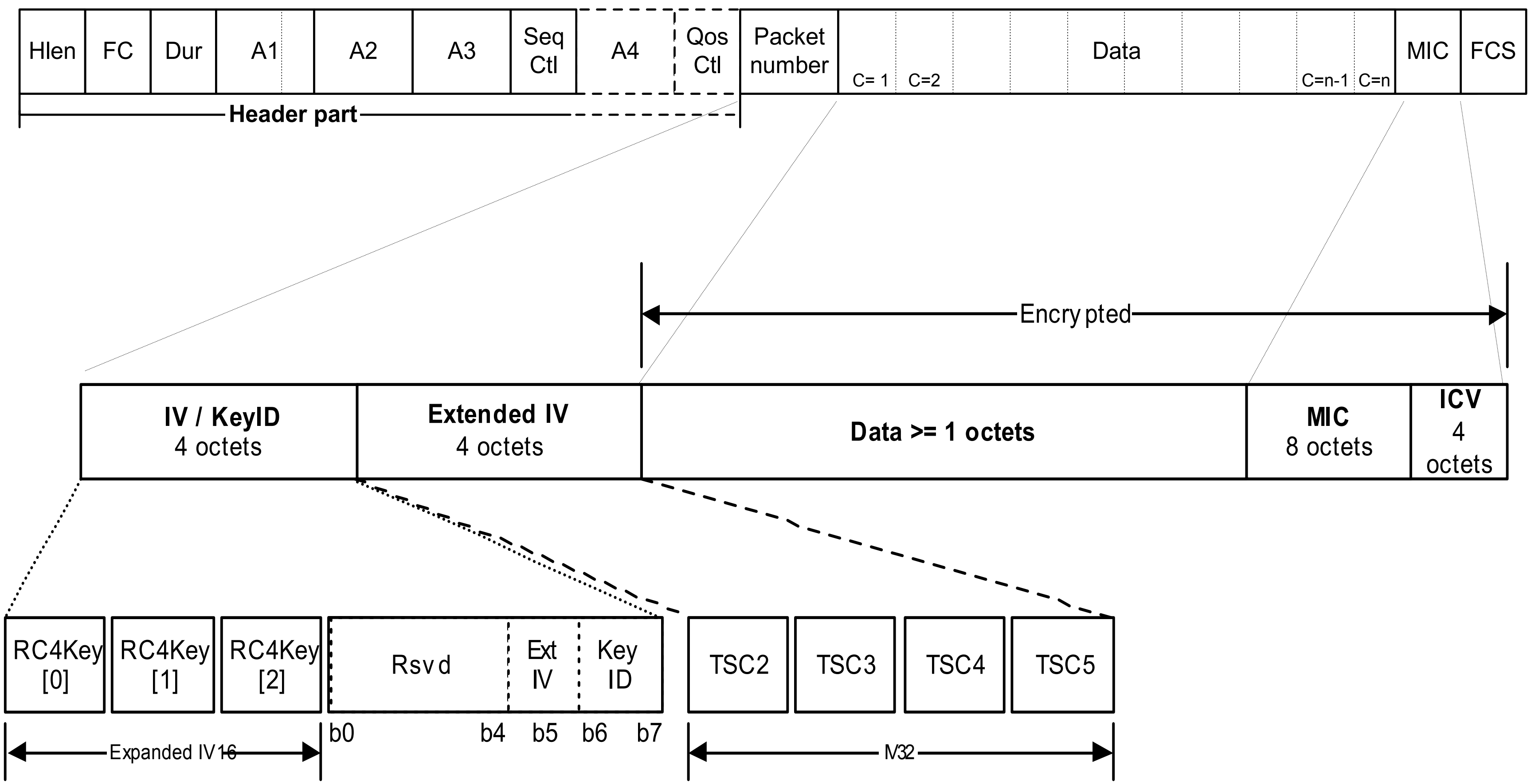
TKIP Summary

- TKIP: *T*emporal *K*ey *I*ntegrity *P*rotocol
- Designed as a wrapper around WEP
  - Can be implemented in software
  - Reuses existing WEP hardware
  - Runs WEP as a sub-component
- Meets criteria for a good standard: everyone unhappy with it

TKIP design challenges

- Mask WEP’s weaknesses…
  - Prevent data forgery
  - Prevent replay attacks
  - Prevent encryption misuse
  - Prevent key reuse
- … On existing AP hardware
  - 33 or 25 MHz ARM7 or i486 already running at 90% CPU utilization before TKIP
  - Utilize existing WEP off-load hardware
  - Software/firmware upgrade only
  - Don’t unduly degrade performance

TKIP MPDU Format



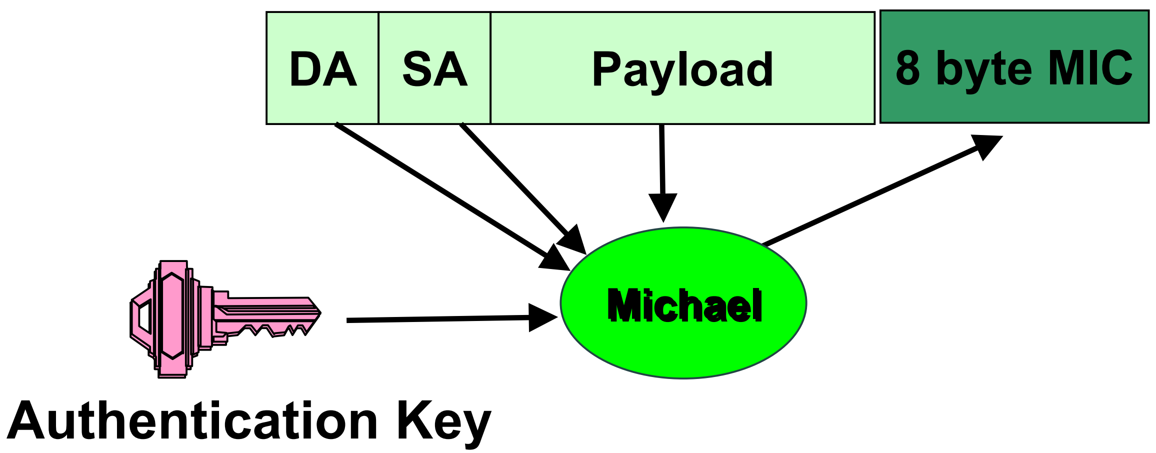
TKIP Keys

- TKIP Keys
  - Temporal encryption key = PTK bits 256-383, GTK 0-127 bits
  - Temporal data origin authenticity keys = PTK bits 384-511, GTK bits 128-255

TKIP Design (1) -- Michael

Protect against forgeries

- Must be cheap: CPU budget ≤ 5 instructions/byte
- Unfortunately is weak: a 2^{29} message attack exists
- Computed over MSDUs, while WEP is over MPDUs
- Uses two 64-bit keys, one in each link direction
- Requires countermeasures: rekey on active attack, rate limit rekeying



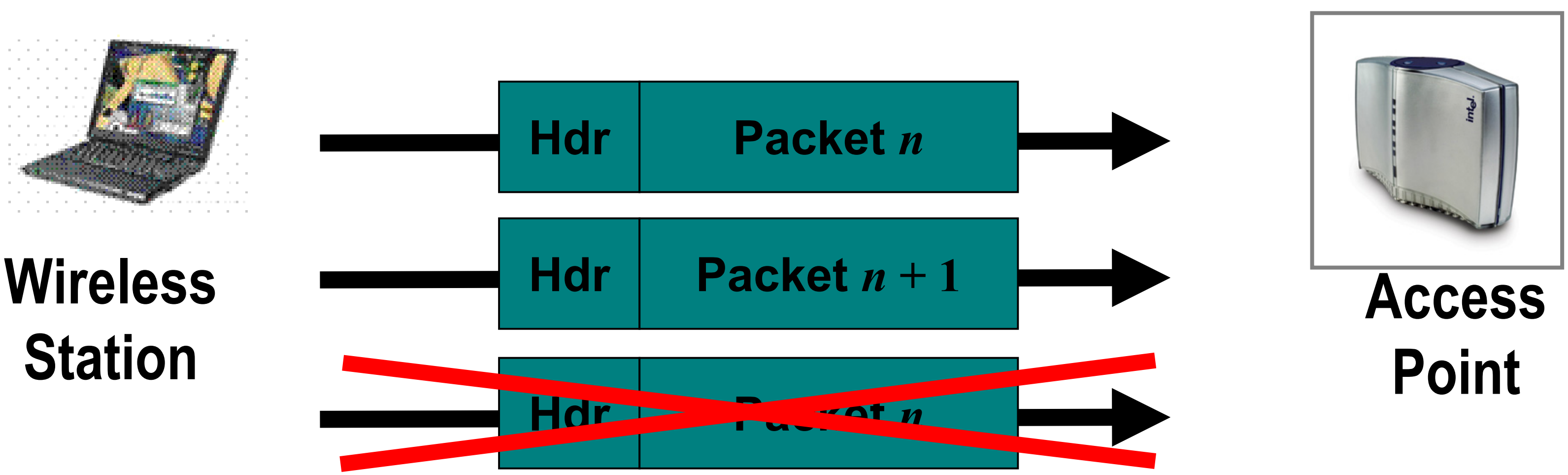
TKIP Countermeasures

- Check CRC, ICV, and IV before verifying MIC
  - Minimizes chances of false positives
  - If MIC failure, almost certain active attack underway
- If an active attack is detected:
  - Stop using keys
  - Rate limit key generation to 1 per minute

TKIP Design (3)

Protect against replay

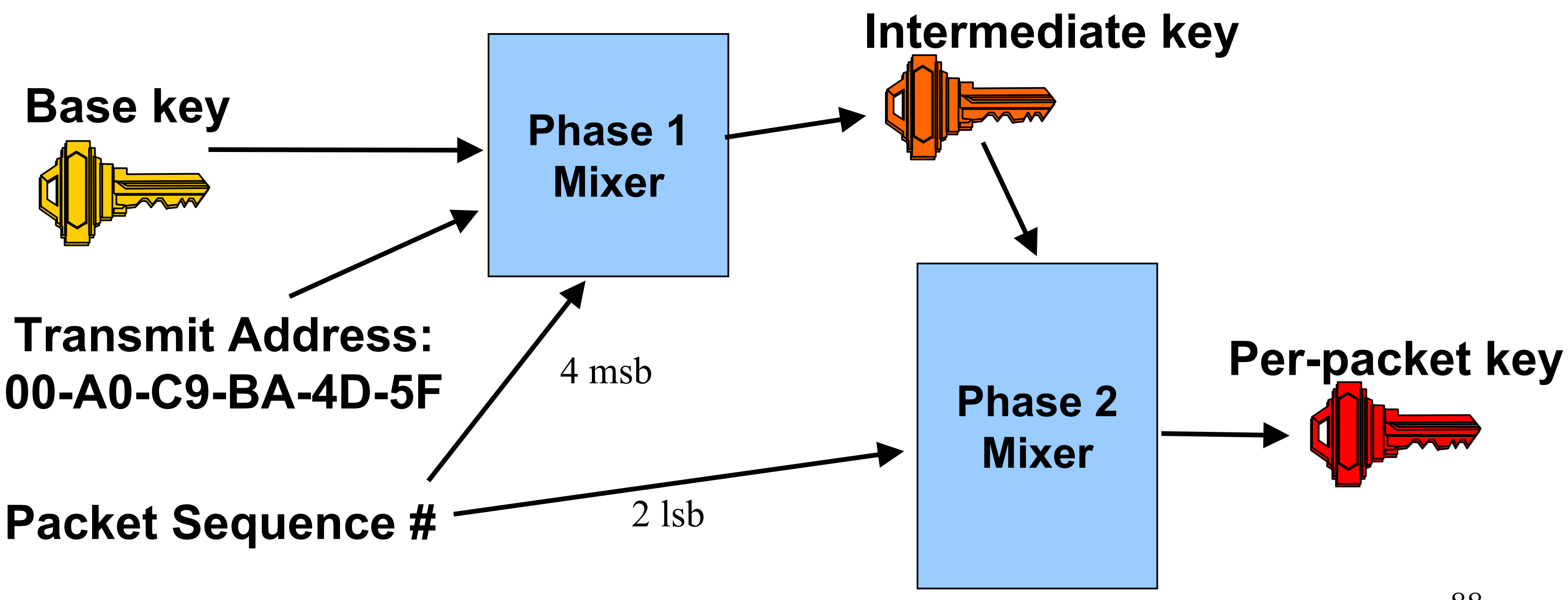
- reset packet sequence # to 0 on rekey
- increment sequence # by 1 on each packet
- drop any packet received out of sequence



TKIP Design (4)

Stop WEP's encryption abuse

- Build a better per-packet encryption key…
- … by preventing weak-key attacks and decorrelating WEP IV and per-packet key
- must be efficient on existing hardware



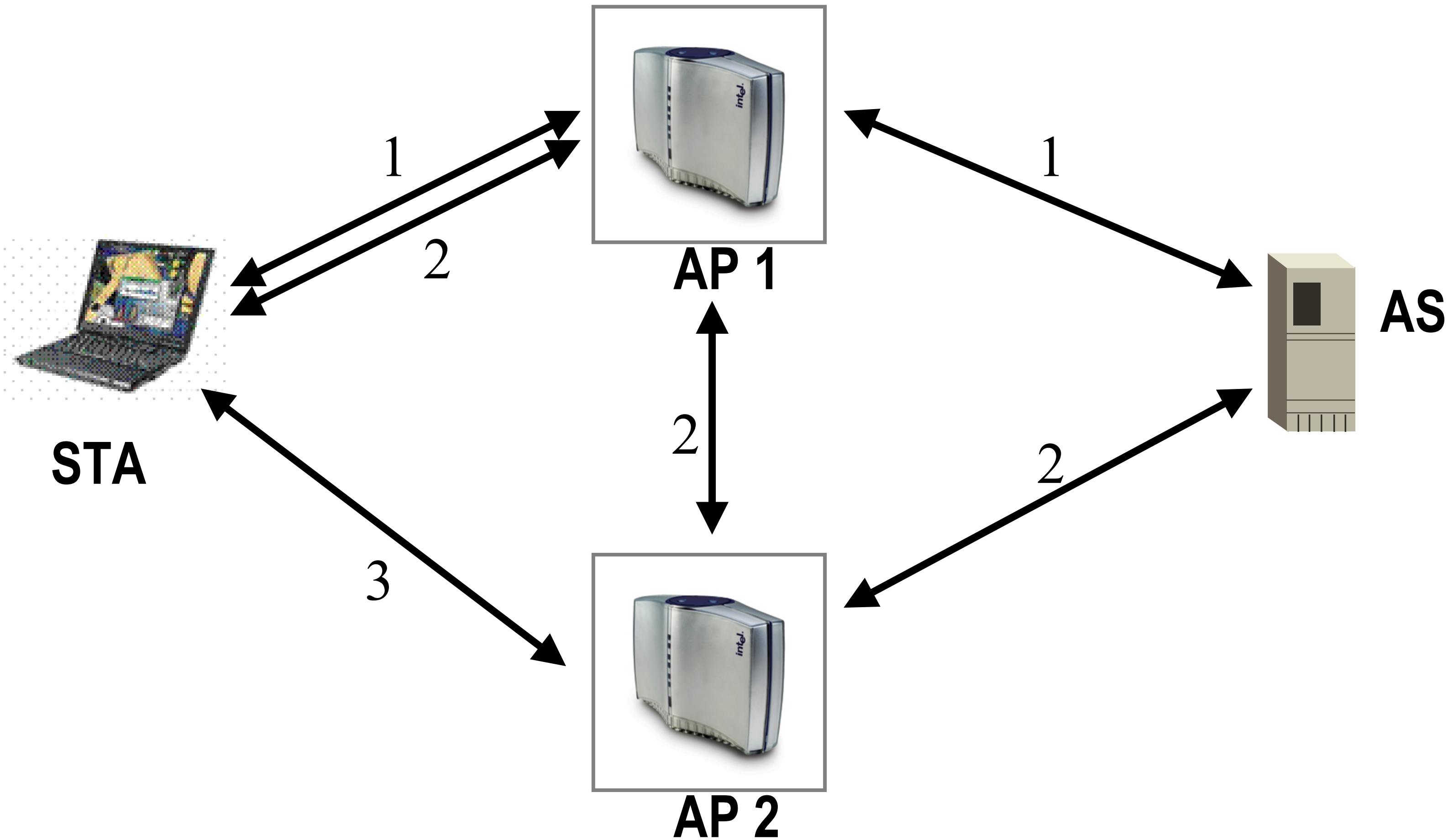
Summary

	WEP	TKIP	CCMP
Cipher	RC4	RC4	AES
Key Size	40 or 104 bits	128 bits encryption, 64 bit auth	128 bits
Key Life	24-bit IV, wrap	48-bit IV	48-bit IV
Packet Key	Concat.	Mixing Fnc	Not Needed
Integrity			
Data	CRC-32	Michael	CCM
Header	None	Michael	CCM
Replay	None	Use IV	Use IV
Key Mgmt.	None	EAP-based	EAP-based

Other 802.11i Features

- Pre-authentication and roaming
- PEAP and legacy authentication support
- Pre-shared key without authentication
- Ad hoc networks
- Password-to-Key mapping
- Random number generation

Pre-authentication (1)

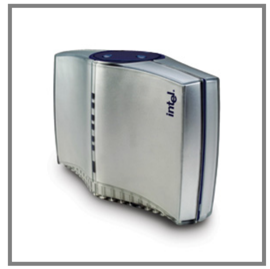


Pre-authentication (2)

- While not part of 802.11i, TLS-resume can expedite re-authentication
  - New MK $\leftarrow$ TLS-PRF(Old MK, clientHello.random | serverHello.random)
  - Optimizes away expensive public key operations
- Consequences of TLS-resume not studied in the context of architecture's weak binding

PEAP Overview

Wireless Station



AP

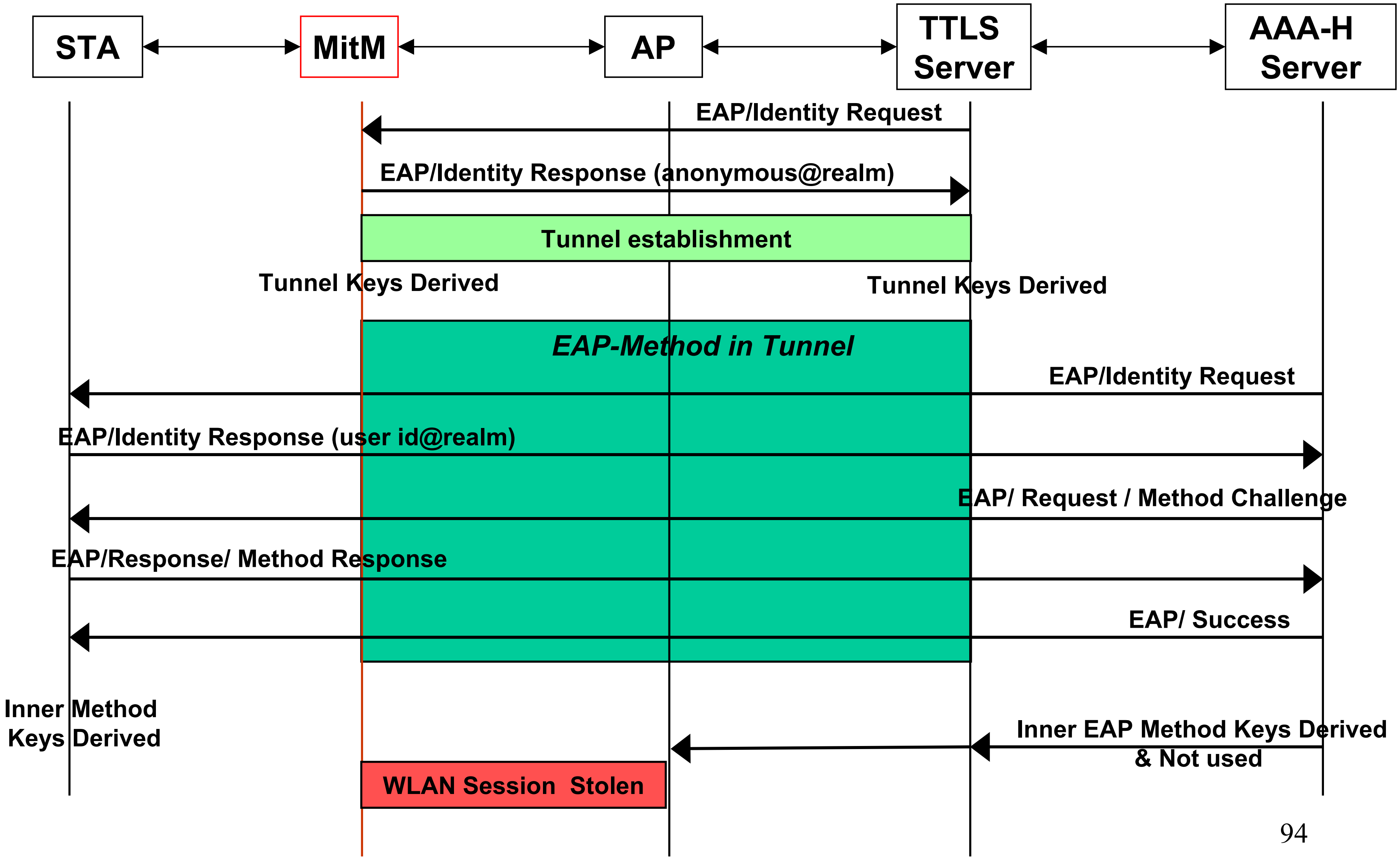
Authentication Server

Step 1: Use EAP-TLS to authenticate AS to Station

Step 2: Use TLS key to protect the channel between Station, AS

Step 3: Use Legacy method protected by TLS key to authenticate Station to AS

PEAP Man-in-Middle Attack

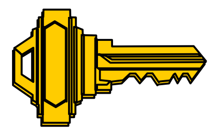


Fixing (?) PEAP

- Compound Keys
  - Use PRF to combine "inner" and "outer" keys when both are available
- "Extra" mutual authentication round
  - Protect "extra" exchange with combined key
- Distribute "combined" key as the PMK protecting the session
- Problem: legacy credentials can still be exposed if credentials reused without PEAP?

Pre-shared Key (1)

Wireless Station



PSK, used directly as a PMK

AP

802.11 security capabilities discovery

Enhanced 802.1X key mgmt (no authentication)

CCMP, WRAP, or TKIP

Pre-shared Key (2)

- No explicit authentication!
  - The entire 802.1X authentication exchange elided
- Can have a single pre-shared key for entire network (insecure)…
- …or one per STA pair (secure)
- PSK motive:
  - Ad hoc networks
  - Home networks

Ad hoc networks

- Configure a network-wide pre-shared key and SSID
- Each STA in ad hoc network initiates 4-way handshake based on PSK when
  - It receives following from a STA with whom it hasn't established communication
  - Beacons with same SSID
  - Probe Requests with same SSID
- Each STA distributes its own Group Key to each of the other STAs in ad hoc network

Password-to-Key Mapping

- Uses PKCS #5 v2.0 PBKDF2 to generate a 256-bit PSK from an ASCII password
  - PSK = PBKDF2 (*Password*, *ssid, ssidlength*, *4096*, 256)
  - Salt = SSID, so PSK different for different SSIDs
- Motive: Home users might configure passwords, but will never configure keys
  - Is something better than nothing?

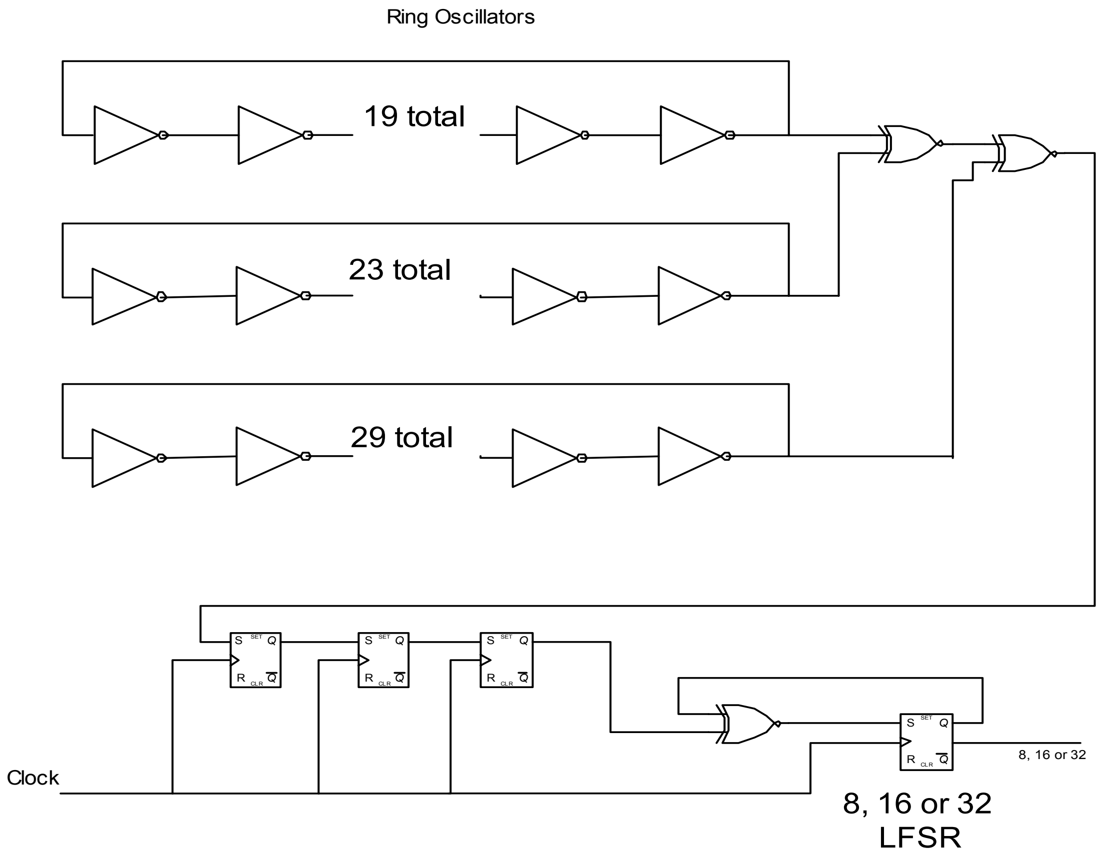
Randomness Needed

- All systems implementing crypto need cryptographic quality pseudo-random numbers
- Therefore, 802.11 supplies implementation guidelines for minimal quality generators
- Suggests two techniques:
  - Software-based sampling
  - Hardware-based sampling

Software Based Sampling

```
result ← empty
LoopCounter ← 0
Wait until network traffic
Repeat until global key counter "random enough" or 32 times {
      result ← EAPoL-PRF(“”, "Init Counter", Local Mac Address | Time | result | LoopCounter)
      LoopCounter ← LoopCounter + 1
      Repeat 32 times {
            If Ethernet traffic available then
                        result ← result | lowest byte of time of Ethernet packet
            else
                        Initiate 4-way handshake, but but break off after Message 2
                        result ← result | lowest byte of time when Message 1 sent
                                    | lowest byte of time when Message 2 received
                                    | lowest byte of Received Signal String Indicator when
                                                Message 2 received
                                    SNonce from Message 2
      }
}
result ← EAPoL-PRF (“”, "Init Counter", Local Mac Address | Time | result | LoopCounter | 256)
```

Hardware Assisted Sampling



Driver for Hardware Assist

```
Initialize result to empty array
Repeat 1024 times {
    Read LFSR
    result = result | LFSR
    Wait a time period
}
Global key counter = PRF-256(0, "Init Counter", result)
```

802.11i Summary

- New 802.11i data protocols provide confidentiality, data origin authenticity, replay protection
- These protocols require fresh key on every session
- Key management delivers keys used as authorization tokens, proving channel access is authorized
- Architecture ties keys to authentication

Feedback?