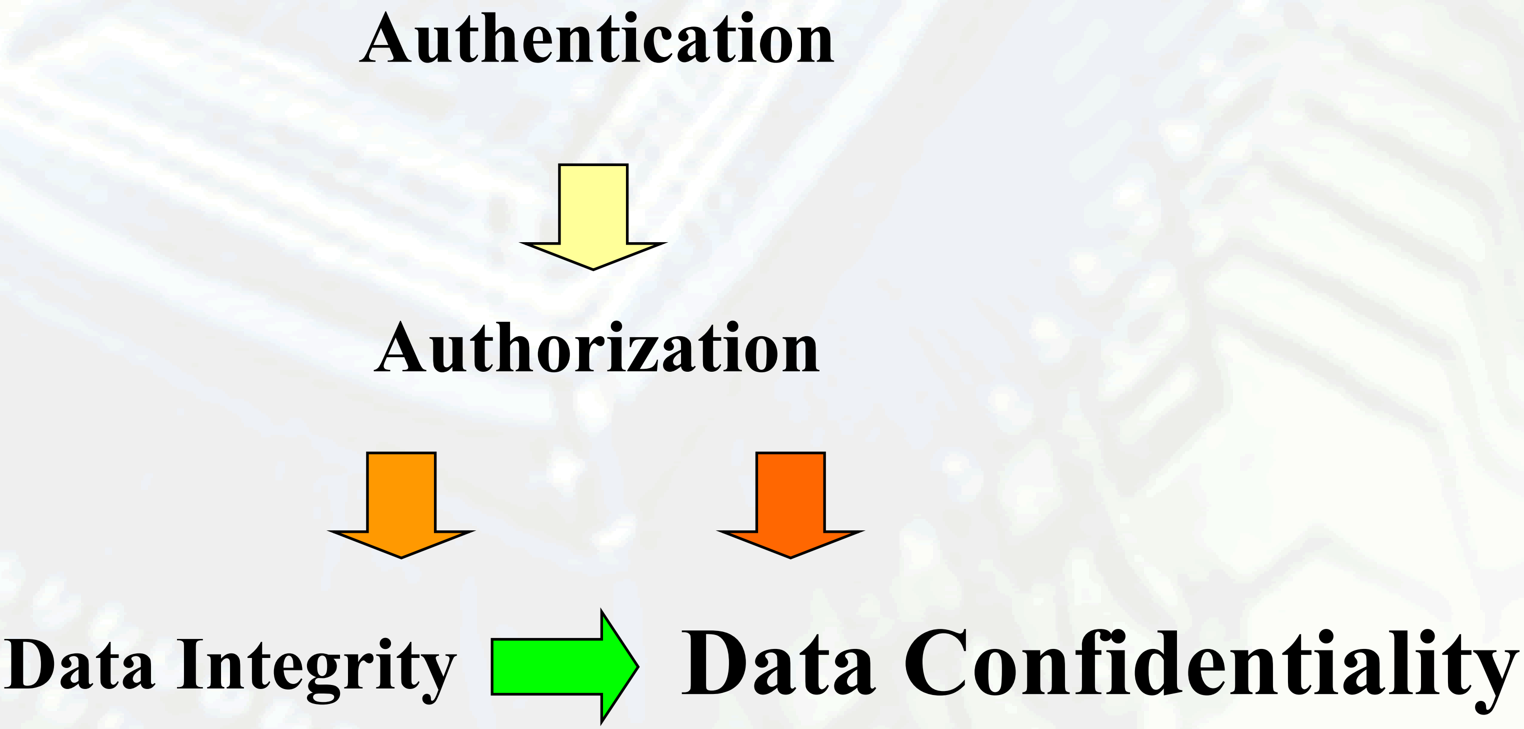
Where does WiFi Security Come From?

Jesse Walker
Intel Corporation
jesse.walker@intel.com

Agenda

- The Chain of Trust
- How 802.11i Delivers

The Chain of Trust



Authentication

Authorization

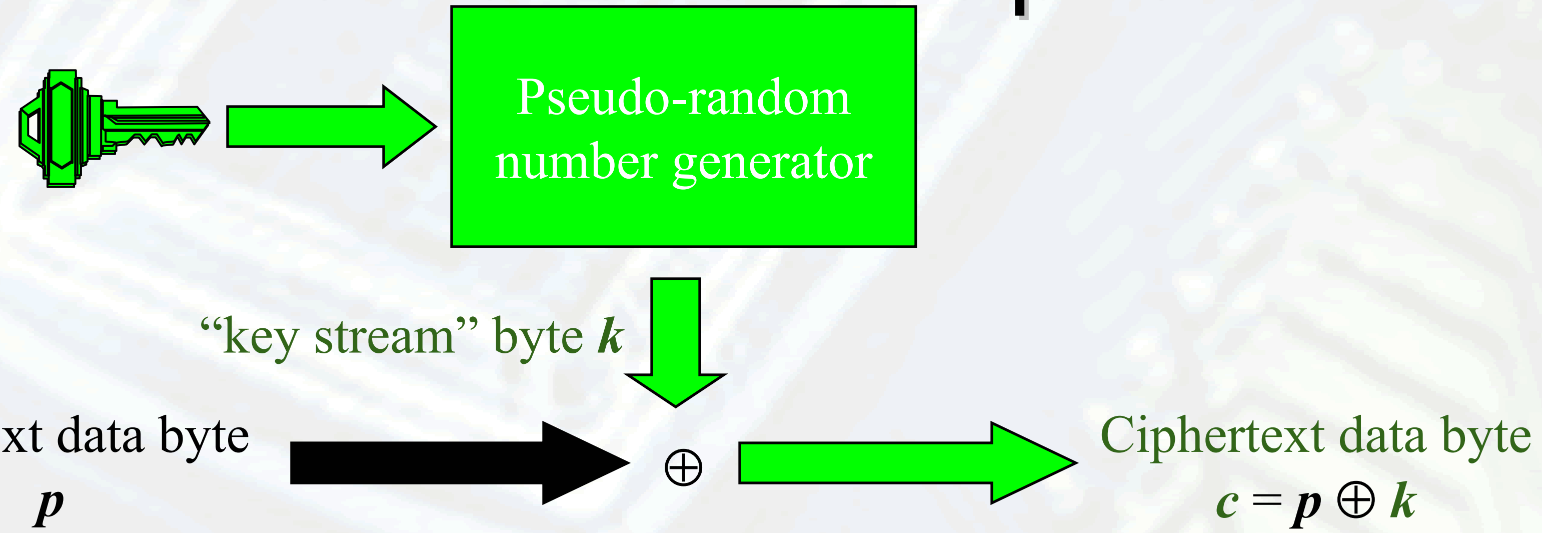
Data Integrity

Data Confidentiality

Data Confidentiality

- Purpose: Control read access of the channel
  - So an attacker cannot steal your data
- How:
  - Use a cryptographic key to
    - encrypt every packet sent over the channel
    - decrypt every packet received over the channel
  - Discard all unencrypted packets
  - Key use proves authorization to access channel
- Questions:
  - Required qualities of the cryptographic key?
  - How do you know the decrypted data is any good?

Review of stream ciphers



Pseudo-random number generator

“key stream” byte $\boldsymbol{k}$

Plaintext data byte $\boldsymbol{p}$

$\oplus$

Ciphertext data byte $\boldsymbol{c} = \boldsymbol{p} \oplus \boldsymbol{k}$

Decryption works the same way: $\boldsymbol{p} = \boldsymbol{c} \oplus \boldsymbol{k}$

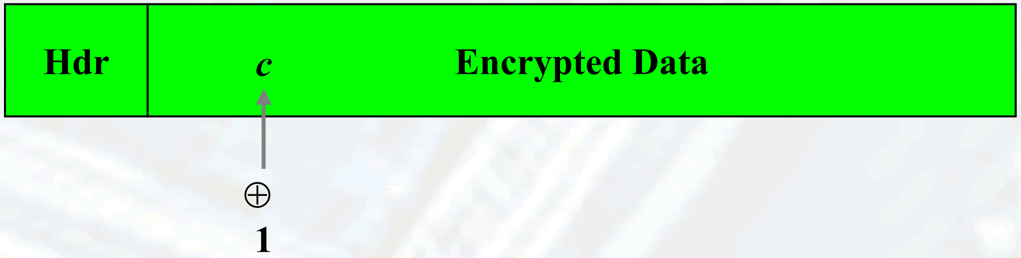
Thought Experiment: what happens if you encrypt two different plaintexts under the same key stream by $\boldsymbol{k}$?

$$\boldsymbol{c}_1 = \boldsymbol{p}_1 \oplus \boldsymbol{k} \qquad \boldsymbol{c}_2 = \boldsymbol{p}_2 \oplus \boldsymbol{k}$$

$$\boldsymbol{c}_1 \oplus \boldsymbol{c}_2 = \boldsymbol{p}_1 \oplus \boldsymbol{k} \oplus \boldsymbol{p}_2 \oplus \boldsymbol{k} = \boldsymbol{p}_1 \oplus \boldsymbol{p}_2$$

Conclusion: can’t reuse the key stream byte $\boldsymbol{k}$ across different packets; need a ***fresh*** key ***every*** time stream cipher is reinitialized

Forgery attacks

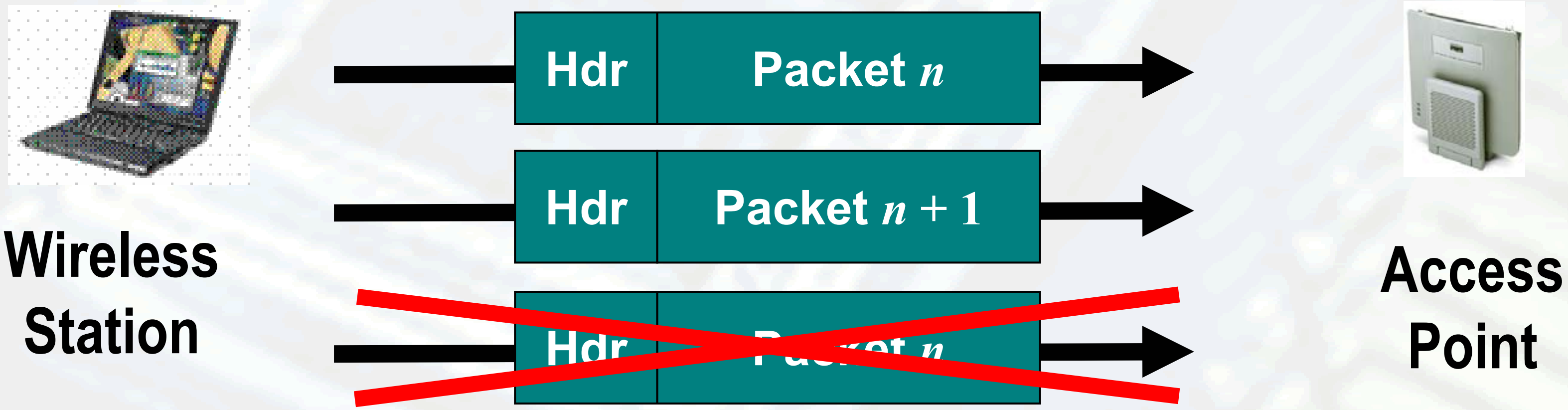


- Defeating stream ciphers:
  - □ Capture an in-flight encrypted packet
  - □ Pick any byte *c* of ciphertext data and flip one of its bits. Then we know $c = p \oplus k$ for some key stream byte ***k***
  - □ Release captured altered packet
  - □ On decryption, since $p' = c \oplus k$, the byte with bit flipped will decrypt as $p' = p \oplus 1$
- Encryption only provides confidentiality, not integrity!

Data Integrity

- Purpose: Control write access to the channel so attacker cannot
  - ◇ impersonate you to the network
  - ◇ impersonate the network to you
  - ◇ use the network to decrypt your data
- How:
  - ◇ Give each packet a sequence number
  - ◇ Use a cryptographic key deriving from authentication to“sign” every packet, including sequence number
  - ◇ Use a cryptographic key to verify every “signed” packet received over the channel
  - ◇ Discard all packets with invalid “signatures”
  - ◇ Discard all unsigned packets
  - ◇ Discard all packets received out of order (wrong seq #)
  - ◇ Key use proves authorization to access channel
- Questions:
  - ◇ Required qualities of the integrity key?

Data Integrity's Achilles Heel



Thought Experiment: what happens if you reuse sequence numbers with the same cryptographic key?

Answer: Attacker can replay packets

Conclusion: Need a ***fresh*** data integrity key each time the packet sequence space is restarted

Authorization

- Purpose: to make a decision
  - ◇ Decide whether you want to talk with the network
  - ◇ The network will do the same with you
- How:
  - ◇ Look for the network on list of approved networks
  - ◇ Network will look for you among list of authorized users/devices
  - ◇ Device and network agree on fresh cryptographic keys
  - ◇ Device and network use agreed-upon cryptographic key to enforce access on each subsequent packets
- Questions:
  - ◇ How do you know the peer is really the authorized party?

Authentication

- Discover the peer’s identity
  - ◇The network proves who it is to you, so you can decide if you *really* do want to talk with it
  - ◇You (or your device) proves who it is to the network can decide whether to talk with you
- How:
  - ◇Authentication based on credentials exchange
- Questions:
  - ◇Where do the credentials used in the exchange come from?
  - ◇What properties are required of the exchange?

Observations

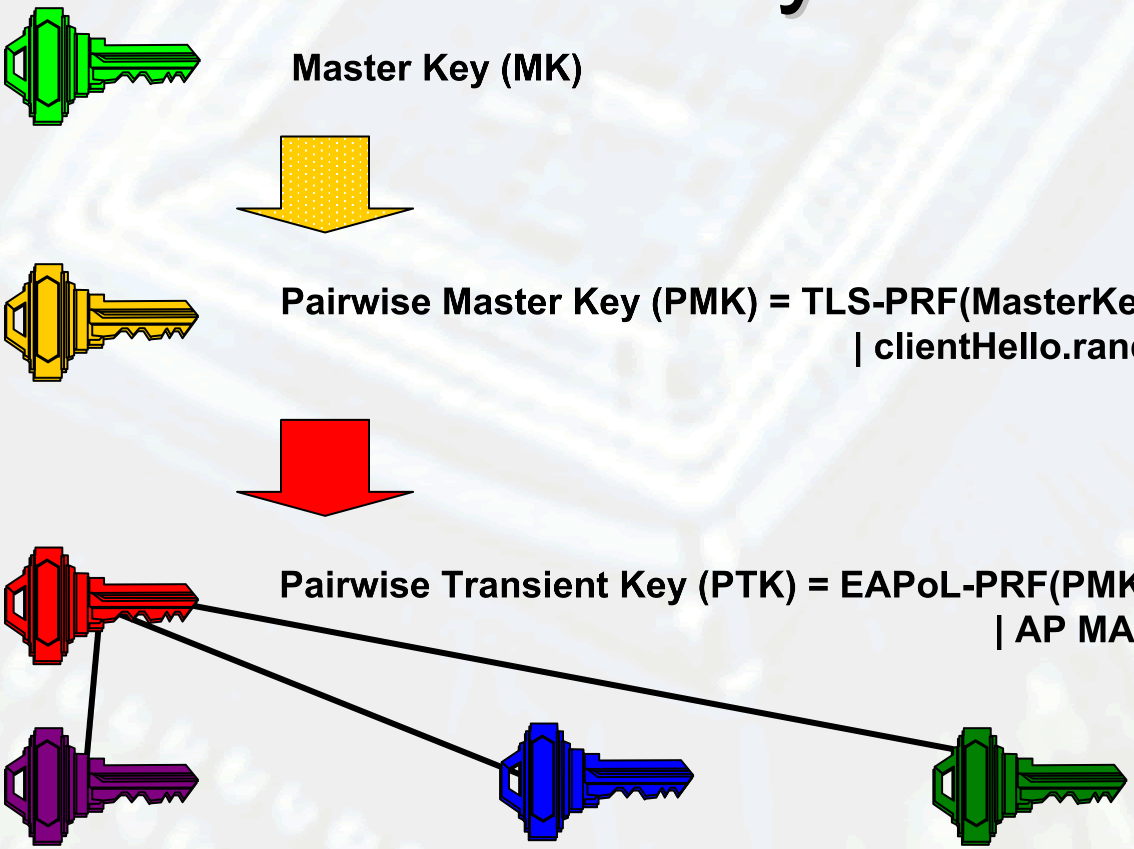
- Can’t obtain confidentiality without integrity
  - An attacker can use the infrastructure to help break the encryption key
- Can’t obtain confidentiality without authorization
  - Need to create a fresh data encryption key to to limit read access to the data
- Can’t obtain integrity without authorization
  - Need to create a fresh data integrity key to prove traffic is authorized
- Can’t obtain authorization without authentication
  - How do you know if they are allowed if you don’t know who they are?

No Security without all the links in the chain

What is…?

- TGi – IEEE working group tasked to "fix" WiFi security
- 802.11i – Standard that will be produced by TGi
- WPA – WiFi Protected Access; pre-standard subset of 802.11i
    - Includes TKIP, to "replace" WEP
    - Includes 802.11i key management
    - Includes 802.1X authentication

802.11i Key Hierarchy



Master Key (MK)

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

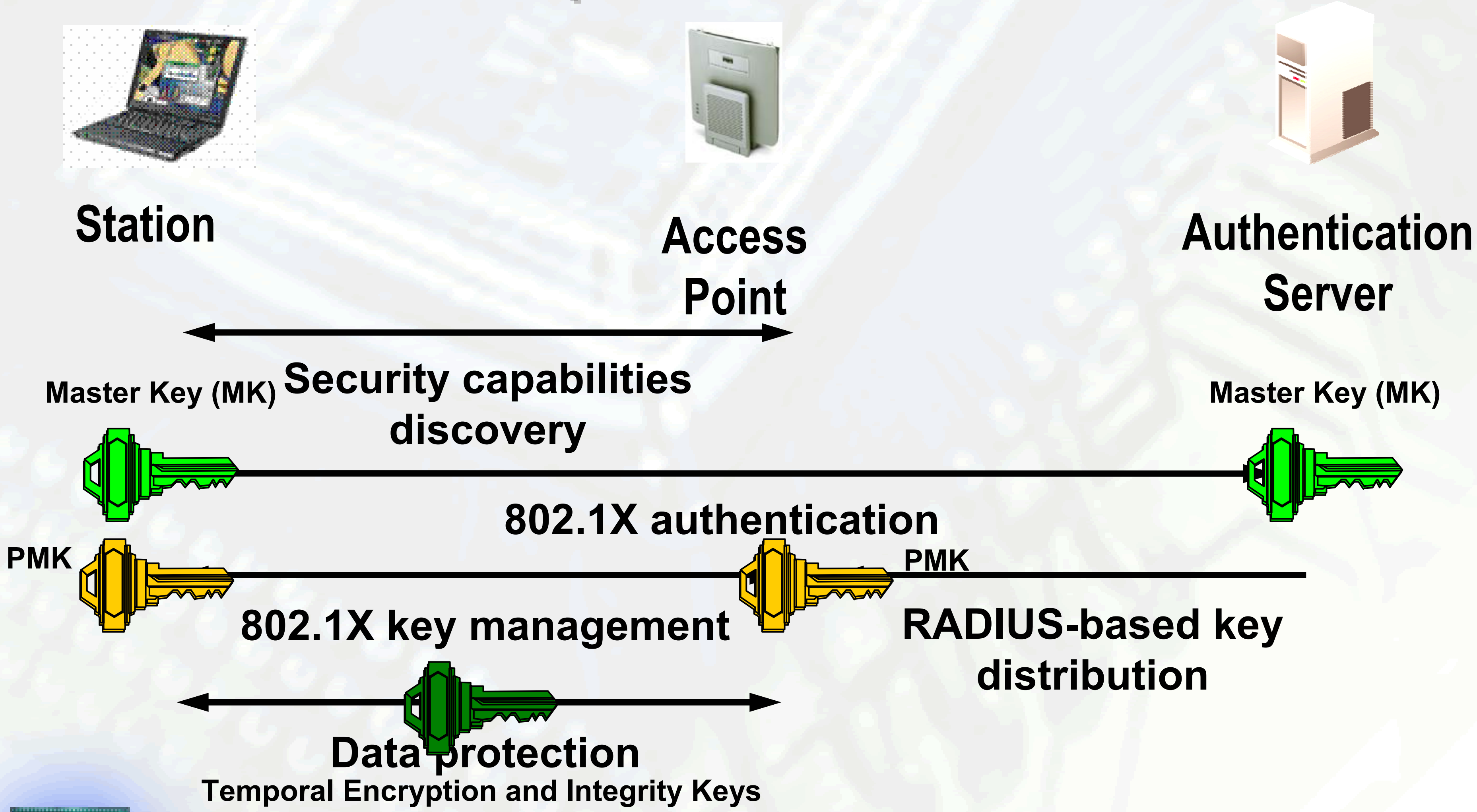
Pairwise Transient Key (PTK) = EAPoL-PRF(PMK, AP Nonce | STA Nonce | AP MAC Addr | STA MAC Addr)

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

802.11 Operational Phases



Station

Access Point

Authentication Server

Master Key (MK)

Security capabilities discovery

Master Key (MK)

802.1X authentication

PMK

PMK

802.1X key management

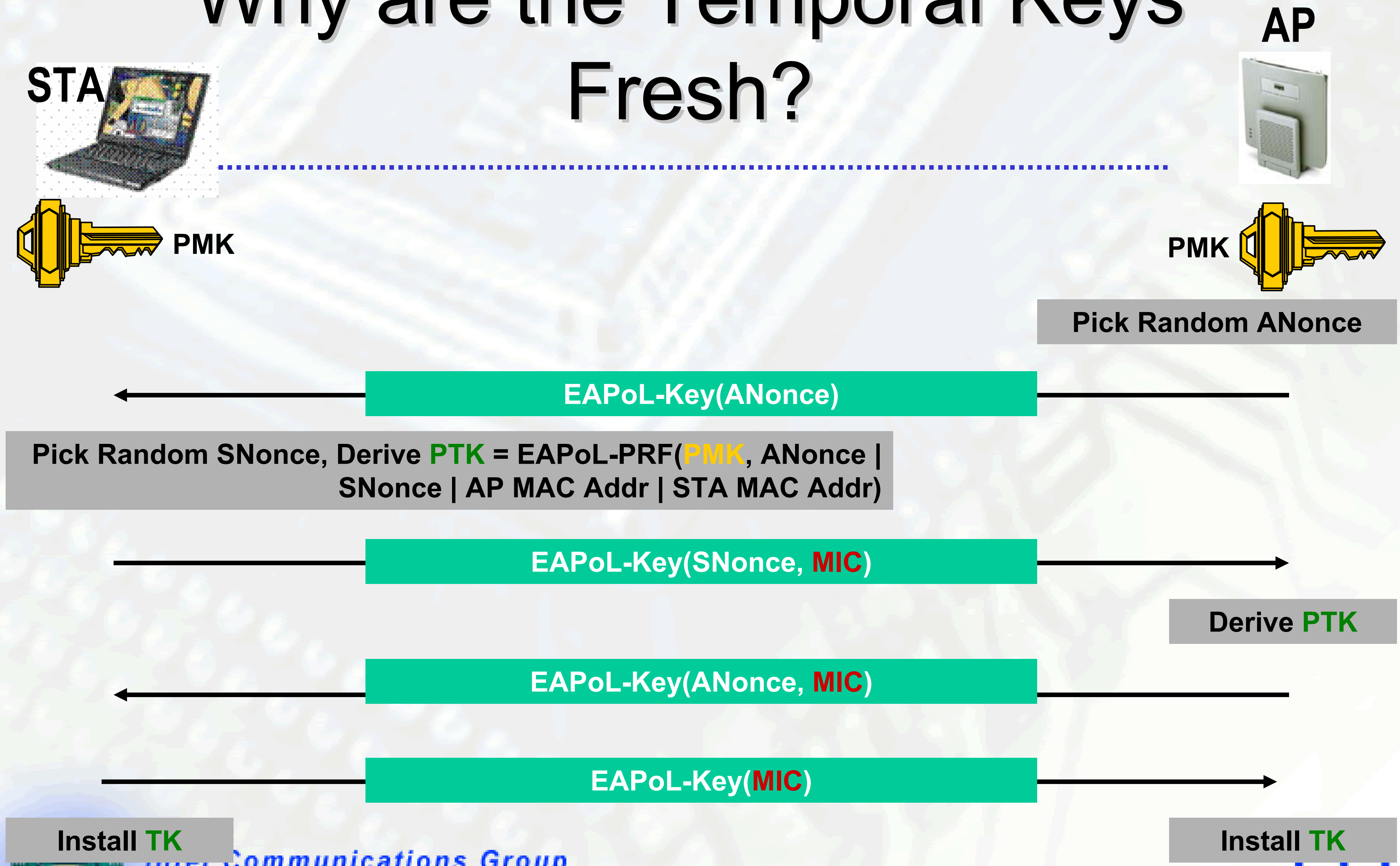
RADIUS-based key distribution

Data protection

Temporal Encryption and Integrity Keys

How 802.11i Delivers

Why are the Temporal Keys Fresh?



STA

AP

PMK

PMK

Pick Random ANonce

EAPoL-Key(ANonce)

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

EAPoL-Key(SNonce, MIC)

Derive PTK

EAPoL-Key(ANonce, MIC)

EAPoL-Key(MIC)

Install TK

Install TK

How does 802.11i provide…?

Seq #	Data	MIC
48 bits	>=0 octets	8 octets

802.11i fixes WiFi protocol security

Summary

- No Security without all the links in the chain
- 802.11i fixes WiFi protocol security

Feedback?