Overview of 802.11 Security

Jesse Walker, Intel Corporation
(503) 712-1849
jesse.walker@intel.com

Acknowledgements

- Bernard Aboba
- Dave Halasz
- Nikita Borisov
- Duncan Kitchin

I am grateful to these individuals for allowing me to freely plagiarize their presentations and documents to prepare this presentation

Agenda

- Introduction and Goals
- 802.11 Security Today
- What's Wrong Today?
- Proposed Encapsulation
- Proposed Authentication, Authorization, and Key Management
- Summary

Introduction

- 802.11 security recently gained notoriety:
  - Celebrated UC Berkeley paper by Borisov *et. al*. publicized attacks on 802.11
  - Most of these were discussed privately in 802.11 TGe throughout last year and documented in doc. IEEE 802.11/00-362
  - UC Berkeley group wasn't participating in the process, but 802.11 was living in sin

Goals

- Develop a wider understanding of network security requirements
- Explain how 802.11 security works now
- Describe its major issues
- Understand what is being done to address the issues

Agenda

- Introduction and Goals
- 802.11 Security Today
- What's Wrong Today?
- Proposed Encapsulation
- Proposed Authentication, Authorization, and Key Management
- Summary

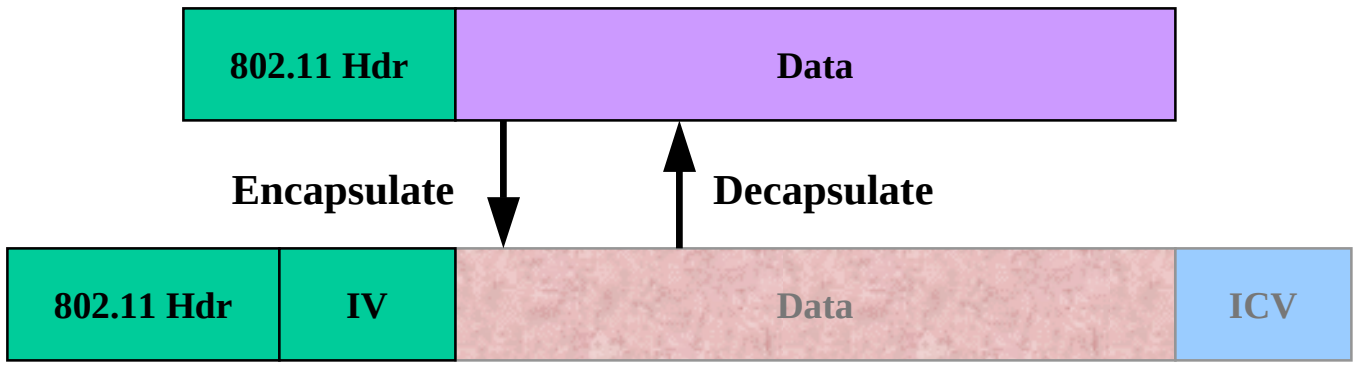
802.11 Security Today

- Goals of existing 802.11 security
- Existing security consists of two subsystems
  - A data encapsulation technique called ***W***ired ***E***quivalent ***P***rivacy (WEP)
  - An authentication algorithm called Shared Key Authentication

Existing 802.11 Security Goals

- Create the privacy achieved by a wired network
  - Only prevent intellectual property from leaking through casual browsing
- Simulate physical access control by denying access to unauthenticated stations

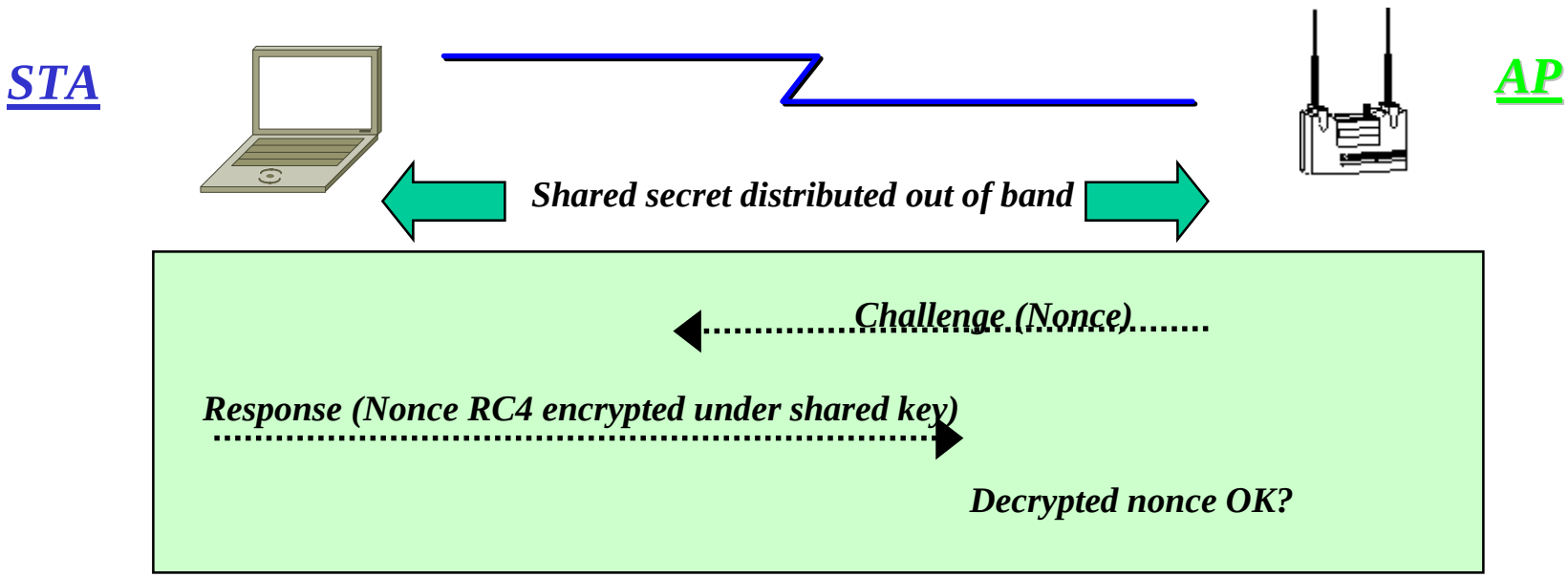
WEP Encapsulation



WEP Encapsulation Summary:

- Encryption Algorithm = RC4
- Per-packet encryption key = 24-bit IV concatenated to a pre-shared key
- WEP allows IV to be reused with any frame
- Data integrity provided by CRC-32 of the plaintext data (the “ICV”)
- Data and ICV are encrypted under the per-packet encryption key

WEP Authentication



802.11 Authentication Summary:

- Authentication key distributed out-of-band
- Access Point generates a “randomly generated” challenge
- Station encrypts challenge using pre-shared secret

Agenda

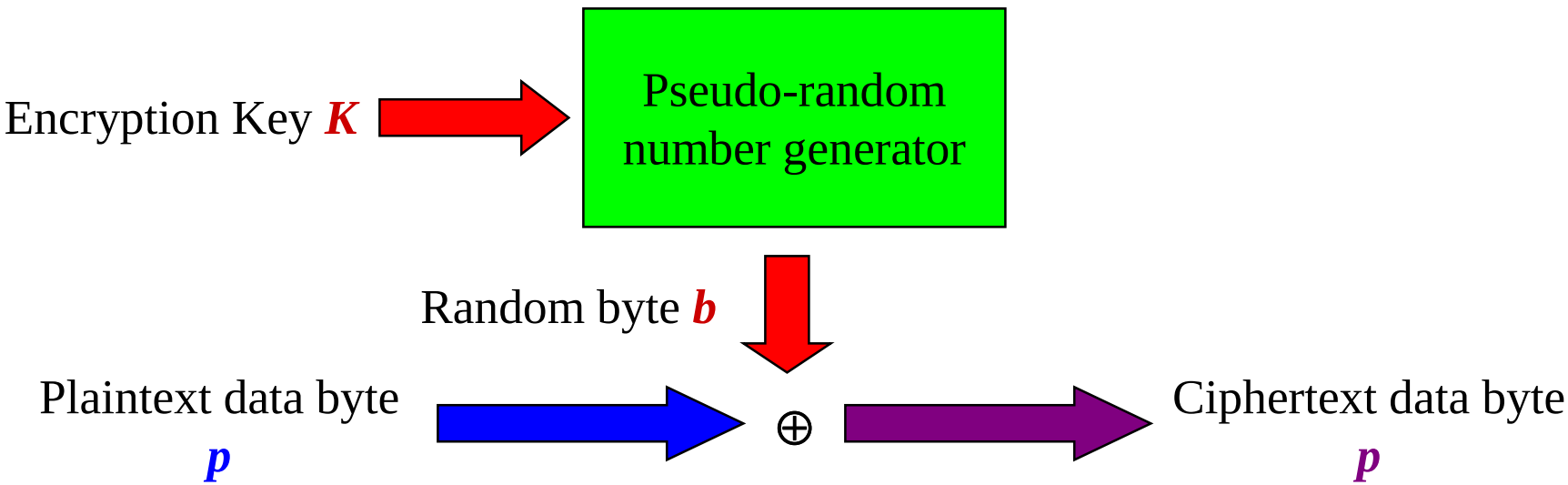
- Introduction and Goals
- 802.11 Security Today
- So What's Wrong Today?
- Proposed Encapsulation
- Proposed Authentication, Authorization, and Key Management
- Summary

So What's Wrong Today?

- Properties of Vernam Ciphers
- How to read WEP Encrypted Traffic
- How to authentication without the key
- Traffic modification
- Lessons
- Requirements for a networked data encapsulation scheme

Properties of Vernam Ciphers (1)

The WEP encryption algorithm RC4 is a Vernam Cipher:



Decryption works the same way: $p = c \oplus b$

Properties of Vernam Ciphers (2)

Thought experiment 1: what happens when p_1 and p_2 are encrypted under the same "random" byte b?

$$c_1 = p_1 \oplus b \qquad c_2 = p_2 \oplus b$$

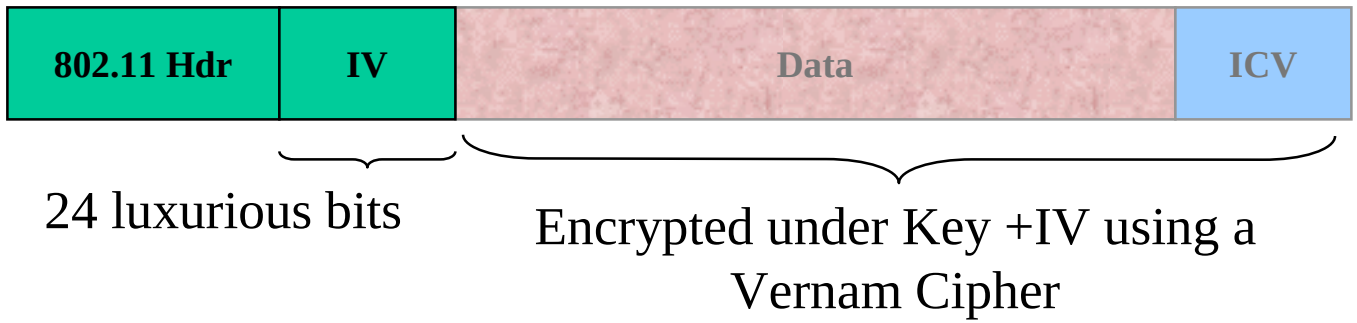
Then:

$$c_1 \oplus c_2 = (p_1 \oplus b) \oplus (p_2 \oplus b) = p_1 \oplus p_2$$

Conclusion: it is a very bad idea to encrypt any two bytes of data using the same byte output by a Vernam Cipher PRNG.

Ever.

How to Read WEP Encrypted Traffic (1)

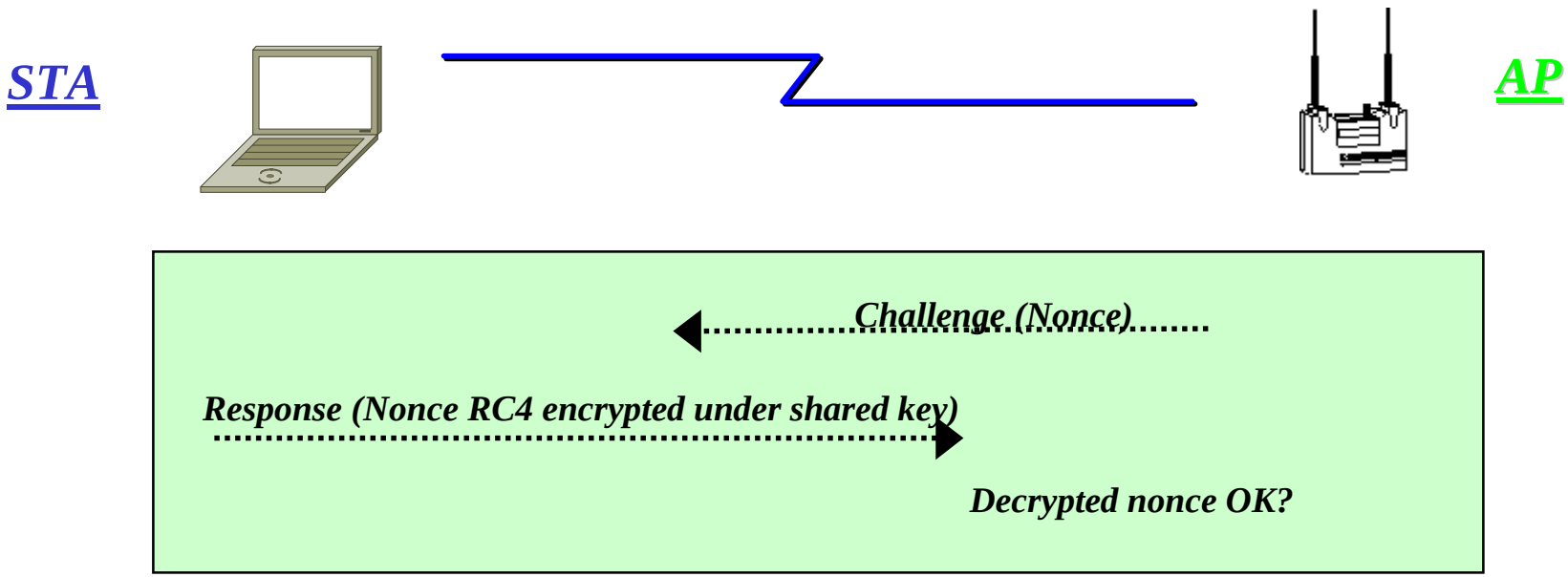


- By the Birthday Paradox, probability P_n two packets will share same IV after n packets is $P_2 = 1/2^{24}$ after two frames and $P_n = P_{n-1} + (n-1)(1-P_{n-1})/2^{24}$ for $n > 2$.
- 50% chance of a collision exists already after only 4823 packets!!!
- Pattern recognition can disentangle the XOR'd recovered plaintext.
- Recovered ICV can tell you when you've disentangled plaintext correctly.
- After only a few hours of observation, you can recover all 2^{24} key streams.

How to Read WEP Encrypted Traffic (2)

- Ways to accelerate the process:
  - Send spam into the network: no pattern recognition required!
  - Get the victim to send e-mail to you
    - The AP creates the plaintext for you!
  - Decrypt packets from one Station to another via an Access Point
    - If you know the plaintext on one leg of the journey, you can recover the key stream immediately on the other
  - Etc., etc., etc.

How to Authenticate without the Key



With our background, an easy attack is obvious:

- Record one challenge/response with a sniffer
- Use the challenge to decrypt the response and recover the key stream
- Use the recovered key stream to encrypt any subsequent challenge

Traffic Modification (1)

Vernam cipher thought experiment 2: how hard is it to change a genuine packet's data, so ICV won't detect the change?

Answer: Easy as pie

Represent an n-bit plaintext as an n-th degree polynomial:

$$\boldsymbol{p} = p_n x^n + p_{n-1} x^{n-1} + \ldots + p_0 x^0 \qquad (\text{each } p_i = 0 \text{ or } 1)$$

Then the plaintext with ICV can be represented as :

$$\boldsymbol{p}x^{32} + \mathrm{ICV}(\boldsymbol{p}) = \boldsymbol{p}_n x^{n+32} + \boldsymbol{p}_{n-1} x^{n-31} + \ldots + \boldsymbol{p}_0 x^{32} + \mathrm{ICV}(\boldsymbol{p})$$

If the n+32 bit RC4 key stream used to encrypt the body is represented by the n+32nd degree polynomial ***b***, then the encrypted message body is

$$\boldsymbol{p}x^{32} + \mathrm{ICV}(\boldsymbol{p}) + \boldsymbol{b}$$

Traffic Modification (2)

But the ICV is linear, meaning for any polynomials *p* and *q*

$$\mathbf{IVC(}p\mathbf{+}q\mathbf{)} = \mathbf{ICV(}p\mathbf{)} + \mathbf{ICV(}q\mathbf{)}$$

This means that if q is an arbitrary nth degree polynomial, i.e., an arbitrary change in the underlying message data:

$$\mathbf{(}p\mathbf{+}q\mathbf{)}x^{32} + \mathrm{ICV}\mathbf{(}p\mathbf{+}q\mathbf{)} + \mathbf{b} = \mathbf{p}x^{32} + \mathbf{q}x^{32} + \mathrm{ICV}\mathbf{(}p\mathbf{)} + \mathrm{ICV}\mathbf{(}q\mathbf{)} + \mathbf{b}$$

$$= \mathbf{((}px^{32} + \mathrm{ICV}\mathbf{(}p\mathbf{))} + \mathbf{b)} + \mathbf{(}qx^{32} + \mathrm{ICV}\mathbf{(}q\mathbf{))}$$

Conclusion: Anyone can alter an WEP encapsulated packet in arbitrary ways without detection!!

Lessons

- Data encryption by itself offers no protection from attack
  - "*It's access control, stupid*"
  - *there is no meaningful privacy if the data authenticity problem is not solved*
- It is profoundly easy to mis-use a cipher
  - "*don't try this at home*"
  - Get any cryptographic scheme reviewed by professionals

Requirements for a networked data encapsulation scheme

- Data traffic:
  - packets must be authentication, not just encrypted
  - packets must have sequence numbers to prevent replay
- Authentication:
  - Mutual authentication required
  - Per-packet keys have to be tied to the authentication

Agenda

- Introduction and Goals
- 802.11 Security Today
- What's Wrong Today?
- Proposed Encapsulation
- Proposed Authentication, Authorization, and Key Management
- Summary

Encapsulation Proposal

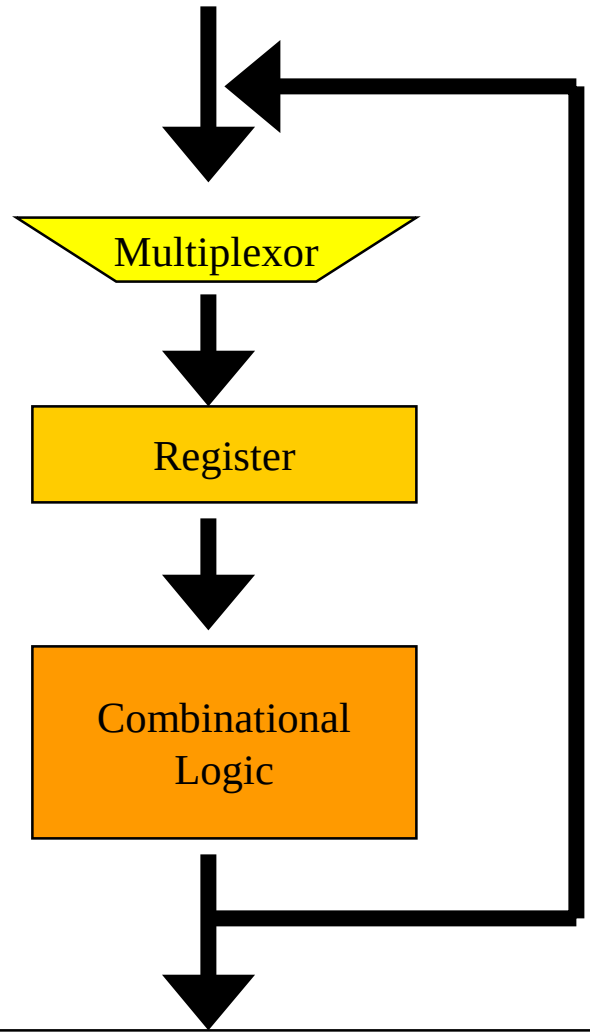
Current TGe plan of record:

- Use AES-128 as the new cryptographic primitive
- Use AES in Offset Codebook Mode OCB mode
  - Phil Rogaway submission to AES modes of operation
  - algorithm provides both privacy and data integrity
- Add session sequence number to avoid replay
- Map base key to session key
  - use OCB mode tag to compute session key, to minimize number of cryptographic primitives implemented

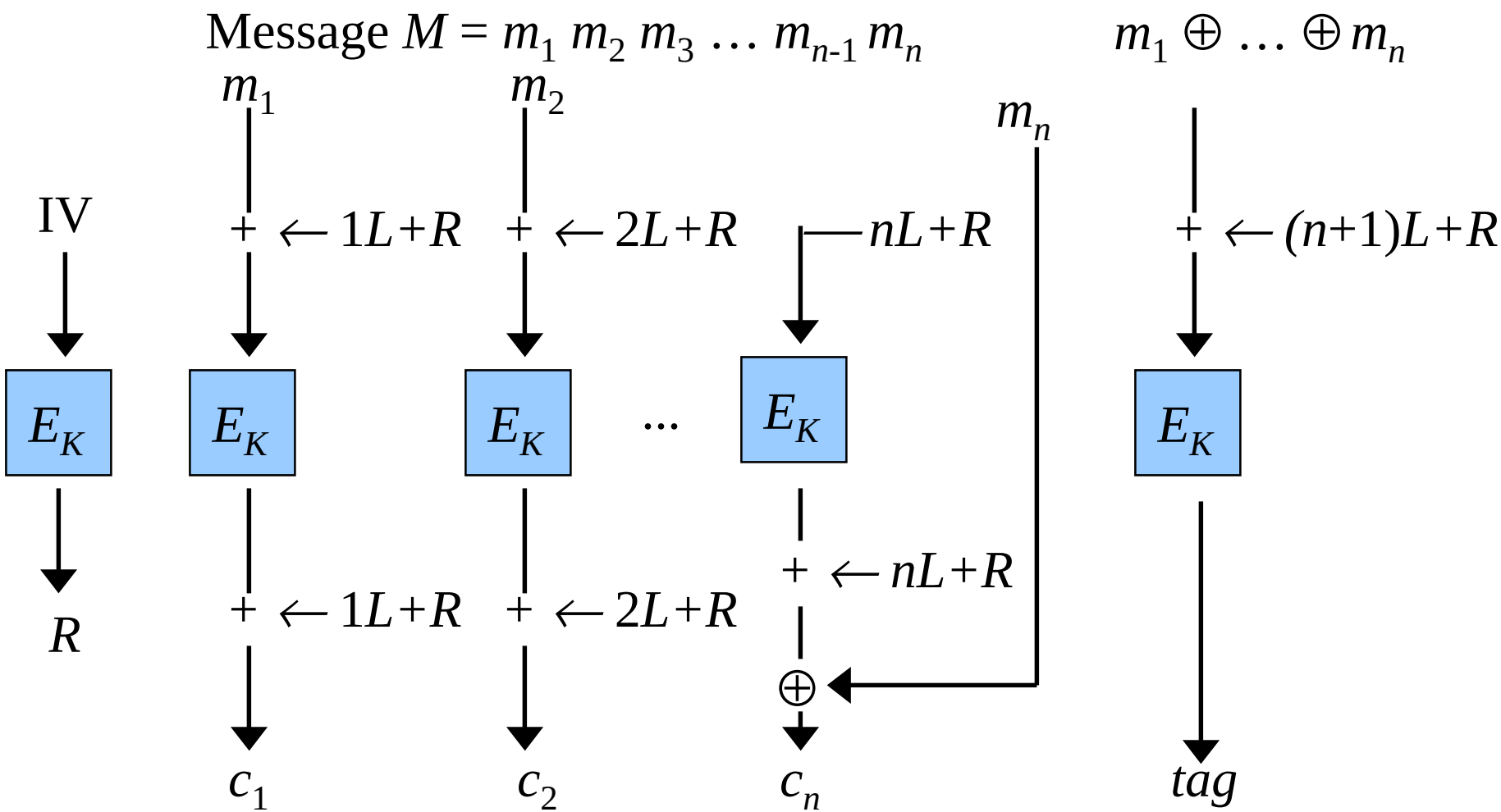
AES Facts

- **NIST standard for iterated block ciphers**
  - **Block cipher: encrypts or decrypts 128-bit blocks of data, not individual bytes**
  - **Uses 128-, 192-, or 256-bit keys**
  - **Highly parallelizable**
- **Critical path instructions:**
  - **8×8 S-box, XOR6, XOR5, XOR2, MUX2**
- **Performance**
  - **200 MHz Pentium Pro: 284 cycles/block**

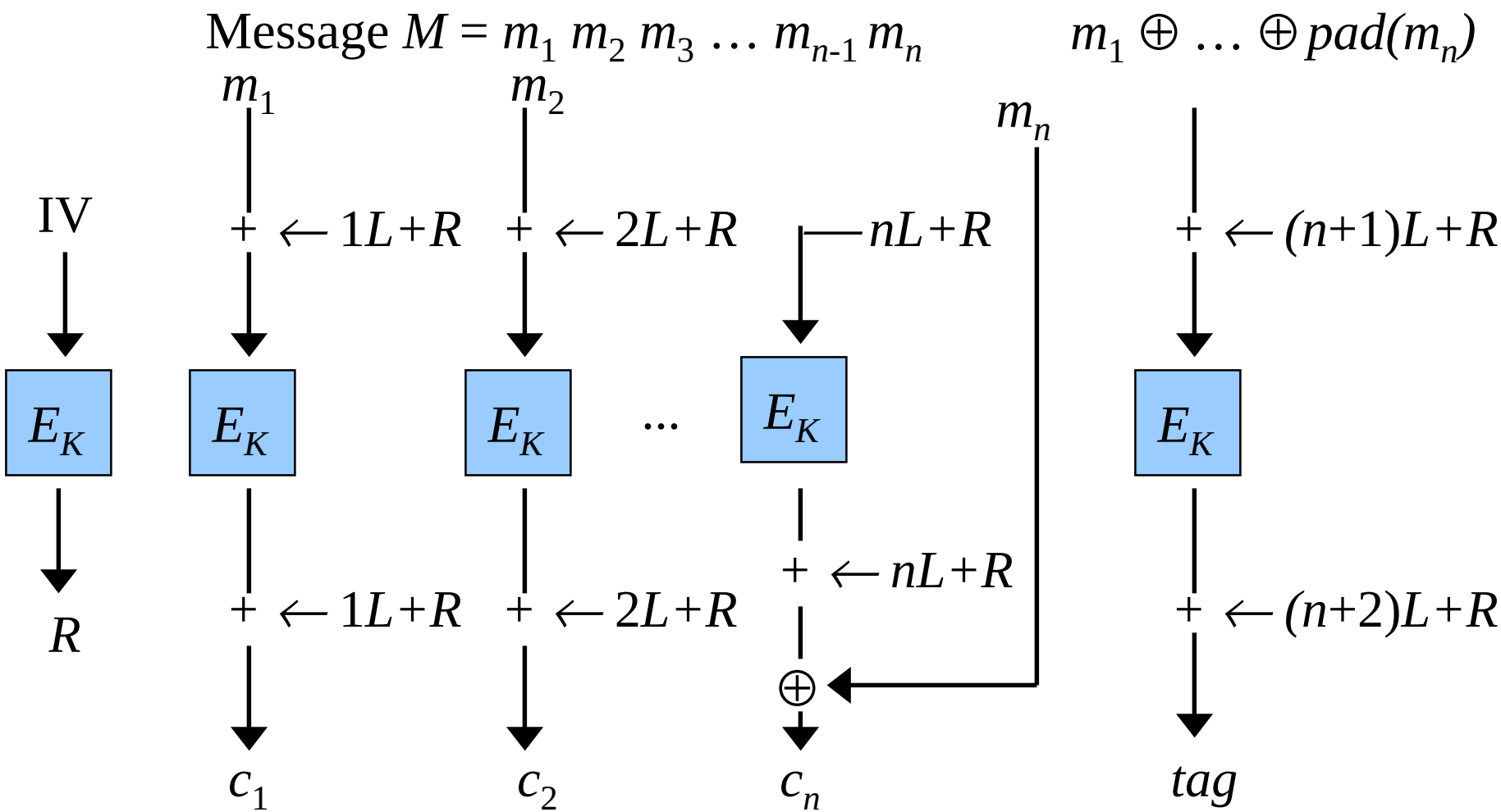
Iterated Block Ciphers



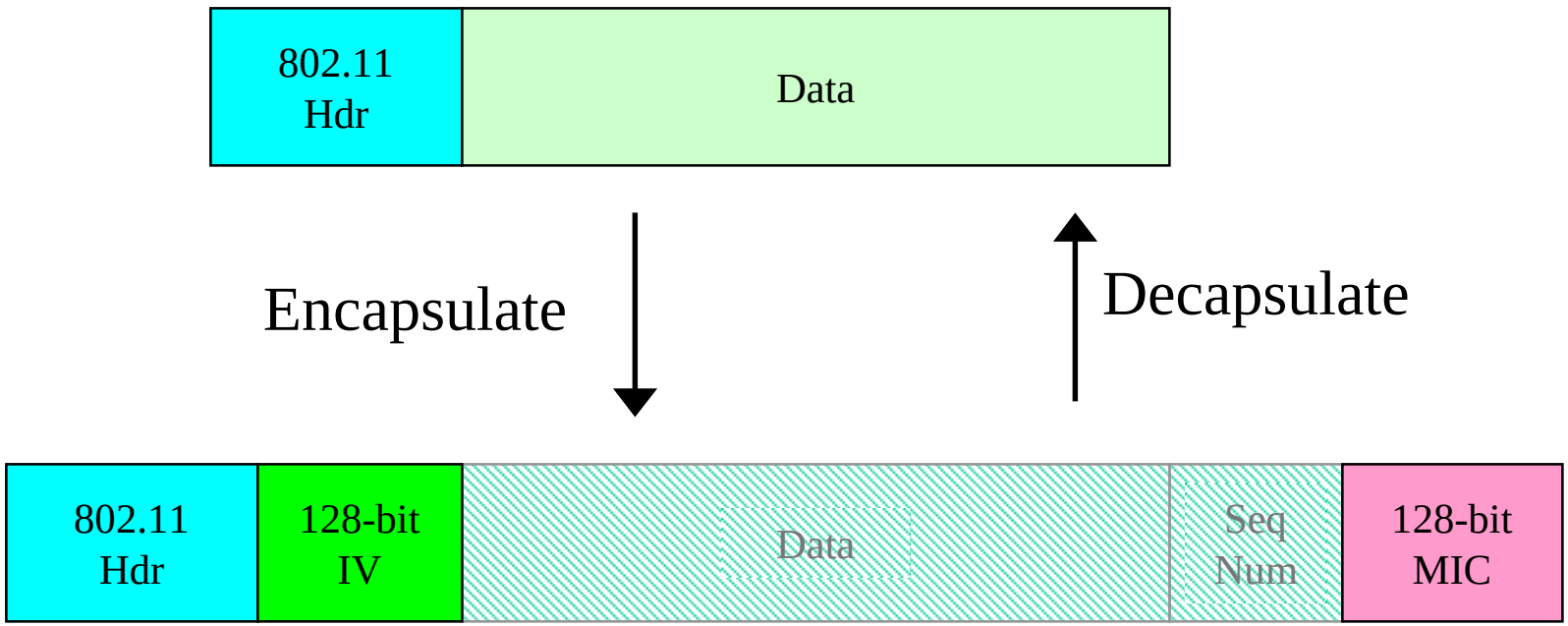
OCB Mode (Full last block)



OCB Mode (Partial last block)



AES-Based WEP Format



Rationale

- AES strong and fast and parallelizable
- OCB mode provides both data authenticity and encryption
  - This avoids many common implementation and design errors
  - Minimizes number of gates to implement
  - Fast and parallelizable
  - Doesn’t fail catastrophically if IV is reused

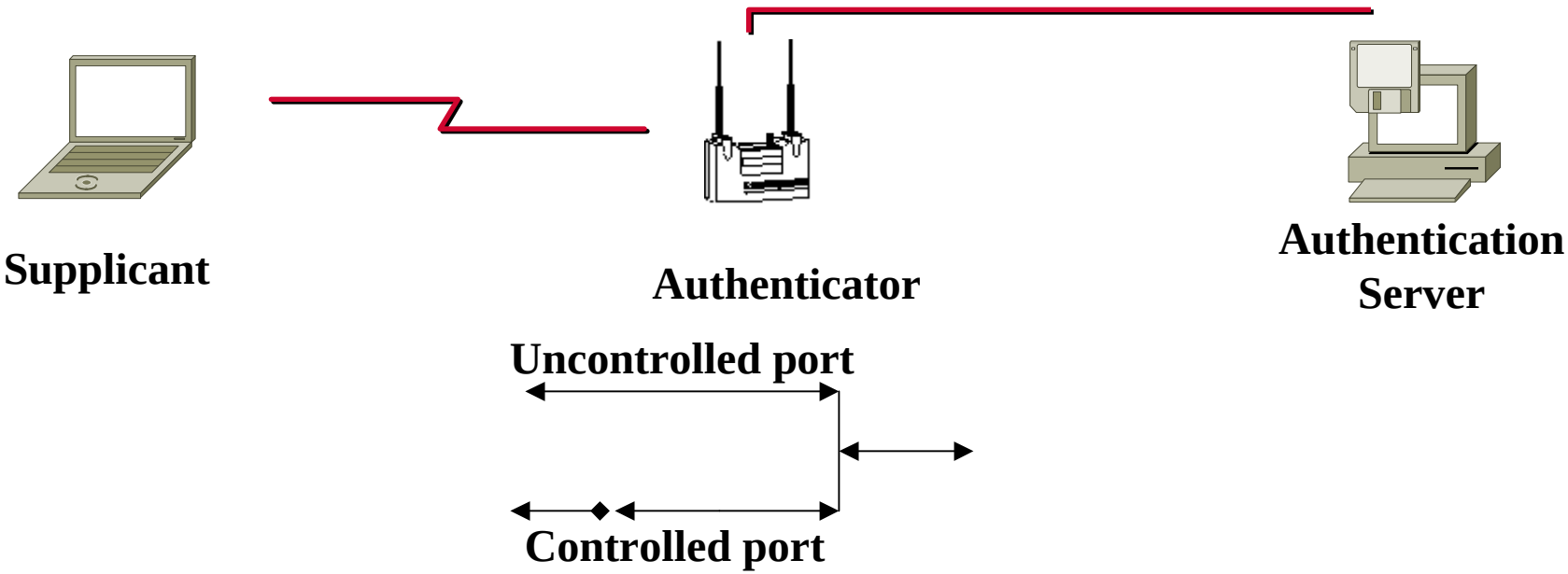
Agenda

- Introduction and Goals
- 802.11 Security Today
- What's Wrong Today?
- Proposed Encapsulation
- Proposed Authentication, Authorization, and Key Management
- Summary

Approach

- Based on existing protocols
  - Kerberos V (RFC 1510)
  - GSS-API (RFC 2743)
  - IAKERB (draft-ietf-cat-iakerb-05.txt)
  - EAP-GSS (draft-aboba-pppext-eapgss-02.txt)
  - EAP (RFC 2284)
  - 802.1X/EAPOL
- 802.11enhancements
  - MAC security management
  - New model for authentication/association sequences

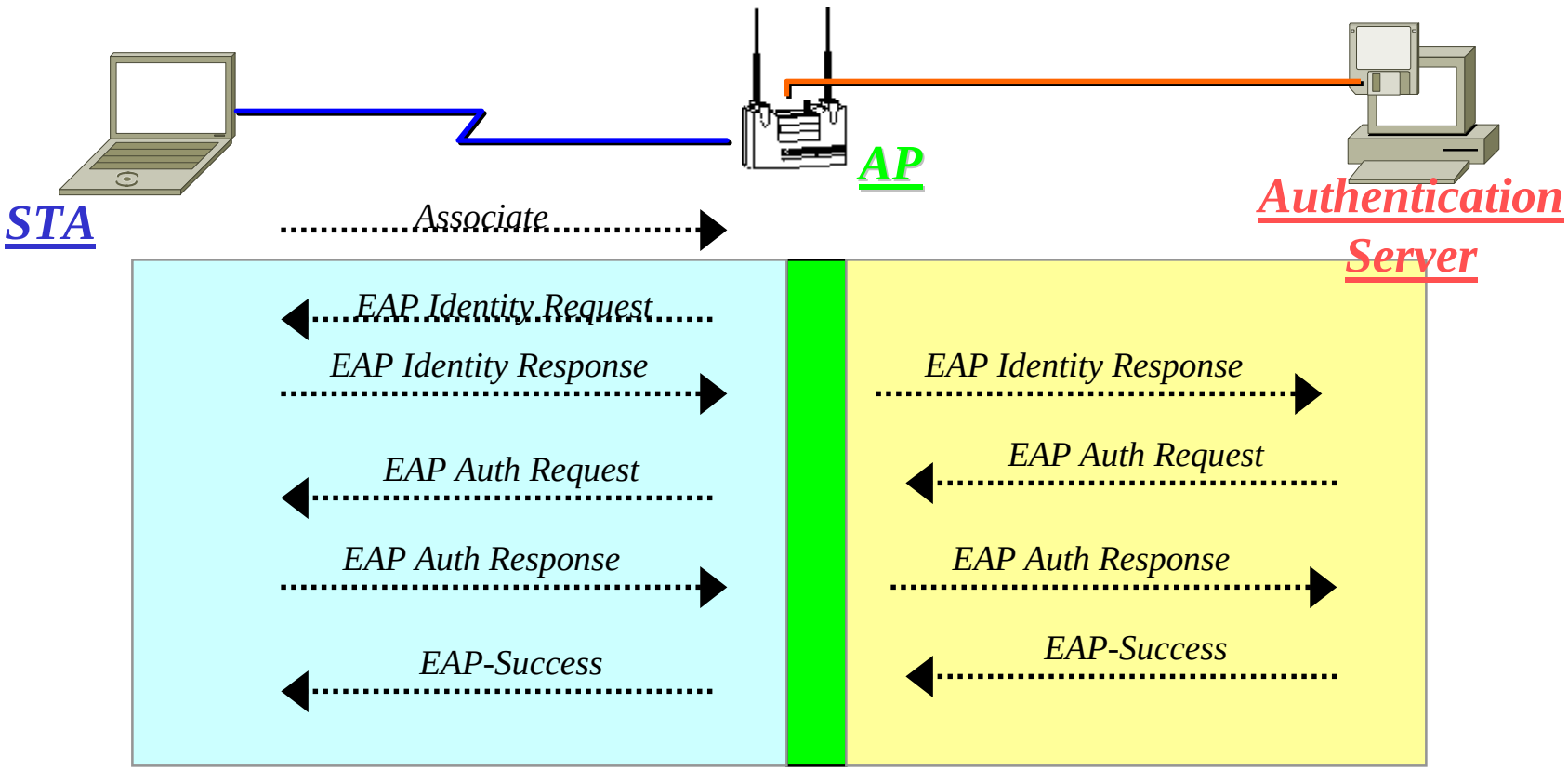
IEEE 802.1X Terminology



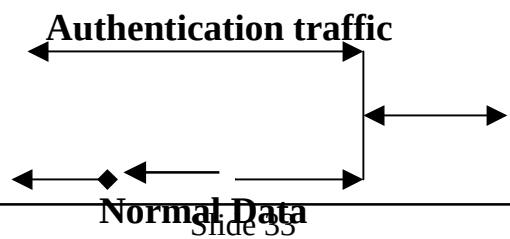
802.1X

- created to control access to any 802 LAN
- used as a transport for ***Extensible Authentication Protocol*** (EAP, RFC 2284)

802.1X Model



Port Status:



EAP Framework

- EAP provides a flexible link layer security framework
  - Simple encapsulation protocol
    - No dependency on IP
    - ACK/NAK, no windowing
    - No fragmentation support
  - Few link layer assumptions
    - Can run over any link layer (PPP, 802, etc.)
    - Assumes no re-ordering
    - Can run over lossy or lossless media
      - Retransmission responsibility of authenticator (not needed for 802.1X or 802.11)
- EAP methods based on IETF standards
  - Transport Level Security (TLS) (supported in Windows 2000)
  - GSS_API (including Kerberos)

EAP-GSS and IAKERB

- EAP-GSS (draft-aboba-pppext-eapgss-02.txt)
  - Use of GSS_API authentication methods within EAP
- IAKerb (draft-ietf-cat-iakberb-05.txt)
  - GSS-API method enabling proxy Kerberos
  - STA not able to talk to KDC directly prior to authentication
  - Initial authentication
    - IAKERB enables STA to obtain TGT, Ticket to AP
  - Handoff
    - Ticket to AP

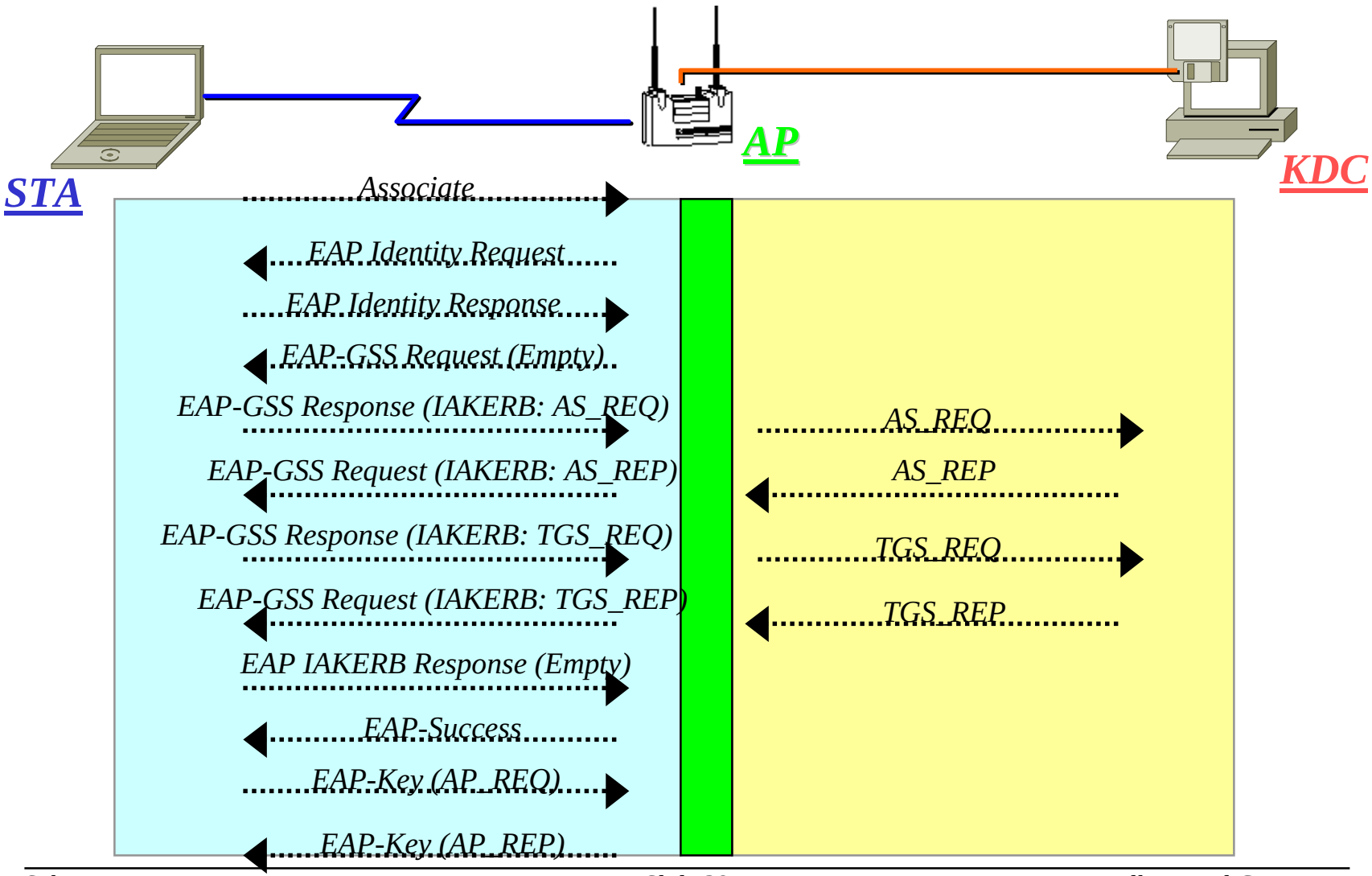
Advertising Security Options

- Modeled on “supported rates”
- AP advertises security options in probe response
  - Placed in probe response only if STA requests it in probe request
- STAs collect this information prior to associations and can make association and roaming decisions based upon it

Selecting security options

- STA requests security options in association request from available options contained in probe response
- AP accepts/rejects association based on request contents
- No additional protocol handshakes necessary
  - No impact on roaming performance

Initial Contact



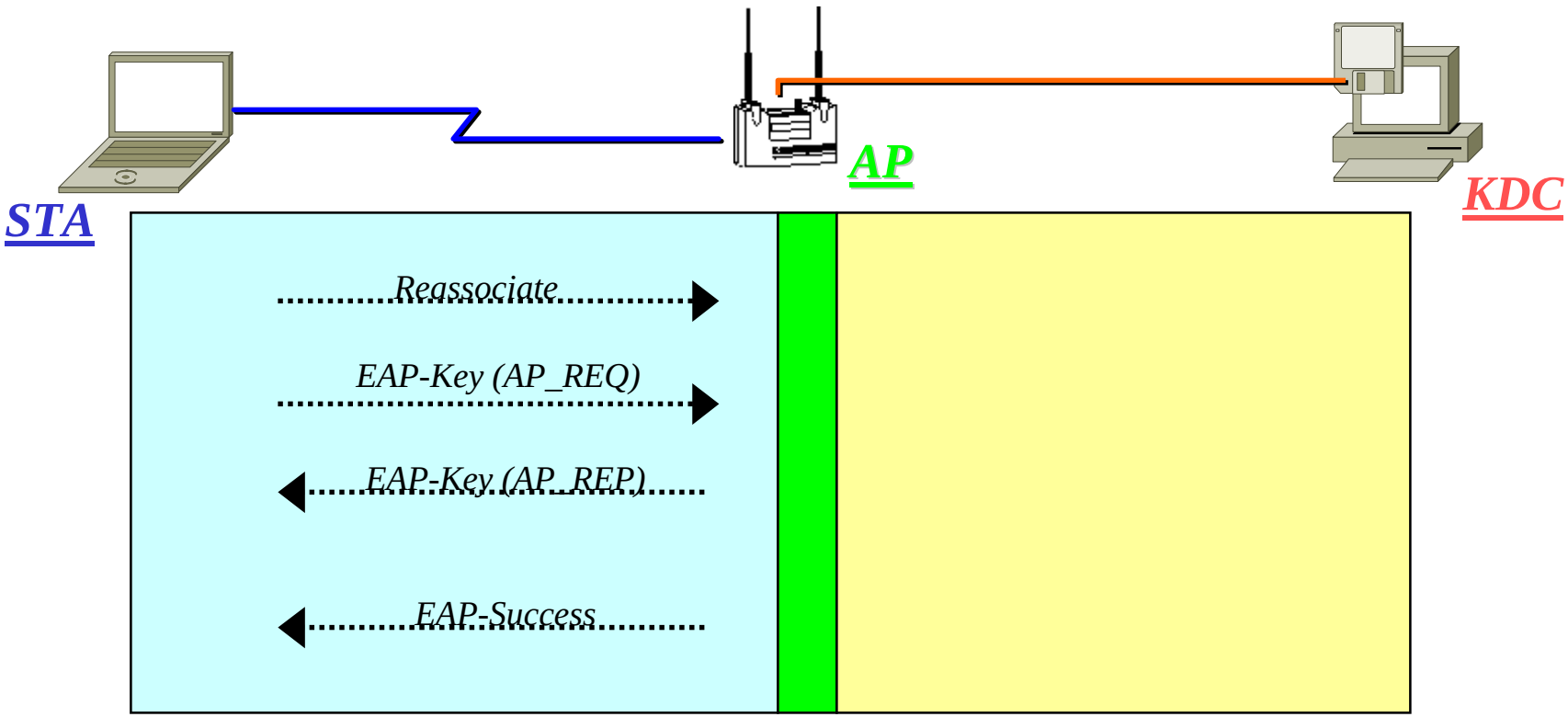
Operational Details

- Authentication method defaults to IAKERB
  - STA can attempt SPNEGO
  - AP can choose IAKERB if it doesn't support anything else
- EAP-Key packets passed up and down via driver interface and 802.11 SAP
  - On STA, GSS_API implementation needs to be able to generate AP_REQ, send it down to driver
  - On AP, need ability to validate received AP_REQ, force 802.1X controlled port into authorized state
- 802.11 encryption turned on after AP_REQ/AP_REP exchange
  - AP turns on encryption after sending AP_REP
  - STA turns on encryption after receiving AP_REP
  - If EAP-Key exchange occurs prior to completion of 802.1X, then part of the 802.1X exchange may be encrypted!

Security Services

- Authentication of client to KDC (TGS_REQ)
  - PADATA typically NOT used with AS_REQ!
- Authentication of KDC to client (AS_REP, TGS_REP)
- Session key for client-AP communication (TGS_REP, AP_REQ)
- TGT, Session key for client-KDC communication (AS_REP)
- Authentication of client to AP (AP_REQ)
- Authentication of AP to client (AP_REP)

Roaming Within Realm



Assumes: APs can share identity and key

Security Services

- Authentication method defaults to IAKERB
- 802.11 encryption turned on after AP_REQ/AP_REP exchange
- Authentication of client to KDC (TGS_REQ)
- Authentication of KDC to client (AS_REP, TGS_REP)
- Session key for client-AP communication (TGS_REP, AP_REQ)
- TGT, Session key for client-KDC communication (AS_REP)
- Authentication of client to AP (AP_REQ)
- Authentication of AP to client (AP_REP)

Agenda

- Introduction and Goals
- 802.11 Security Today
- What's Wrong Today?
- Proposed Encapsulation
- Proposed Authentication, Authorization, and Key Management
- Summary

Summary

- 802.11 security doesn't meet any of its security objectives today
- 802.11 TGe is working to replace
  - Authentication scheme using 802.1X and Kerberos
  - Encryption scheme using AES in OCB mode
- More to come
  - "*The paint's not dry*"

Feedback?