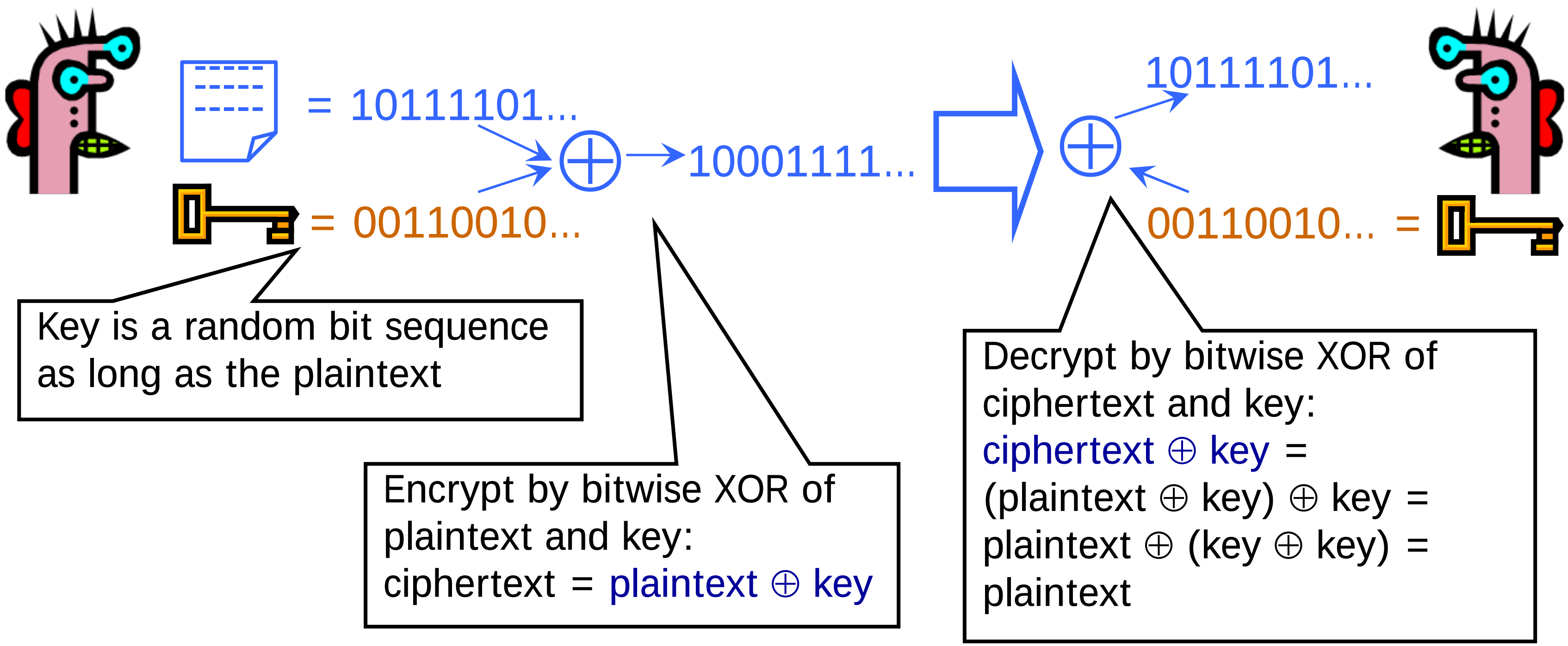
CS 380S

How Things Goes Wrong: Misuse of Cryptography in Secure System Design

Vitaly Shmatikov

One-Time Pad



Cipher achieves perfect secrecy if and only if there are as many possible keys as possible plaintexts, and every key is equally likely (Claude Shannon)

Problems with One-Time Pad

- Key must be as long as plaintext
  - Impractical in most realistic scenarios
  - Still used for diplomatic and intelligence traffic
- Does not guarantee integrity
  - One-time pad only guarantees confidentiality
  - Attacker cannot recover plaintext, but can easily change it to something else
- Insecure if keys are reused
  - Attacker can obtain XOR of plaintexts

Stream Ciphers

- With one-time pad,

  Ciphertext(Key,Message)=Message⊕Key

  - Key must be a random bit sequence as long as message
- Idea: replace "random" with "pseudo-random"
  - Encrypt with pseudo-random number generator (PRNG)
  - PRNG takes a short, truly random secret seed and expands it into a long "random-looking" sequence
    - E.g., 128-bit seed into a 10^6-bit pseudo-random sequence

No efficient algorithm can tell this sequence from truly random

- Ciphertext(Key,Msg)=IV, Msg⊕PRNG(IV,Key)
  - Message processed bit by bit, not in blocks

Properties of Stream Ciphers

- Usually very fast (faster than block ciphers)
  - Used where speed is important: WiFi, DVD
- Unlike one-time pad, stream ciphers do not provide perfect secrecy
  - Only as secure as the underlying PRNG
  - If used properly, can be as secure as block ciphers
- PRNG is, by definition, unpredictable
  - Given the stream of PRNG output (but not the seed!), it's hard to predict what the next bit will be
    - If PRNG(unknown random seed)=$b_1 \ldots b_i$, then b_{i+1} is "0" with probability ½, "1" with probability ½

Stream Cipher Terminology

- Seed of pseudo-random generator often consists of initialization vector (IV) and key
  - IV is usually sent with the ciphertext
  - The key is a secret known only to the sender and the recipient, not sent with the ciphertext
- The pseudo-random bit stream produced by PRNG(IV,key) is referred to as keystream
  - PRNG must be cryptographically secure
- Encrypt message by XORing with keystream
  - ciphertext = message $\oplus$ keystream

How Random is “Random?”

Cryptographically Secure PRNG

- Next-bit test
  - Given N bits of the pseudo-random sequence, predict $(N+1)^{st}$ bit
  - Probability of correct prediction should be very close to 1/2 for any efficient adversarial algorithm
- PRNG state compromise
  - Even if attacker learns complete or partial state of the PRNG, he should not be able to reproduce the previously generated sequence
    - ... or future sequence, if there'll be future random input(s)
- Common PRNGs are not cryptographically secure

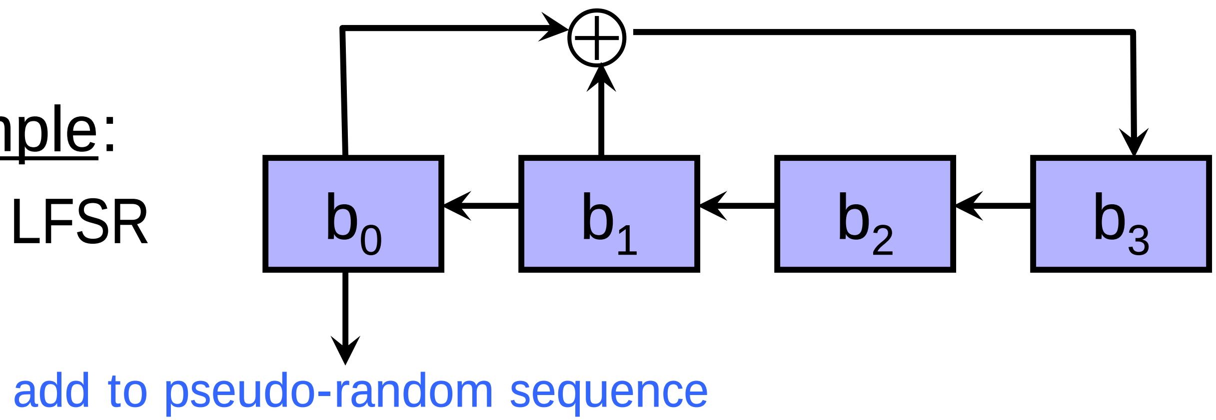
Weaknesses of Stream Ciphers

- No integrity
  - Associativity & commutativity: $(X \oplus Y) \oplus Z = (X \oplus Z) \oplus Y$
  - $(M_1 \oplus PRNG(seed)) \oplus M_2 = (M_1 \oplus M_2) \oplus PRNG(seed)$
- Known-plaintext attack is very dangerous if keystream is ever repeated
  - Self-cancellation property of XOR: $X \oplus X = 0$
  - $(M_1 \oplus PRNG(seed)) \oplus (M_2 \oplus PRNG(seed)) = M_1 \oplus M_2$
  - If attacker knows M_1, then easily recovers M_2
    - Most plaintexts contain enough redundancy that knowledge of M_1 or M_2 is not even necessary to recover both from $M_1 \oplus M_2$

Linear Feedback Shift Register (LFSR)

Example:

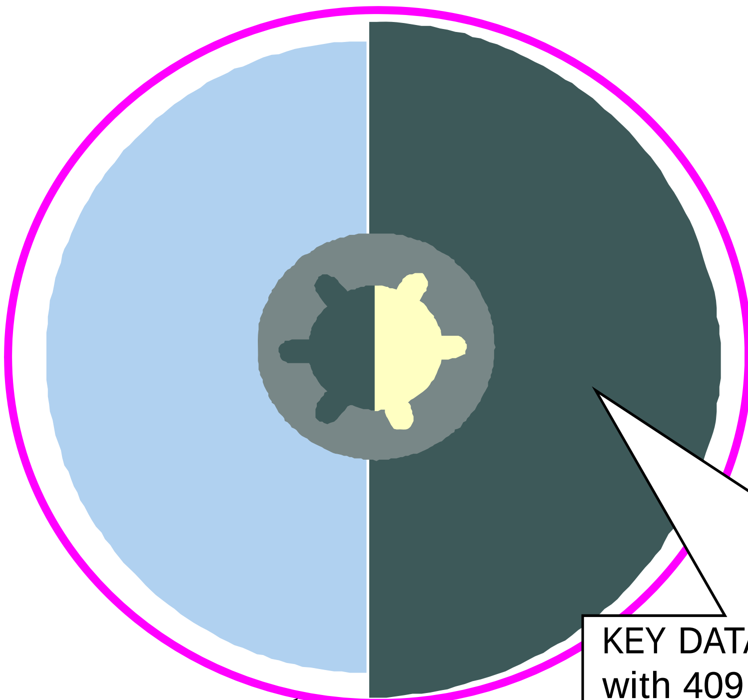
4-bit LFSR

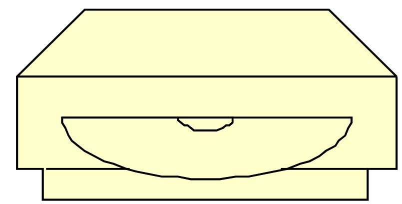


- Key is used as the seed
  - For example, if the seed is 1001, the generated sequence is 1001101011110001001...
- Repeats after 15 bits (2^4-1)

Content Scrambling System (CSS)

- DVD encryption scheme from Matsushita and Toshiba





Each player has its own PLAYER KEY
(409 player manufacturers,
each has its player key)

KEY DATA BLOCK contains disk key encrypted with 409 different player keys:

- $\text{Encrypt}_{\text{DiskKey}}$(DiskKey)
- $\text{Encrypt}_{\text{PlayerKey1}}$(DiskKey) ... $\text{Encrypt}_{\text{PlayerKey409}}$(DiskKey)

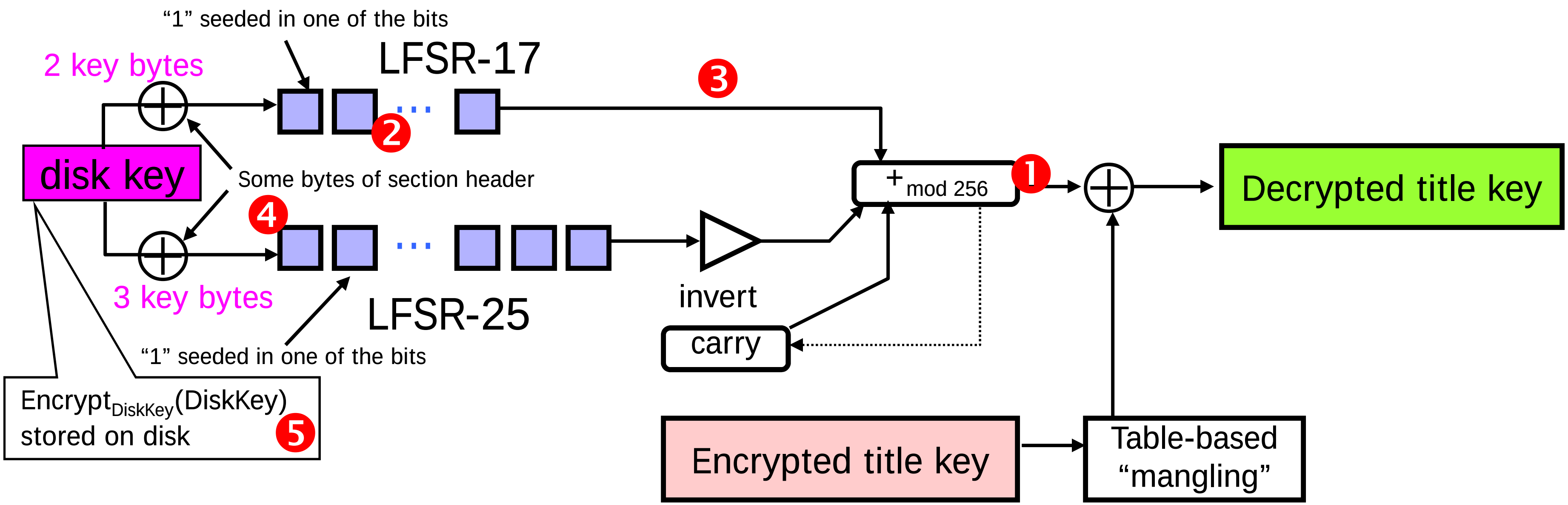
Each DVD is encrypted with
a disk-specific 40-bit DISK KEY

This helps attacker
verify his guess of disk key

What happens if even a single
player key is compromised?

Attack on CSS Decryption Scheme

[Frank Stevenson]



❶ With known ciphertext and plaintext, guess the byte output by XOR; repeat 5 times to recover 5 output bytes – this takes $O(2^8)$

❷ Guess 2 bytes of the key contained in LFSR-17 – this takes $O(2^{16})$

❸ Clock out 3 bytes out of LFSR-17, use them to determine the corresponding output bytes of LFSR-25 (this reveals all of LFSR-25 except the highest bit)

❹ Clock back 3 bytes, try both possibilities – this takes O(2)

❺ Verify the key

This attack takes $O(2^{25})$

DeCSS

- In CSS, disk key is encrypted under hundreds of different player keys
  - ... including Xing, a software DVD player
- Reverse engineering object code of Xing revealed its decryption key
  - Recall that every CSS disk contains the master disk key encrypted under Xing's key
  - One bad player $\Rightarrow$ entire system is broken!
- Easy-to-use DeCSS software

DeCSS Aftermath

- DVD CCA sued Jon Lech Johansen, one of DeCSS authors (eventually dropped)
- Publishing DeCSS code violates copyright
  - Underground distribution as haikus and T-shirts
  - "Court to address DeCSS T-Shirt: When can a T-shirt become a trade secret? When it tells you how to copy a DVD." - From Wired News

RC4

- Designed by Ron Rivest for RSA in 1987
- Simple, fast, widely used
  - SSL/TLS for Web security, WEP for wireless

```
Byte array S[256] contains a permutation of numbers from 0 to 255
i = j := 0
loop
    i := (i+1) mod 256
    j := (j+S[i]) mod 256
    swap(S[i],S[j])
    output (S[i]+S[j]) mod 256
end loop
```

RC4 Initialization

```
Divide key K into L bytes
for i = 0 to 255 do
    S[i] := i
j := 0
for i = 0 to 255 do
        j := (j+S[i]+K[i mod L]) mod 256
        swap(S[i],S[j])
```

Key can be any length up to 2048 bits

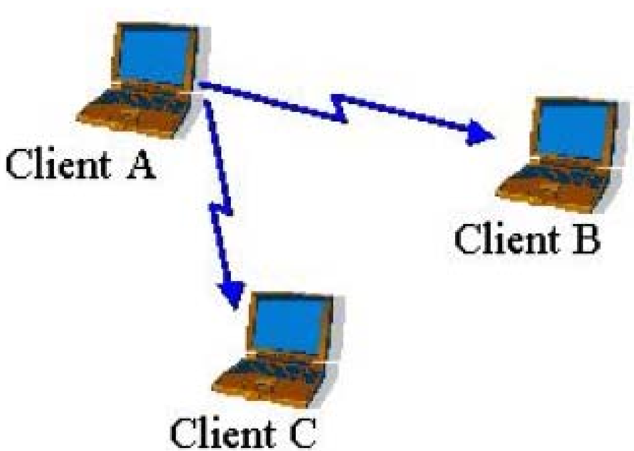
Generate initial permutation from key K

- To use RC4, usually prepend initialization vector (IV) to the key
  - IV can be random or a counter
- RC4 is not random enough! 1st byte of generated sequence depends only on 3 cells of state array S. This can be used to extract the key.
  - To use RC4 securely, RSA suggests discarding first 256 bytes

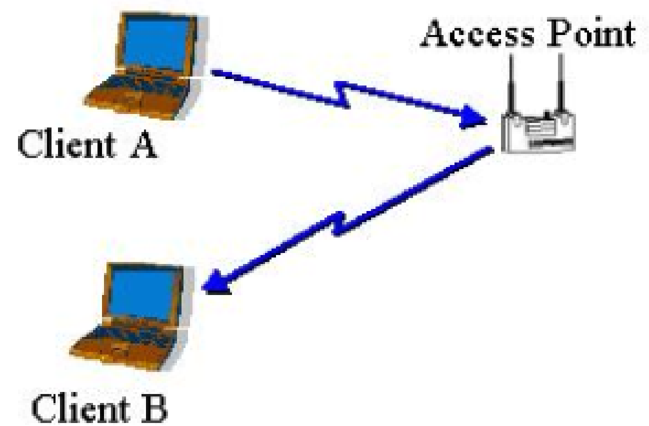
Fluhrer-Mantin-Shamir attack

802.11b Overview

- Standard for wireless networks (IEEE 1999)
- Two modes: infrastructure and ad hoc



IBSS (ad hoc) mode



BSS (infrastructure) mode

Access Point SSID

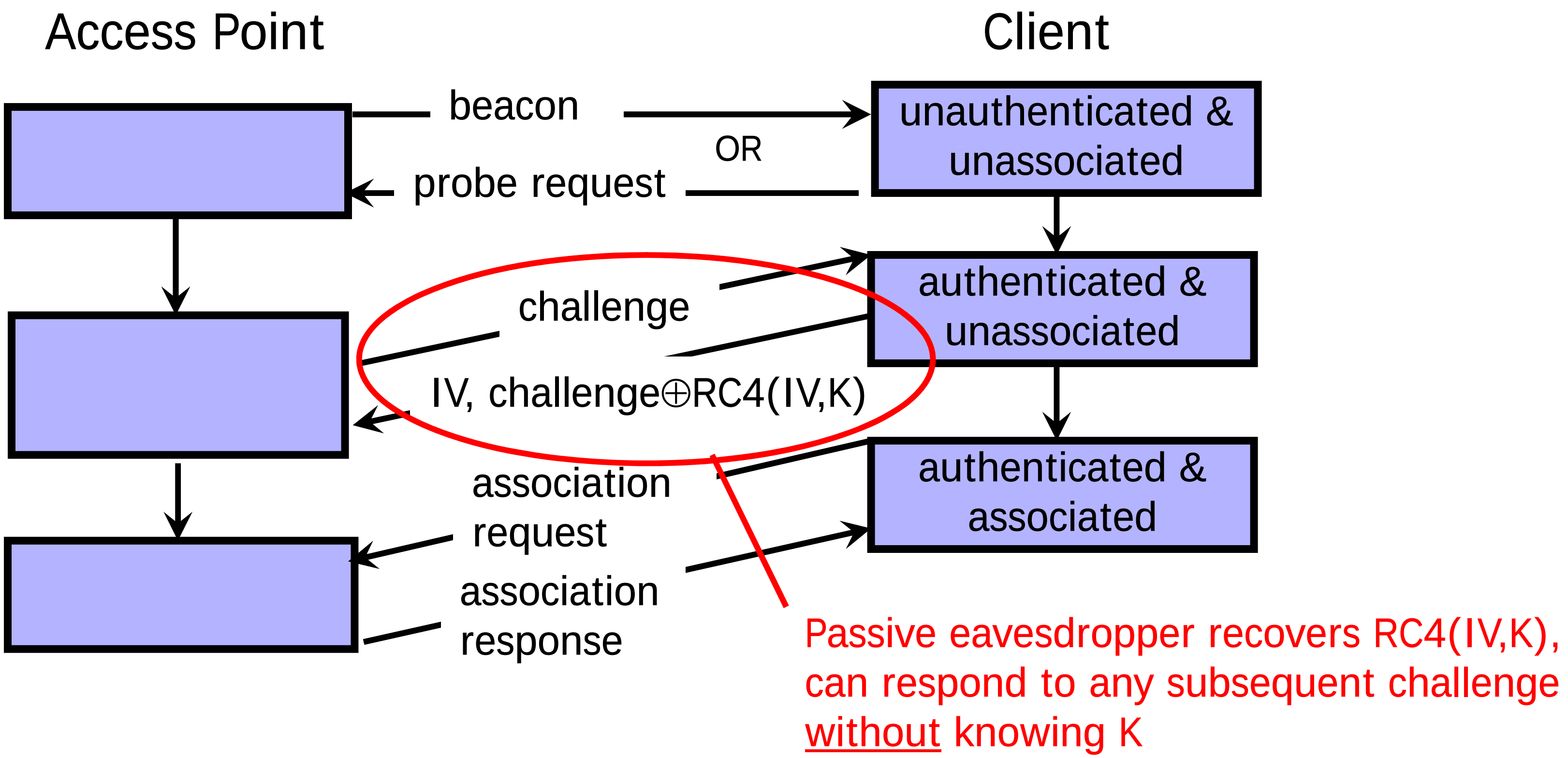
- Service Set Identifier (SSID) is the “name” of the access point
  - By default, access point broadcasts its SSID in plaintext “beacon frames” every few seconds
- Default SSIDs are easily guessable
  - Linksys defaults to “linksys”, Cisco to “tsunami”, etc.
  - This gives away the fact that access point is active
- Access point settings can be changed to prevent it from announcing its presence in beacon frames and from using an easily guessable SSID
  - But then every user must know SSID in advance

WEP: Wired Equivalent Privacy

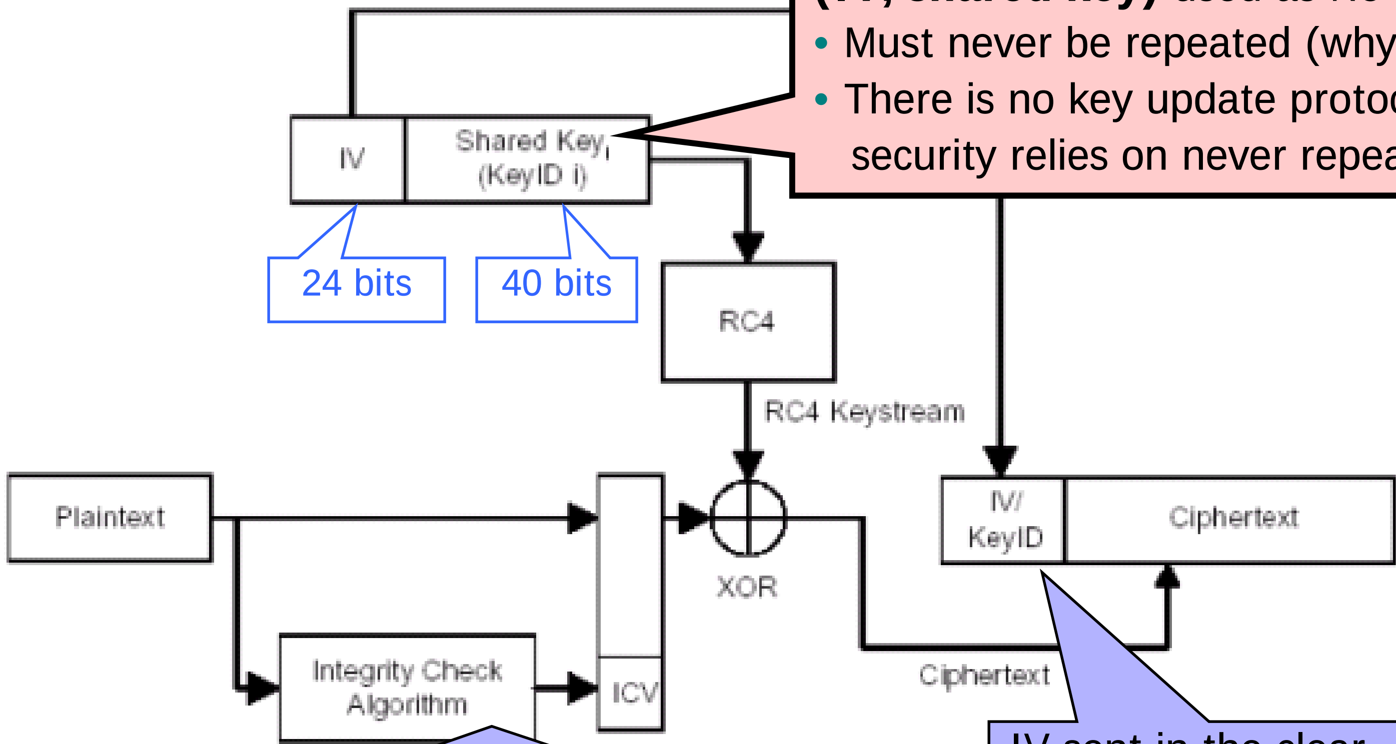
- Special-purpose protocol for 802.11b
  - Intended to make wireless as secure as wired network
- Goals: confidentiality, integrity, authentication
- Assumes that a secret key is shared between access point and client
- Uses RC4 stream cipher seeded with 24-bit initialization vector and 40-bit key
  - Terrible design choice for wireless environment

Shared-Key Authentication

Prior to communicating data, access point may require client to authenticate



How WEP Works



(IV, shared key) used as RC4 seed

- Must never be repeated (why?)
- There is no key update protocol, so security relies on never repeating IV

IV sent in the clear

Worse: <u>changing IV with each packet is optional!</u>

CRC-32 checksum is linear in ⊕:
if attacker flips some plaintext bits, he knows which bits of CRC to flip to produce the <u>same checksum</u>

no integrity!

RC4 Is a Bad Choice for Wireless

- Stream ciphers require synchronization of key streams on both ends of connection
  - This is not suitable when packet losses are common
- WEP solution: a separate seed for each packet
  - Can decrypt a packet even if a previous packet was lost
- But number of possible seeds is not large enough!
  - RC4 seed = 24-bit initialization vector + fixed key
  - Assuming 1500-byte packets at 11 Mbps, 2^{24} possible IVs will be exhausted in about 5 hours
- Seed reuse is deadly for stream ciphers

Recovering Keystream

- Get access point to encrypt a known plaintext
  - Send spam, access point will encrypt and forward it
  - Get victim to send an email with known content
- If attacker knows plaintext, it is easy to recover keystream from ciphertext
  - $C \oplus M = (M \oplus RC4(IV,key)) \oplus M = RC4(IV,key)$
  - Not a problem if this keystream is not re-used
- Even if attacker doesn't know plaintext, he can exploit regularities (plaintexts are not random)
  - For example, IP packet structure is very regular

Keystream <u>Will</u> Be Re-Used

- In WEP, repeated IV means repeated keystream
- Busy network will repeat IVs often
  - Many cards reset IV to 0 when re-booted, then increment by 1 $\Rightarrow$ expect re-use of low-value IVs
  - If IVs are chosen randomly, expect repetition in $O(2^{12})$ due to birthday paradox
- Recover keystream for each IV, store in a table
  - (KnownM $\oplus$ RC4(IV,key)) $\oplus$ KnownM = RC4(IV,key)
- Wait for IV to repeat, decrypt and enjoy plaintext
  - (M' $\oplus$ RC4(IV,key)) $\oplus$ RC4(IV,key) = M'

It Gets Worse

- Misuse of RC4 in WEP is a design flaw with no fix
  - Longer keys do not help!
    - The problem is re-use of IVs, their size is fixed (24 bits)
  - Attacks are passive and very difficult to detect
- Perfect target for Fluhrer et al. attack on RC4
  - Attack requires known IVs of a special form
  - WEP sends IVs in plaintext
  - Generating IVs as counters or random numbers will produce enough “special” IVs in a matter of hours
- This results in key recovery (not just keystream)
  - Can decrypt even ciphertexts whose IV is unique

Weak Countermeasures

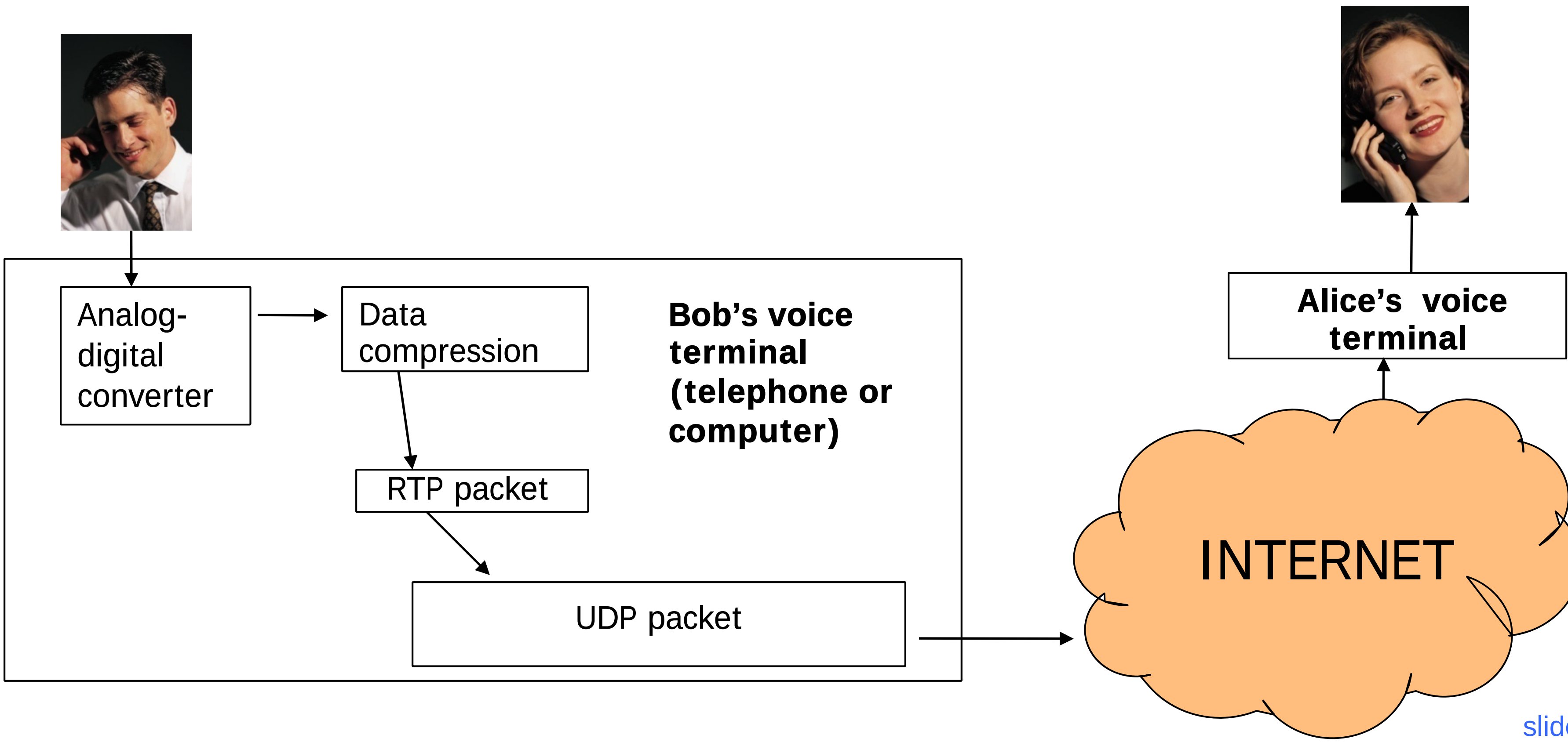
- Run VPN on top of wireless
  - Treat wireless as you would an insecure wired network
  - VPNs have their own security and performance issues
    - Compromise of one client may compromise entire network
- Hide SSID of your access point
  - Still, raw packets will reveal SSID (it is not encrypted!)
- Have each access point maintain a list of network cards addresses that are allowed to connect to it
  - Infeasible for large networks
  - Attacker can sniff a packet from a legitimate card, then re-code (spoof) his card to use a legitimate address

Fixing the Problem

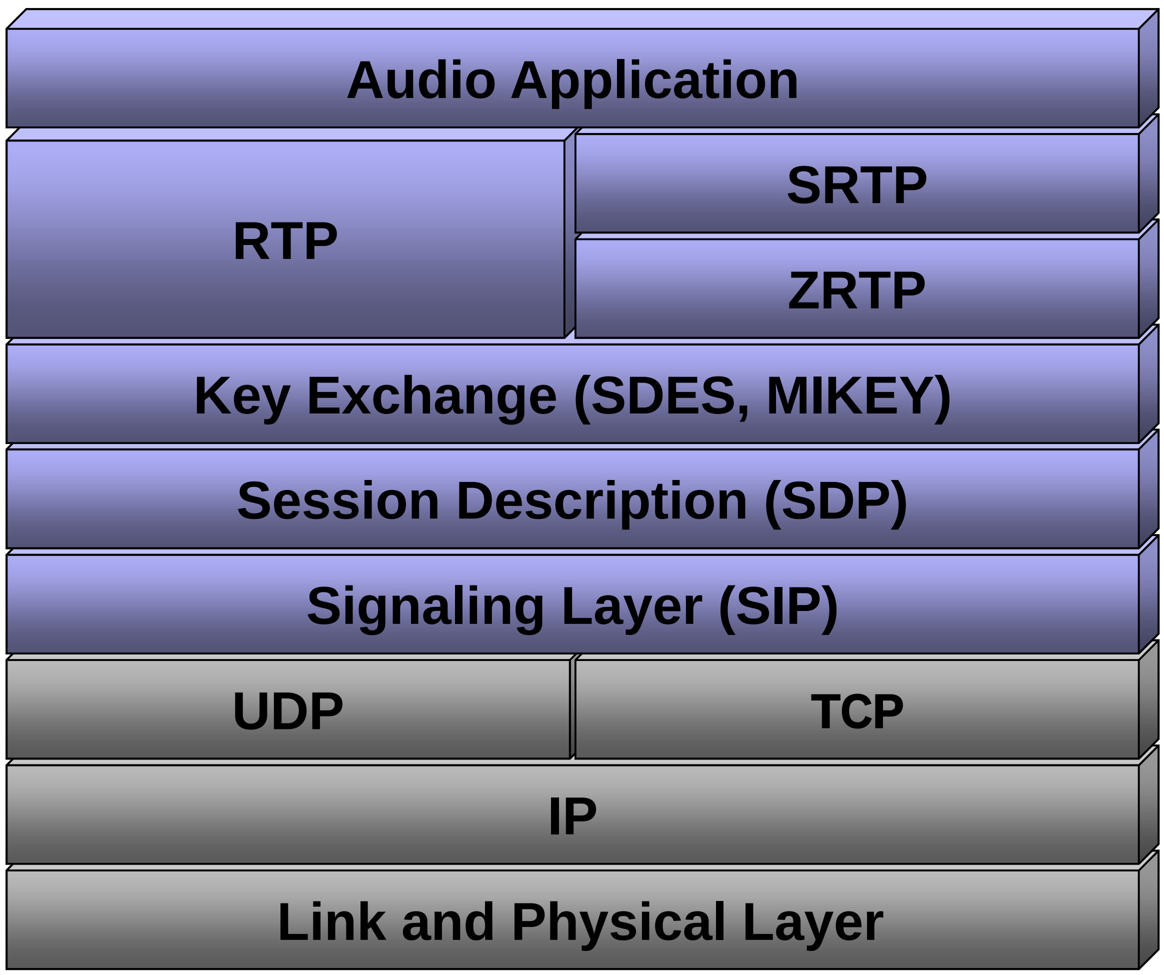
- Extensible Authentication Protocol (EAP)
  - Developers can choose their own authentication method
    - Passwords (Cisco EAP-LEAP), public-key certificates (Microsoft EAP-TLS), passwords OR certificates (PEAP), etc.
- 802.11i standard fixes 802.11b problems
  - Patch: TKIP. Still RC4, but encrypts IVs and establishes new shared keys for every 10 KBytes transmitted
    - No keystream re-use, prevents exploitation of RC4 weaknesses
    - Use same network card, only upgrade firmware
  - Long-term: AES in CCMP mode, 128-bit keys, 48-bit IVs
    - Block cipher (in special mode) instead of stream cipher
    - Requires new network card hardware

VoIP

- **V**oice **O**ver **I**nternet **P**rotocol
  - Voice communications over packet data networks



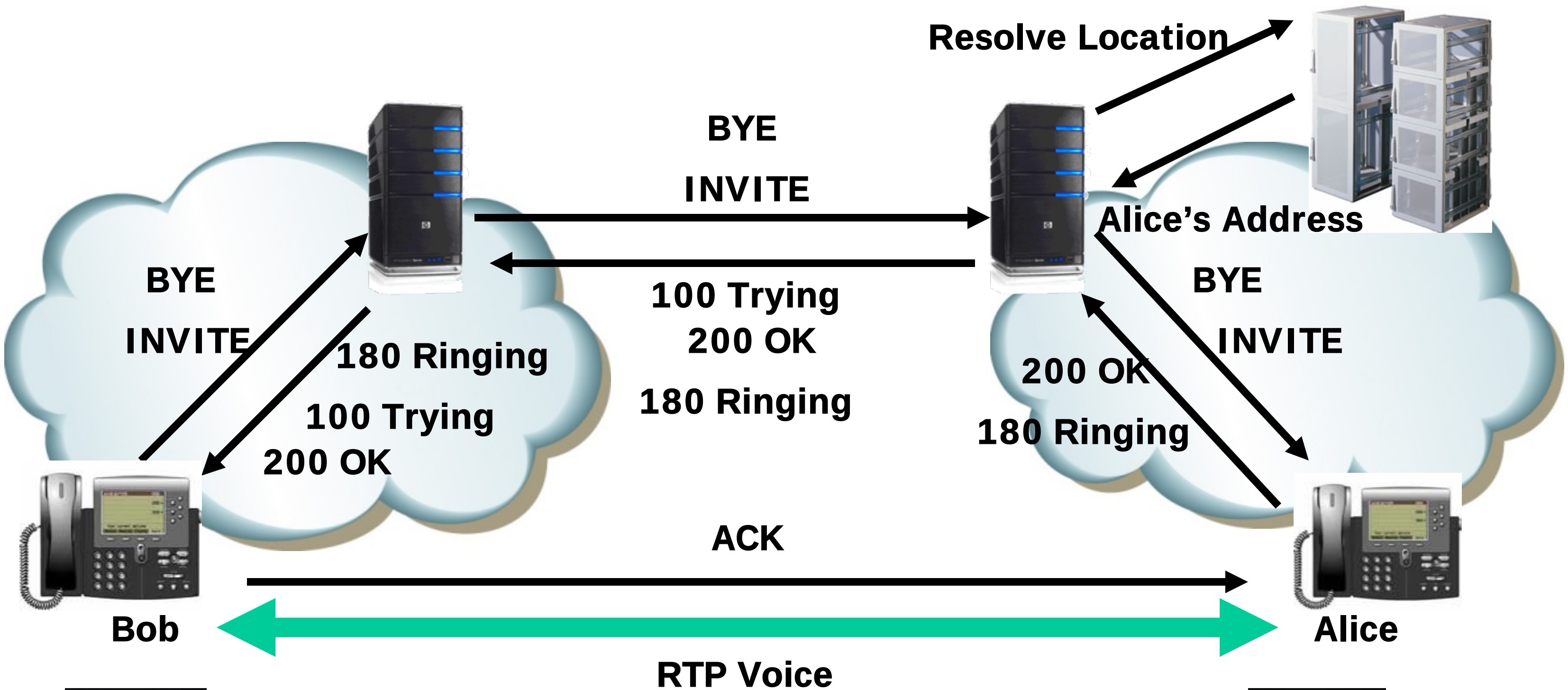
VoIP Protocol Stack

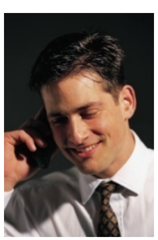


SIP

- Session Initiation Protocol (SIP) is an application-layer protocol for creating, modifying and terminating VoIP sessions
- SIP network elements
  - End Points/User Agents (UAs), Proxy/Redirect Server, Location Server, Registrar
- Modeled on HTTP, uses URI similar to email addresses
  - sip: alice@domain.com

SIP Call Flow Illustration





Bob calls Alice

Hi.

Bye.

Hello.

Alice answers the phone

What Does VoIP Security Mean?

- Confidentiality: Voice datagrams under encryption are indistinguishable from random
- Message authentication: If Alice receives a msg from Bob, then indeed it was sent by Bob
- Data integrity: Any modification of data in transit is detected by recipients
- Replay protection: Any attempt to replay data from an old session is detected by participants

SIP Security Mechanisms (1)

- HTTP digest authentication
  - One-way authentication, replay prevention
- Network/transport layer
  - IPsec
    - Hop-by-hop security for UDP, TCP, SCTP
    - Need to manage keys, IPsec profiles
  - TLS
    - Must be implemented by all proxies
    - Hop-by-hop security (i.e., all proxies must be trusted)
    - Can not be applied to UDP-based SIP (only TCP or other reliable transport protocol)

SIP Security Mechanisms (2)

- S/MIME
  - Encrypted email, basically
  - Use for public key distribution, authentication, integrity, and confidentiality of SIP signaling data
  - Protect SIP header fields through tunneling entire SIP message as an S/MIME body
    - End-to-end (as opposed to hop-by-hop) security
- SIP Authenticated Identity Body
  - Basically same as S/MIME tunneling, but instead of “tunneling” the entire message, only a specific subset of headers are signed

Attacks on SIP

- SIP is very vulnerable to denial of service
- Reflection: send a large number of spoofed requests to SIP proxies using victim's IP address
- Session termination using unauthenticated BYE requests
- Registration requests may not be properly authenticated - register/deregister any user
- Flood location server with false bindings
- Impersonate proxy server

VoIP Key Establishment

- Use a special-purpose key exchange protocol
  - MIKEY, ZRTP, several others

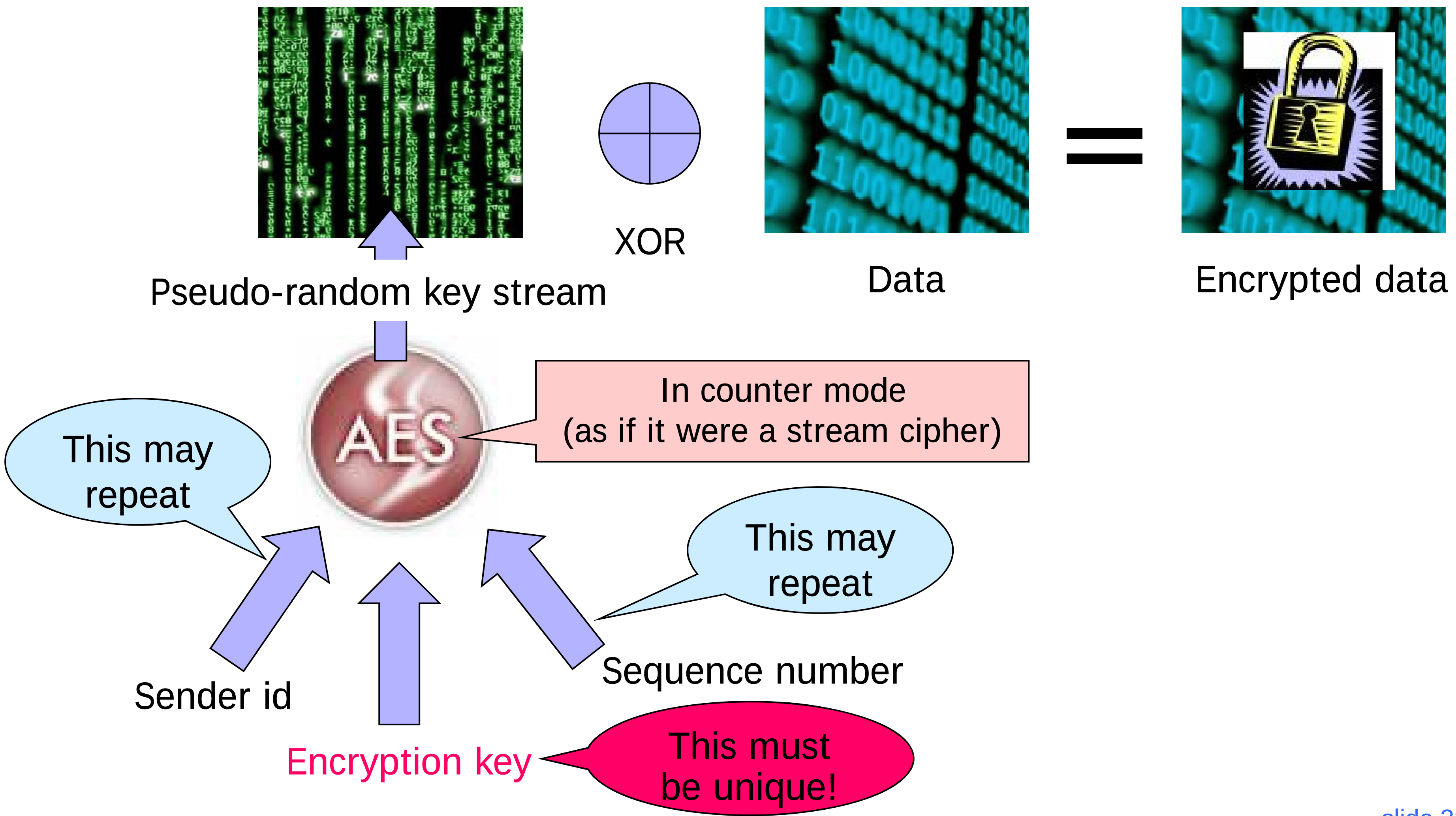
OR

- Initiator sends key as part of Session Description Protocol (session setup)
  - How is the key protected? This is the responsibility of signaling protocol (SIP)

SDP and SDES

- VoIP session parameters are described by Session Description Protocol (SDP)
  - This includes key establishment (SDES)
- SDES: key is directly embedded in the payload of a SIP message
  - Effectively, relies on SIP to transmit key securely
- This key will be used by Secure Real-Time Transport Protocol (SRTP) to protect actual media stream (voice datagrams)

SRTP Encryption



Security of SRTP Encryption

- Security of stream ciphers relies critically upon keystream never repeating
  - If keystream repeats, attacker can replay datagrams from an old session and they will decrypt correctly
- SRTP encryption keys are computed deterministically from key material provided by the key establishment protocol
  - Key material must never repeat

The property must be guaranteed by the key establishment protocol!

SDES (with S/MIME) + SRTP

- When SDES is used for key establishment, uniqueness of master key is SIP's responsibility
- SIP may protect keys using S/MIME (basically, keys are sent inside encrypted email)
  - Not supported by most existing VoIP installations, but can be used if end-to-end security is essential
  - S/MIME does <u>not</u> guarantee freshness
    - No replay prevention mechanisms
  - Mismatch between SRTP requirements and SIP
- Result: SRTP encryption re-uses keystream
  - Attack similar in spirit to attack on 802.11b

Lessons

- Protocol interaction matters!
- It's important to analyze the entire protocol stack, not just protocols in isolation
- Mismatch between inter-layer assumptions and guarantees leads to attacks
  - SIP is similar to HTTP, but end-to-end protection requires S/MIME, not TLS
  - SRTP expects that the key is fresh each time, but there is no replay protection in S/MIME