# Solutions to Selected Problems

Guide to Internet Cryptography

Companion Material

February 7, 2026

## Preface

This document provides solutions to selected problems from the book *Guide to Internet Cryptography: Security Protocols and Real-World Attack Implications*. The material is intended for educational use in courses and self-study.

**Book website:** https://link.springer.com/book/10.1007/978-3-031-19439-9

## 1 Chapter 6: WLAN

**Problem 6.1 LAN specific attacks**

Can you think of a non-cryptographic countermeasure to detect ARP Spoofing attacks?

**Solution**

### Non-Cryptographic Countermeasures for ARP Spoofing Detection

### 1. Static ARP Tables

**Method:** Manually configure and maintain static ARP entries for critical hosts.
**Implementation:**

```
# Linux/Unix
arp -s 192.168.1.1 00:11:22:33:44:55

# Windows
arp -s 192.168.1.1 00-11-22-33-44-55
```

**Advantages:**

- Prevents ARP cache poisoning for specified entries
- Simple to implement for small networks
- No performance overhead

**Disadvantages:**

- Doesn't scale to large networks
- Requires manual maintenance
- Network changes require manual updates

### 2. ARP Table Monitoring and Anomaly Detection

**Method:** Monitor ARP tables for suspicious changes.
**Detection criteria:**

- **Duplicate IP addresses:** Multiple MAC addresses claiming the same IP
- **MAC address changes:** An IP address suddenly associated with a different MAC
- **Frequency analysis:** Unusually high rate of ARP replies
- **Unsolicited ARP replies:** ARP replies without corresponding requests

**Tools:**

- `arpwatch` – monitors ARP activity and reports changes
- `arpalert` – detects unusual ARP activity
- Custom scripts monitoring `/proc/net/arp` or using `arp -a`

### 3. Passive Network Monitoring

**Method:** Deploy a dedicated monitoring system that passively observes all ARP traffic.
**Detection techniques:**

1. **IP-MAC binding database:**
   - Build a database of legitimate IP-to-MAC mappings
   - Alert on deviations from known-good mappings
2. **Gratuitous ARP detection:**
   - Monitor for gratuitous ARP packets (unsolicited announcements)
   - Flag unexpected gratuitous ARPs, especially for gateway addresses
3. **ARP request/reply correlation:**
   - Track ARP requests and verify replies match
   - Detect replies for non-existent requests

### 4. Switch Port Security

**Method:** Configure network switches to limit MAC addresses per port.
**Implementation on Cisco switches:**

```
interface FastEthernet0/1
  switchport mode access
  switchport port-security
  switchport port-security maximum 1
  switchport port-security mac-address sticky
  switchport port-security violation restrict
```

**How it helps:**

- Limits number of MAC addresses per physical port
- Prevents attacker from spoofing multiple MAC addresses
- Can automatically learn and lock legitimate MAC addresses

### 5. DHCP Snooping Integration

**Method:** Use DHCP snooping to build a trusted IP-MAC binding database.
**Process:**

1. Switch monitors DHCP transactions
2. Builds binding table: (IP, MAC, Port, VLAN, Lease time)
3. Only allows ARP packets matching the binding table
4. Blocks ARP packets with mismatched IP-MAC pairs

**Note:** While DHCP snooping itself requires switch configuration, the binding database can be used for non-cryptographic ARP validation.

### 6. Bidirectional Verification

**Method:** Verify ARP mappings in both directions.
**Process:**

1. When receiving ARP reply: IP-A $\leftrightarrow$ MAC-A
2. Send ARP request for MAC-A (reverse ARP query)
3. Verify the response matches IP-A
4. If mismatch detected, flag as potential spoofing

### 7. Timing Analysis

**Method:** Analyze timing patterns of ARP traffic.
**Anomalies to detect:**

- **ARP flooding:** Excessive ARP requests/replies in short time window
- **Periodic patterns:** Attackers often send periodic ARP spoofs to maintain poisoned cache
- **Race conditions:** Legitimate host and attacker both responding to ARP requests

**Threshold example:**

$$\text{Alert if: } \frac{\text{ARP packets from MAC-X}}{\text{Time window}} > \text{Threshold} \tag{1}$$

### 8. Cross-Layer Validation

**Method:** Correlate ARP information with other network layers.
**Techniques:**

- **IP-MAC consistency:** Check if IP packets source address matches ARP-claimed MAC
- **Switch CAM table comparison:** Compare ARP table with switch's MAC address table
- **Physical topology validation:** Verify MAC addresses appear on expected switch ports

### 9. Vendor OUI Verification

**Method:** Validate MAC address Organizationally Unique Identifier (OUI).
**Checks:**

- First 3 bytes of MAC identify manufacturer
- Verify known hosts use expected vendor OUIs
- Example: Gateway router should have Cisco OUI, not random/spoofed OUI
- Flag sudden OUI changes for same IP address

**Example:**

```
Known gateway: 192.168.1.1 -> 00:1A:2B:... (Cisco)
Suspicious:    192.168.1.1 -> DE:AD:BE:... (Unknown/Local)
```

### 10. Network Segmentation and VLANs

**Method:** Limit ARP spoofing impact through network design.
**Strategy:**

- Separate network into smaller broadcast domains (VLANs)
- Reduces number of hosts vulnerable to single ARP spoof
- Critical servers on isolated VLANs
- Limits attacker's lateral movement

### Best Practice: Defense in Depth

**Recommendation:** Combine multiple countermeasures:

- Static ARP for critical infrastructure (gateway, DNS, domain controllers)
- DHCP snooping + Dynamic ARP Inspection on switches
- Passive monitoring with `arpwatch` or similar tools
- Port security to limit MAC addresses per port
- Regular monitoring and alerting on anomalies

**Problem 6.2 WEP: IVs**

In SSL 3.0, a technique called *IV chaining* was used: instead of transmitting a randomly chosen IV, the last bytes of the previous ciphertext were used as the IV for the next ciphertext. Could this approach also be applied to WEP, adding 24-bit data capacity to each WLAN frame?

**Solution**

### Short Answer

**No, this would NOT work for WEP and would actually make security worse.**

While it might seem attractive to save the 24-bit IV overhead, the fundamental differences between SSL/TLS and WEP make this approach unsuitable and dangerous.

### Understanding IV Chaining in SSL 3.0

In SSL 3.0 with CBC mode:

- First record: Uses explicit random IV
- Subsequent records: $\text{IV}_n = \text{Last ciphertext block}_{n-1}$
- Stream of records in a **single TCP session**
- **Ordered, reliable delivery** guaranteed by TCP

$$C_0 = E_K(P_0 \oplus IV_{\text{random}}) \tag{2}$$

$$C_i = E_K(P_i \oplus C_{i-1}) \quad \text{for } i > 0 \tag{3}$$

### Why This Fails for WEP

#### 1. Unreliable Transport (UDP/Wireless)

**Problem:** WEP operates at Layer 2 (MAC layer), not over reliable TCP.

- **Packets can be lost:** Wireless is inherently lossy
- **Packets can arrive out of order:** No guaranteed sequencing
- **No retransmission at MAC layer:** Upper layers handle reliability

**Consequence:**

$$\text{If frame } n \text{ is lost} \Rightarrow \text{Cannot decrypt frame } n+1, n+2, \ldots \tag{4}$$

All subsequent frames become undecryptable until resynchronization!

#### 2. No Session Context

WEP has no concept of sessions:

- Each frame is independent
- Multiple stations communicate simultaneously
- Frames from different stations interleaved
- Broadcasts and multicasts mixed with unicast

**Question:** Which "previous ciphertext" should be used?

- Previous frame from *this* station?
- Previous frame to *this* destination?
- Previous frame on *this* channel?

There's no clear answer without complex state management.

**3. Initial Synchronization Problem**

**Problem:** How do sender and receiver agree on the initial IV?
In SSL 3.0:

- Explicit IV sent in first record
- Subsequent records chain from there
- Handshake establishes session context

In WEP:

- No handshake protocol
- Stations can join/leave anytime
- No way to signal "this is the first frame of a chain"
- **When station boots up, what IV does it use?**

### Problem 6.3 WEP: Decryption Dictionary

How can you find the correct keystream from a decryption dictionary for a fixed secret WEP key $k = K[3]|K[4]|...$ to decrypt an intercepted WEP frame?

### Solution

Since the WEP key is fixed, the different keystreams can be distinguished solely based on the IV. The descryption dictionary consists of pairs (IV,keystream), which we may sort according to IV. So to find the correct keystream, we only have to search for the IV of the actual packet in our dictionary.

### Problem 6.4 WPA: 4-Way Handshake

What happens if one of the four messages of the 4-way handshake gets lost?

### Solution

**What happens:** The handshake uses a **timeout and retransmission mechanism**.

- Each message (except Message 4) expects a response
- If response not received within timeout: retransmit
- Maximum retry attempts (typically 3-5)
- If all retries fail: handshake aborted, association may be terminated

### Problem 6.5 WPA2: KRACK Attack

In AES-CCMP, a strong cryptographic checksum is used to protect the integrity of the plaintext. Is it nevertheless possible to decrypt a ciphertext if a known plaintext/ciphertext pair is given for the same combination TK/FC?

**Solution**

Remark: The strong cryptographic checksum does not matter here, since we only want to decrypt a message, not produce a fake one.
Only vulnerability:Nonce reuse (implementation bug).
In RC4, the keystream only depends on the static key $k$ and the known IV.
In AES-CCMP, the keystream depends on the static key $k$, the nonce $n$, and a counter $ctr$. The value $ctr$ is used to produce an arbitrarily long keystream; we start with $ctr = 0$, and increment it. For each value of $ctr$, 128 bit of keystream is produced.
Since $ctr$ alsways starts with 0, we need another value to make sure that all keystreams are different. This is the role of the nonce $n$. So $n$ has the same role as IV in RC4, but it is much longer. Using some deterministic construction, cipher modes like AES-CTR (part of AES-CCMP) make sure that the nonce never repeats itself.
So without Nonce reuse, knowing a plaintext/ciphertext pair does not help.

**Problem 6.6 WPA3: Dragonfly Handshake**

Is it possible to use one of the two values $mac_A$ or $mac_B$ to conduct an offline dictionary attack on the password $pw$?

**Solution**

Observation: $mac_B$ cannot be used in an offline dictionary attack, since it is only received after the attacker has sent a valid MAC $mac_A$. So let's concentrate on $mac_A$ and assume that the attacker is in the role of responder.
The attacker (acting as B in Figure 6.13) can perform all computations up to the reception of $mac_A$, but he has to guess and use a specific password $pw_0$ to do so. He commits to this value $pw_0$ when he sends $(s_B, E_B)$, and this value will be used by A to compute all keys. So the MAC key used by A depends on the 'real' password $pw$ and the guessed password $pw_0$, and $mac_A$ can only be used to check if $pw_0 = pw$. It cannot be used to check against other password candidates like $pw_1$, since $pw_0$ was influencing key generation.