Chapter 12

Wireless LAN Security: 802.11 and Wi-Fi

The official version of this book is available at
https://www.springer.com/gp/book/9783030834104

ISBN: 978-3-030-83410-4 (hardcopy), 978-3-030-83411-1 (eBook)

version: 29 June 2021

Chapter 12

Wireless LAN Security: 802.11 and Wi-Fi

This chapter considers wireless local area network (WLAN) security. The focus is WLANs based on the IEEE 802.11 standard, and related subsets marketed under the *Wi-Fi* brand by an industry association to facilitate product interoperability. This provides a rich opportunity to explore and analyze the security implications of specific design choices actually made in a major fielded system—avoiding any claims that "no one would ever have made those choices in practice" and long lists of "what if" or "suppose that" alternatives. Wireless access introduces both new security challenges, and an opportunity to apply standard security tools and mechanisms to wireless LANs as a particular use case. Our study will also illustrate that building real-life systems involves a large number of low-level security-relevant decisions—and that design mistakes are made even by international committees, and in widely deployed products from large multi-national corporations.

12.1 Background: 802.11 WLAN architecture and overview

Ethernet is a dominant family of networking technologies for wired LANs (local area networks); the standard governing it is IEEE 802.3. Loosely speaking, the analogous technology and standard for WLANs are Wi-Fi and IEEE 802.11. From a network stack viewpoint (Chapter 10), both Wi-Fi and Ethernet provide an interface insulating higher layers from the implementation details of the data link and physical layers.

Mobile devices such as laptop computers commonly access network resources by connecting to an *access point* over a wireless medium in specific radio frequency (RF) ranges. The access point itself is typically connected to a physically wired local network, thereby providing Internet connectivity. Aside from connecting to a cellular network, many mobile phones can connect by WLAN to an access point—to access Internet services more cost-effectively (if mobile phone cellular service is billed on a usage basis), or more reliably if cellular reception is poor (e.g., indoors).

PRIMARY 802.11 COMPONENTS. The main components of a WLAN (Fig. 12.1) are the wireless endpoint or *mobile station* (STA), *access point* (AP), and *authentication server* (AS). An AP creates logical connections between STA devices and a *distribution system*

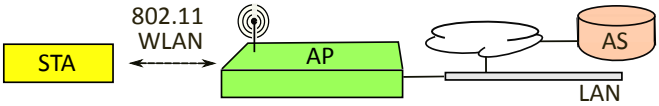

Figure 12.1: Three main components of a WLAN: mobile station (STA), access point (AP), and authentication server (AS). Using the 802.1X model, the AP includes an authenticator, which communicates with the AS (often known as a RADIUS server).

(DS), usually wired to the rest of the network, providing forwarding and routing services. The AS may be built into the AP in simple designs (e.g., for home networks), often using no more than a username-password list (we discuss more advanced authentication later).

INFRASTRUCTURE MODE. Our main focus and the common use of an 802.11 WLAN is what is called *infrastructure mode*. This involves one or more APs. An AP establishes a wireless connection with a STA, logically connecting it to the DS (or perhaps to another STA). The AP's *authenticator* sub-component allows the STA to access the DS via an *access control* mechanism (shown later in Fig. 12.4) relying on an approval decision from the AS. An AP plus one or more STAs in this mode is called a *basic service set* (BSS); two or more BSSs connected by a DS form an *extended service set* (ESS).

‡**AD HOC MODE.** A different 802.11 network configuration—now without any AP—is *ad hoc mode*. Endpoints (STAs) connect directly to each other, forming what is called an *independent basic service set* (IBSS), with no connection to a DS or other network.

FRAME TYPES IN 802.11. IEEE 802.11 specifies three types of frames (Fig. 12.2):

1. *data frames* (highest level). Satisfying the end-goal of WLANs, these carry upper-layer protocol data for the far-end destination, and also (upper-layer) authentication messages. Early 802.11 protection mechanisms focused largely on data frames.
2. *management frames*. These include messages related to beacons, probes, associations and related (low-level non-cryptographic) authentication. They support new STA-AP connections, and handovers between APs as mobile devices move into/out of range.
3. *control frames* (lowest level). These frames are related to accessing the wireless media, and facilitate management frames and data frames. We say little more about them.

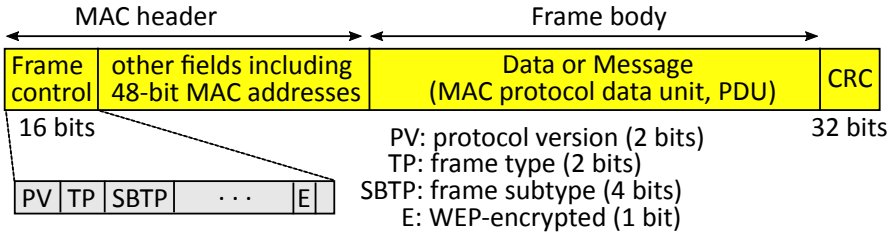

Figure 12.2: 802.11 MAC frame format. Frame type (TP) may be: control, management, or data frame. MAC frames are the payload of the physical layer, whose own header information (not shown) includes a Physical Layer Convergence Protocol (PLCP) *preamble* allowing recognition of 802.11 frames, and other information such as an explicit byte count or transmission period in μs specifying the payload length before the trailing CRC.

Radio spectrum *bands* allocated for Wi-Fi (e.g., contiguous frequency ranges around 2.4 GHz and 5 GHz) are subdivided into narrower ranges called *channels*. A channel is selected by or configured into the AP.

SSIDs ARE NOT SECRETS. A *service set identifier* (SSID) is an alphanumeric WLAN name, up to 32 characters. This is distinct from the *BSSID* (ID for a BSS), an AP's 48-bit MAC address. An SSID may also serve as *ESSID* (ID for an ESS). Some early 802.11 systems treated SSIDs as secrets, allowing a device to access an AP if it knew the SSID; this fails as an access control mechanism, because SSIDs are visible as plaintext in management frames. A rogue AP may spoof an SSID simply by asserting the text string.

MULTICAST ADDRESSES AND INFRASTRUCTURE MODE. An 802.11 frame includes 48-bit source and destination *media access control* (MAC) addresses, user data, and a *cyclic redundancy code* (CRC) checksum designed to detect transmission errors. A *unicast* destination address specifies a single recipient; a *multicast* destination address specifies a group of recipients, with special case of a designated *broadcast address* denoting all LAN devices. In 802.11 infrastructure mode, the AP may send multicast messages; a STA wishing to do so communicates this to the AP, which can send the multicast on the STA's behalf. Our interest is in WLAN security (not addressing), but broadcast messages are important in security: they are encrypted under a group key shared by all local (WLAN) devices, and group key management (page 366) is important in the security architecture.

ASSOCIATION, BEACONS AND PROBES. For a wireless device (STA) to use an AP in its signal range, a (layer 2) link-level connection or *association* must first be set up. The steps for a STA to associate with an AP are as follows (as summarized in Table 12.1).

1. The STA sends *probe* messages soliciting information from all APs within range. APs already advertise their presence by transmitting periodic *beacon* frames (e.g., every 100 ms) with network information including SSID and supported capabilities (e.g., data rates, security options), and to synchronize network timing. Probes elicit such information sooner.
2. The STA selects an AP with suitable capabilities, and begins a low-level authentication sequence using management frames. Two options were provided in the original 802.11 standard (1999): *Shared Key* and *Open System*. Shared Key authentication is now deprecated.[1] Open System (also called *null authentication*) omits actual authentication at this stage—instead, in a *pro forma* exchange, the STA simply sends an authenticate-request frame, and the AP returns an authenticate-response (accepted) frame. Stepping through this apparent bit of nonsense moves the formal protocol state to "802.11-authenticated", as needed for a STA to proceed with association, next.
3. The STA begins an association-request sequence, ending in the 802.11 state *associated*, provided that the capabilities (including security) of the AP and STA are compatible. Depending on configured criteria, the STA might automatically associate with an AP from a pre-specified list, from among APs previously used, by strongest AP signal, or the user may be prompted to make a manual selection from a list of advertised SSIDs.

[1] It is worse (page 353) than no authentication at all; a student might suspect that it was designed by an attacker. Further core 802.11 authentication options added in later revisions include SAE (page 368).

4. Finally, for an AP to accept data frames, an associated STA must succeed in an upper-layer 802.1X authentication method with the *authenticator* (explained later, Fig. 12.4).

New action	State	802.11 authenticated	802.11 associated	802.1X authenticated
Start state (fresh start)	0	☐	☐	☐
Low-level authentication succeeded	1	☑	☐	☐
Association succeeded	2	☑	☑	☐
802.1X authentication succeeded	3	☑	☑	☑ †

Table 12.1: 802.11 connection state transitions. Upper-layer 802.1X authentication of state 3 is distinct from the 802.11 MAC layer (e.g., null) authentication of state 1. †*Open* networks (without encryption, authentication) allow network access without this stage.

AP SECURITY POLICY. An `Information Element` field in beacons and probe responses specifies the AP's *security policy* and available options, e.g., whether the AP uses an external AS or pre-shared keys (Section 12.3), and supported security algorithms (Section 12.8). The STA's association request must choose an offered security suite for each of broadcast (group) and unicast security, determining what will be used for this connection.

12.2 WLAN threats and mitigations

With this basic understanding of WLANs as context, we now consider wireless LAN threats and mitigations. Our interest is in generic issues and recurring error patterns; but to make clear that the concerns raised are not purely academic, we use examples of exploited vulnerabilities in fielded 802.11 products. We discuss some flaws even where later designs eliminate them, because the threats themselves do not disappear once one standards group or product team gains awareness—our community advances when we retain knowledge and make it accessible to newcomers joining the field, otherwise unaware of painful historical lessons. Case studies help us avoid repeating past mistakes. In later sections (12.4–12.6) we examine more specific choices made in the initial design of 802.11, to get a deeper sense of what can go wrong when security systems are designed by non-experts. To be clear, we do not pursue a security analysis of recently fielded commercial systems, nor a training course for technicians to configure specific products, but rather aim to understand fundamental security issues in the design of wireless systems.

WIRELESS SECURITY: LINK VS. END-TO-END. A common wish related to wireless networks is that they be as secure as wired networks. However, because the wireless scenario introduces new risks (as we will discuss), achieving this wish requires significant extra effort. It should also be noted up front that a WLAN itself is foremost about the (layer 2) link between a device and an access point, not end-to-end communications. Thus when an AP decrypts and forwards received data, any protection associated with the wireless link is removed as the link ends (at the AP), leaving plaintext data—the default on

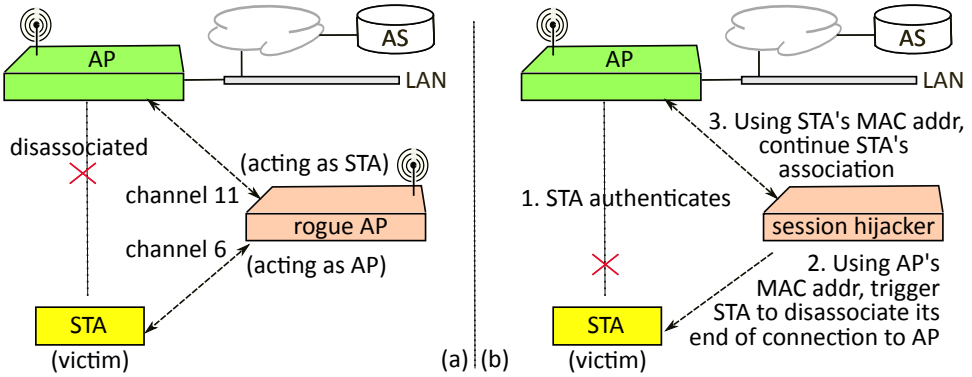

Figure 12.3: (a) Rogue AP or WLAN middle-person attack. (b) WLAN session hijacking. These attacks were common when early 802.11 became popular. Both attacks assert false MAC addresses. In (a), the rogue AP could choose to connect to a real AP, LAN or far-end server; using different channels facilitates separate connections.

traditional wired networks—unless a protocol above the link layer has secured the data.

ROGUE AP AND SESSION HIJACKING ATTACKS. To illustrate wireless risks, we begin with rogue AP (middle-person) and wireless session hijacking attacks, which plagued early 802.11 products; publicly available tools automated the attacks, allowing easy exploitation of weakly configured products. The attacks are easier than their wired counterparts, as they require no physical access to the target devices or network wires.

1. *Rogue* AP. This attack (Fig. 12.3a) is possible when there is no mutual authentication of the AP to the STA. Without this, a STA has no assurance who is offering service, and relies on an SSID that is simply asserted (a character string easily copied); any MAC address used by a legitimate AP can also simply be asserted by an imposter. First consider the case of no upper-layer authentication at all. The attacker (rogue AP) waits to see a connection attempt from the STA, then offers service, asserting the true AP's MAC address; if the STA was already associated, the attacker can send it a *disassociate* frame to trigger a new connection attempt.[2] The attacker thereafter relays messages from the STA to the true AP, asserting the STA's MAC address to the AP. Second, for the case of unilateral 802.1X authentication of the STA to the AP: as the STA carries out its end of the authentication protocol, the rogue AP (without doing any verification whatsoever) simply sends the protocol message(s) indicating authentication success. The rogue AP then provides expected network connectivity (assigning an IP address and configuration parameters).

   The above assumes that data frames are *not* protected by session encryption and integrity. Suppose they are, and now also mutual authentication is used. A rogue AP offering connectivity to a STA may still be able to carry out a middle-person attack, simply relaying (unaltered authentication messages and thereafter) encrypted frames to a legitimate AP—but the rogue gains little by doing so, lacking keys to access plaintext. It could drop frames

[2] Integrity protection for some management frames including *disassociate* and *deauthenticate* frames was introduced in 802.11w–2009 as an option. This became a requirement for WPA3 certification (Section 12.8).

(inflicting *denial of service*), or alter frames on the fly (this will be detected), but otherwise may just as well passively eavesdrop on the ciphertext. The mutual authentication method here is assumed to be done properly (establishing a key then using it to derive new keys for ongoing session encryption and integrity, as later discussed in Section 12.8).

2. *Session hijacking.* The attack of Fig. 12.3b is possible when no encryption is used. After a legitimate STA completes 802.1X authentication with an AP, the attacker sends the STA a *disassociate* management frame, asserting the AP's MAC address. This triggers the STA to disassociate, but the AP still believes it is associated (with 802.1X authentication state intact). The AP remains ready to respond "normally", continuing to exchange session messages with the attacker, who falsely asserts the MAC address of the STA (victim). Encryption stops an attacker, not knowing the key, from encrypting and decrypting.

‡**Exercise** (Session hijacking). (a) For historical dial-up sessions (page 370), hijacking was a lower risk than for 802.11 connections—why? (Hint: [11, p.146]; a dial-up password was the protection means.) (b) With 802.11, after the initial 802.1X authentication, how is ongoing protection provided? (Hint: wait for Section 12.3.)

WAR DRIVING (OVERVIEW). *War driving* refers to scanning radio channels in search of in-range wireless networks.[3] A narrow goal is to find an open network (i.e., one that grants service without requiring a password) in order to later use its services, even if such use is not the owner's intent. A broader goal is reconnaissance (Chapter 11): building a map of access points offering service in a predefined area,[4] stereotypically from a moving vehicle carrying a laptop with suitable software, GPS unit to record physical locations, and high-gain antenna. Omni-directional antennas are useful for mapping neighborhoods, while directional antennas suit specific targets or long ranges (e.g., rather than 50-100 m, up to 1 km or more). While war driving itself does not make use of data services, active war driving (below) may send probes that trigger AP responses.

Example *(NetStumbler active scanning).* Circa 2004, the free (Windows-based) NetStumbler tool was popular. It sent probes once per second, soliciting any SSID to respond, and many APs are configured to do so. These frequent *active probes* were "noisy" (easily detected), but nonetheless extracted metadata from APs such as hardware vendor, MAC address, SSID, wireless channel number, signal strength, and whether encryption was used—in which case further attack tools of the day, e.g., AirSnort, might be applied.

LEGALITY AND ETHICS. Definitions of and (severe) penalties for unauthorized network access vary across jurisdictions, and change over time. Therefore, as with other scanning and assessment activities (Chapter 11), best practice is to obtain written permission from the owners of target systems before embarking on any activities that may be interpreted as unethical or illegal in relevant jurisdictions. Passive reception of radio signals may be viewed differently than active broadcast or unauthorized use of data services.

WIRELESS MONITORING AND MANIPULATION TOOLS. Both security analysts and attackers may use powerful, widely available hardware and software tools for 802.11.

[3] The term derives from *wardialing*, the practice of dialing telephone numbers in an enumerative search pattern, seeking computer modems (modulator-demodulators) that will accept connections.

[4] This motivates the alternative term *access point mapping*.

Configurable drivers for wireless *network interface cards* (NICs) allow passive traffic monitoring (*wireless sniffing*). Active manipulation tools allow frames to be altered, replayed, and injected with arbitrary source MAC addresses. Recall that wired-LAN packet analysis tools in *promiscuous mode* (Chapter 11) can collect all packets passing the interface of a host's NIC. The wireless counterpart collects traffic for a single AP (or multiple-AP ESS) to which a wireless NIC is associated; here 100% of WLAN traffic is available for a suitably powerful listener. A broader alternative is to set a wireless NIC into *monitor mode* (*RFMON mode*), to passively collect all frames from all in-range SSIDs without associating to any. Here, ESSIDs (if included) can be extracted from beacon frames, and when ESSIDs are excluded from beacon frames, they may still appear in other frame headers sent by a transmitting STA.

WAR DRIVING (TACTICS). A passive war driving tactic uses a wireless NIC in RFMON mode (above), capturing all frames from one channel (frequency) at a time; *channel hopping* allows a specified set of channels to be monitored. The earlier NetStumbler example, an active tactic, sends repeated probes and collects beacon data from responses. A third tactic exploits the original 802.11 design feature of management frames not being authenticated: a *disassociate* or *deauthenticate* frame is sent, asserting the AP's MAC address (itself obtained by eavesdropping) as source. STAs then disassociate or deauthenticate, and attempt to reassociate—allowing capture of frames with SSID and other data.

‡**Exercise** (Kismet, Aircrack). Describe the technical capabilities of the modern open-source tools: (a) Kismet for passive 802.11 packet capture, analysis, and wireless monitoring; and (b) Aircrack-ng, promoted for assessing 802.11 network security. (c) Summarize Aircrack's circa-2004 functionality, including KoreK's *chop-chop attack* (hint: [24]).

WIRELESS DOS. Denial of service (DoS) is another threat to take into account, wherever possible, in the design of wireless protocols. However, it is generally acknowledged that in commercial (non-military) networks, (1) determined attackers can find ways to disrupt wireless channels even without network access, e.g., by radio jamming; and (2) often such attacks are easily detected, as their nature attracts attention. Stopping 100% of such DoS attacks is thus unrealistic—too costly, if even possible. For related reasons, some experts consider wireless DoS to be a *service* attack, not a security attack. Integrity protection of management frames (footnote page 344) is not possible for frames sent prior to establishment of session keying material; and if this protection is not enabled, as part of a nuisance DoS or session hijacking attempt an eavesdropper observing traffic can inject a *disassociate* or *deauthenticate* frame, to an AP or STA, to terminate an association (spoofing necessary fields such as MAC address simply by assertion). For greater effect, such a frame sent to an AP's broadcast address may trigger all STAs to disassociate.

LOSS OF PHYSICAL BASIS FOR THREAT MODELS. Wireless connectivity alters wired-network threat models and basic assumptions, and who we trust. In wired networks, in many cases we have confidence that service is provided by a trustworthy network (though not always in airport hotels or foreign countries)—whereas in wireless, rogue APs are a constant worry. As another example, in wireless networks such as Wi-Fi hotspots in coffee shops, where one master key is shared by all WLAN users, fellow users

can decrypt or modify your data;[5] note that in wired corporate or home networks, we more commonly assume (perhaps incorrectly) that our colleagues are not our opponents, and that local network devices are to some extent trustworthy. With wireless networks, attackers need not be physically on your premises to be in-range of radio service—we thus lose physically based approaches for access control and trust, perimeter-based mechanisms, and approaches based on physically restricted zones; a division between insiders (trusted, all on the same team) and outsiders (untrusted) is no longer reliable.

NEW AND HEIGHTENED RISKS. Advantages of wireless connectivity at broadband speeds include convenience, and time and cost savings related to avoiding wiring. For example, 802.11b and .11g already respectively offered, in 1999 and 2003, up to 11 and 54 Mbps (page 365). However, security risks also increase with wireless for reasons including widely available tools allowing easy interception, manipulation and injection of traffic (page 345). Table 12.2 summarizes attacks that are typically easier on wireless than wired systems.

Notable attacks on wireless systems	Mitigation (or notes)
A1. war driving (reconnaissance)	hard to stop collection of metadata
A2. passive eavesdropping (on data content)	encryption (payload data)
A3. unauthorized access to network services	entity authentication (authorization)
A4. modification of data content	data integrity mechanism
A5. injection of messages, including replay	origin authentication; replay defenses
A6. rogue AP (wireless middle-person)	authentication of the AP (mutual auth.)
A7. session hijacking (802.11 associations)	authenticated encryption (payload data)
A8. wireless DoS	integrity-protected management frames

Table 12.2: Attacks of concern in wireless systems. With wireless systems, physical access is not needed to read or inject frames. Replay of WLAN data or management frames is viewed as a type of message injection.

12.3 Security architecture: access control, EAP and RADIUS

The preliminary low-level authentication (Section 12.2) was intended as a layer 2 access control mechanism. We now discuss access control in 802.11 further, leading to upper-layer authentication protocols and the supporting frameworks of EAP and RADIUS. Recall again that WLAN security protects data over the local link, here between STA and AP; any link-layer protection disappears after the link's far end (protection is not end-to-end).

SECURITY FRAMEWORK AND STAGES. We view 802.11 WLAN security in four pieces. Here we provide an overview of the first two, while later sections discuss the latter two—including the four-pass handshake, which derives the session keys.

1. *802.1X management and authentication framework.* This includes EAP and RADIUS,

[5] As we will see (page 367), this is a general concern when using shared keys across a group of users.

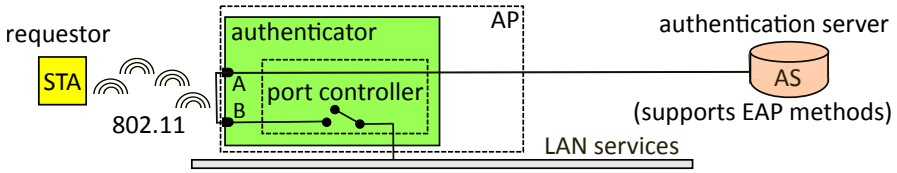

Figure 12.4: 802.1X dual-port access control model as used in a WLAN. An optional EAP method between STA and AS generates a PMK, which is transferred from AS to AP (and known by STA). Upon success of a PMK-based handshake between STA and AP, the *authenticator* will enable port B. Before then, authentication-related traffic (formatted in EAP messages) is all that is accepted from STA, for relay to AS (as EAP over RADIUS).

accommodating a wide variety of upper-layer authentication methods; and 802.1X access control, to condition authorization of network services on authentication success.

2. *Upper-layer authentication.* Organizations desiring a centralized decision point use an external *authentication server* (AS); simpler systems build modest AS functionality into the authenticator in each AP (Fig. 12.4). An upper-layer authentication protocol (EAP *method*) between STA and AS establishes a *pairwise master key* (PMK) as output. This PMK is transferred from AS to the in-AP authenticator for use in a handshake (next). When there is no external AS, authentication relies on a static *pre-shared key* (PSK, page 350) known to both STA and AP, and used to derive a PMK in this case.
3. *Four-pass handshake.* Using this protocol, STA and AP each prove to the other that they know the PMK. In addition, parameters are exchanged to allow derivation of fresh (not replayed) session keys from the PMK. Details are given in Section 12.8.
4. *Per-frame data encryption and authentication (integrity) on the* STA–AP *link.* Each frame can by design be decrypted independently of others, accommodating frame loss.

Note the pattern: a connection setup phase (security association), including negotiation of security parameters and authenticated key establishment, provides keying material (session keys) for ongoing protection (secure data transfer) of the ensuing session. This should look familiar from TLS and SSH session establishment in earlier chapters.

DUAL-PORT ACCESS CONTROL. IEEE 802.1X outlines a model called *port-based access control*.[6] For each associated STA, an *authenticator* in the AP serves to:

1. support establishment of a pairwise master key (PMK) for use between STA and AP (the AP receives the PMK from the AS, when an AS is used); and
2. control network access, by requiring a successful STA-AP handshake before granting a STA network services beyond the AP.

For each STA that associates with it, the AP allocates two virtual ports: one *controlled*, one *uncontrolled* (Fig. 12.4). The uncontrolled port (A) is always enabled, but limited: it allows only authentication-related messages to be relayed between STA and AS. The controlled port (B) begins disabled, preventing exchange of data frames with the rest of the

[6] This 802.1X model is also used for connections to wired Ethernet ports, where LAN services would be granted. In both Ethernet and WLAN cases, the idea is that only authorized users are granted network access.

network. All authentication methods use the following general pattern of messages, encapsulated in EAP messages (Fig. 12.5). STA sends to AP a message including an identity assertion with optional supporting evidence. This is relayed from AP to AS. Additional STA–AS messages are exchanged as needed. A successful outcome (`EAP-Success`), when seen by the authenticator, triggers it to begin the next phase, a four-pass handshake between STA and AP (page 365). First however, the PMK (needed in the handshake) is securely sent from AS to AP. If the handshake succeeds (confirming that STA and AP share the PMK, and that this is the STA authenticated by the AS), the authenticator enables the controlled port (giving STA network access).

MAC ADDRESS ALLOWLISTS. Some early WLAN products used MAC address allowlists (e.g., in APs or databases accessible to them) as a proprietary link-layer access control method. A device could access an AP (only) if its asserted MAC address was listed—sometimes called *MAC filtering*. This came with inconveniences: adding new devices to allowlists before services could be used, and managing common lists across APs within a WLAN (increasingly problematic with scale). It added a modest level of security (vs. *Open* authentication, above), but only against amateur attacks—because asserting false MAC addresses is not hard, and passive observation of cleartext MACs in frames reveals allowlisted addresses. (While open-source 802.11 devices drivers today typically allow MAC addresses to be altered, early device drivers did not.)

‡**NETWORKING TERMINOLOGY.** The terminology "protocol A runs over B" (e.g., TCP over IP) may be understood from a network stack diagram (Fig. 12.5). What is higher in the stack runs "over" what is below it. In terms of packet encapsulation, the higher protocol is the payload of the lower—i.e., the message headers for the lower protocol encapsulate the upper. An alternative view is as a set of concentric pipes or tubes, the smallest in the center: the innermost tube is the top of the stack—fluid running in it is the final payload, corresponding to data in the highest-level protocol in the stack diagram.

ROLE OF EAP. The *Extensible Authentication Protocol* (EAP)[7] is a messaging framework playing several roles in 802.11 authentication (Fig. 12.5), on the STA–AP link and also on the AP–AS path if an AS is used. For an external AS, EAP supports a wide choice of authentication methods, and allows repackaging and relaying of protocol messages (independent of their specific details and message sequences) between STA and AS. This supports execution of any selected EAP method (Section 12.8). For example, for EAP-TLS, since STA and AS are not directly connected, the specific EAP-TLS protocol messages are relayed between STA and AS by custom EAP packaging in two legs. The STA–AP leg uses an encapsulation called EAP over LAN (EAPOL); the AP–AS leg uses EAP messages over RADIUS (i.e., EAP is the payload of RADIUS). The individual authentication protocol selected is relied on for security and authenticated key establishment; EAP provides transport. Once such a protocol succeeds, the AS transfers the PMK to the AP,[8] for use in the four-pass STA–AP handshake. If that succeeds, the authenticator (in the AP) finally enables network services for the STA, via the port controller (Fig. 12.4).

[7] Section 12.9 summarizes EAP's origin.

[8] The AP–AS channel must be secured by some means, beyond the scope of 802.11 and 802.1X.

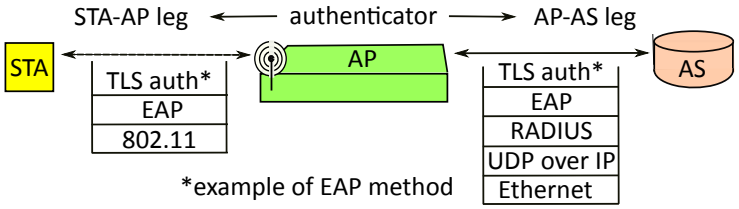

Figure 12.5: EAP and RADIUS roles in WLAN authentication. The *authenticator* function in the AP relays messages between the STA and AS in two legs. Network protocol sub-layers not shown are EAPOL (between EAP and 802.11), EAP-over-RADIUS and TLS-over-EAP; such sub-layers are often needed when refitting old protocols for new purposes.

‡**EAP AND EAPOL MESSAGES.** EAP facilitates specific authentication protocols by carrying their messages in generic EAP messages falling into two pairs {request, response} and {success, failure}, specified by a `code field`. A further EAP `type` field identifies the specific authentication protocol, with protocol-specific data included in a `type-data` field. Our primary interest in EAPOL (EAP over LAN), used over the layer 2 wireless link (Fig. 12.5), is to transport EAP messages using `EAPOL-packet` messages, and to carry messages for the four-pass handshake in `EAPOL-key` messages. Similarly to EAP, EAPOL itself is primarily a means for transport, rather than for security *per se*.

ROLE OF RADIUS. EAP messages are relayed from the AP-authenticator to the AS using a protocol called RADIUS (Remote Access Dial-In User Service).[9] Our other primary interest in RADIUS is its functionality that delivers a resulting PMK from AS to AP. RADIUS itself runs over UDP, which provides network-layer services (e.g., routing to the AS); note that network routing is not needed on the point-to-point STA-AP link. Four RADIUS messages map directly onto (and carry) corresponding EAP messages, again grouped in matched pairs: {access-request, access-challenge} and {access-accept, access-reject}.

PSK (PREDATING EAP). The early 802.11 design used an encryption design called *Wired Equivalent Privacy* (WEP). It did not include an AS, relying instead on a static PSK (no alternative). Here, a pre-shared secret from which a PSK is derived is configured manually into both mobile devices (STA) and the AP.[10] In theory the pre-shared secret may be a random bitstring; in practice, it is often a user-chosen password of ASCII characters. The PSK was originally limited to 40 bits due to US export restrictions, but many manufacturers supported 104 secret keys early on. The use of a centralized authentication server (AS) and automated key management supported by EAP and RADIUS came once the need became apparent (next paragraph).

WHY STUDY WEP. Static PSKs and lack of key management support were among a list of factors contributing to what turned out to be entirely inadequate security in the WEP design of the original 802.11 standard (1999). The security architecture was dramatically improved over 2002-2004 (with WPA and 802.11i, as detailed in Section 12.8). But we recommend study of WEP for two reasons, with Sections 12.4–12.6 exploring the (flawed)

[9] Section 12.9 gives background on the acronym RADIUS, also used for RADIUS servers (in our case, AS).
[10] The distinction between a user password and a PSK derived from it is discussed on page 367.

details of how it uses the RC4 stream cipher. First, knowing this history motivates the new design, and helps us understand it. Second, in security, we learn best from mistakes and attacks on earlier designs—and WEP is rich in this dimension.

12.4 RC4 stream cipher and its use in WEP

We first describe the RC4 stream cipher algorithm, and then its use in WEP.

RC4 BACKGROUND. RC4 uses a hidden state vector S that contains all values 0 through 255. During a key setup phase, the values are rearranged in a manner dependent on the secret key, and again somewhat as each keystream byte is output. The vector always contains the same values, in some (hidden) ordering. The intended effect is that each keystream output byte appears to be a "randomly" drawn value from [0..255], so the keystream approximates a random sequence of bytes to anyone not knowing the key. However, a party knowing the seed value can deterministically generate the same bytestream—this allows decryption. As the pseudo-code below illustrates, RC4 is compact, involves only simple byte operations, and is simple in appearance.

```
Function RC4doKeySetup(len,seed)      % initialization (RC4 key setup)
  (input) len:  byte-length of seed   % e.g., 5 <= len <= 32 (40-256 bits)
  (input) seed: secret key            % byte vector seed[0]..seed[len-1]

1  FOR (i := 0 to 255) { S[i] := i }      % initialize byte i to i
2  j := 0                                 % each value S[i] is 8 bits
3  FOR (i := 0 to 255) {
4     j := j + S[i] + seed[i mod len]     % j is also reduced modulo 256
5     swap byte values S[i] and S[j] }    % key randomizes hidden-state vector S
6  i := 0,  j := 0                        % reset indices for next phase
```

After this initialization, plaintext is encrypted a byte at a time, using the next byte of keystream output produced iteratively as follows. Here the persistent variables i and j, used to index the 256-byte state vector S, are reduced (mod 256), e.g., $255+1=0$.

```
Function RC4genNextKeystreamByte()    % pseudo-random byte generation

7  i := i + 1                         % unsigned byte addition (ignore carry bit)
8  j := j + S[i]                      % further randomize the hidden state
9  swap byte values S[i] and S[j]
10 t := S[ S[i] + S[j] ]
11 return(t)                          % t is next keystream byte
```

Thus if m_i and k_i are the next plaintext and keystream bytes (Fig. 12.6), the next ciphertext byte is $c_i = m_i \oplus k_i$. This is the usual Vernam stream cipher construction (Chapter 2). To decrypt, an identical sequence of keystream bytes k_i is generated, with each plaintext byte recovered simply by XOR'ing k_i to c_i: $m_i' = c_i \oplus k_i = m_i \oplus k_i \oplus k_i = m_i$ since $k_i \oplus k_i = 0$.

WEP USE OF RC4. In WEP, a symmetric key K is shared by the APs and STAs on a WLAN. In practice, K is often derived from a user-chosen password—and changed infrequently due to the inconvenience of coordinating manual update of all devices and

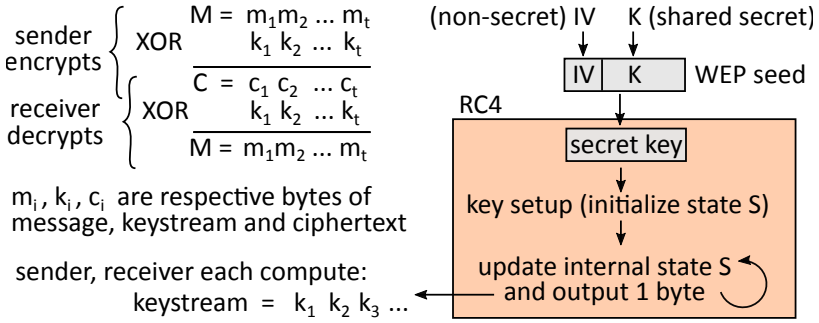

Figure 12.6: RC4 stream cipher as used in WEP. A cryptographer would warn against using a non-secret as a (non-mixed) part of a secret key, here the concatenated initialization vector (IV) in the seed. This was one of the serious problems in WEP's use of RC4, and the only practical method provided to alter the keystream across MAC frames.

APs. Let the 802.11 data frame payload data, or MAC *protocol data unit* (PDU) in Fig. 12.7, be denoted M. Then immediately prior to and as part of WEP encryption, two values are concatenated ("::") before and after it, yielding $IVD::M::ICV$. Here IVD is 4 bytes, not encrypted, and holds a 24-bit IV plus a fourth byte with a KeyID (e.g., supporting *group keys*, page 366). The *integrity-check value* (ICV) is a 32-bit CRC over the data. M::ICV is encrypted using the RC4 keystream (Fig. 12.6), with seed $s = \text{IV}::K$ as the secret key input. On receiving a WEP-encrypted frame, the receiver uses the cleartext IV to reconstruct the RC4 seed, regenerates the RC4 keystream to allow decryption, decrypts, recomputes the ICV from the decrypted PDU, and confirms the locally computed ICV matches the decrypted ICV. If the match fails, the frame data is not made available.

RC4 KEY SIZE. 802.11 formally specified a 40-bit WEP key K. This was to comply with later-abandoned US export rules in place to intentionally weaken security in exported products. Nonetheless, many early products already supported 104-bit keys K, resulting in WEP seeds (after extension by the 24-bit IV) being either 64 or 128 bits. The longer key increases protection against exhaustive key search (from an expected search over 0.5×2^{40} keys to 0.5×2^{104}, a factor of 2^{64}). However for many eventual attacks on WEP—including

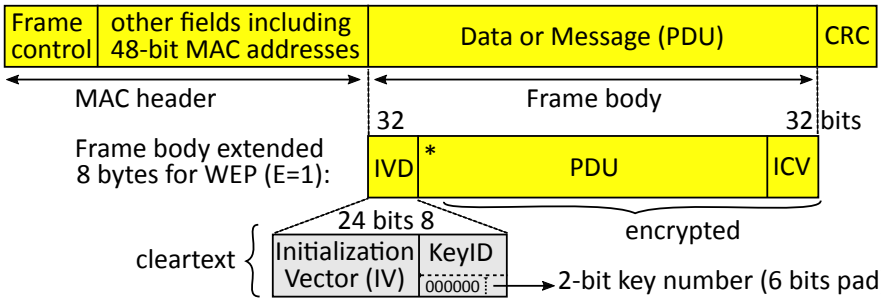

Figure 12.7: 802.11 format for WEP frames. WEP encapsulation (frame control bit E=1) is allowed only if TP = data frame, or TP = management frame and SBTP = authentication. The ICV in a WEP body is distinct from the frame's CRC. PDUs are at least one byte.

both the ICV failure and keystream stripping in Section 12.5—attack time is independent of keysize, while for others it increases only modestly or linearly in key length, e.g., by a factor of 2.6× in the *FMS weak key attack* (page 359).

12.5 WEP attacks: authentication, integrity, keystream reuse

We first describe a challenge-response protocol called *Shared Key* authentication in the 802.11–1999 standard, and an attack on it. For use prior to device association, this was the alternative to (null or) *Open System* authentication. It is a useful pedagogical example of how *not* to design a challenge-response protocol. Other failures discussed include WEP's data integrity mechanism, and design considerations related to IVs and keystreams.

SHARED KEY AUTHENTICATION. This protocol has four messages of fixed format. All field values are also fixed, other than the challenge (random number).

1. STA → AP: auth-request ..... AP=2, SN=1, Info Element empty in Fig. 12.8
2. STA ← AP: auth-challenge ..... M is sent as a plaintext challenge of 128 bytes
3. STA → AP: auth-response ..... C is returned as the encryption of M
4. STA ← AP: auth-status ..... `SC` is returned as the status code

`SC` is `success` if, on decryption, the ICV is correct and the decryption of C matches M.

KEYSTREAM STRIPPING (ON PROTOCOL ABOVE). The attacker observes the radio channel to see M and C in messages 2 and 3. This allows keystream recovery because: $M \oplus C = M \oplus (M \oplus S) = S$, the keystream. This should be clear from Fig. 12.6, where lowercase m_i and c_i denote individual bytes; here the corresponding uppercase letters denote 128-bytes strings. For a (fixed) shared key K, while this does not recover K explicitly, it recovers—by passive eavesdropping of the radio channel—the keystream (for the specific IV used). This keystream can then be used to compute a response for *any* later challenge; note that the choice of IV is left to the responder, and the attack simply reuses the IV. Since use of *Shared Key* authentication thus discloses a (reusable) keystream to attackers, it is

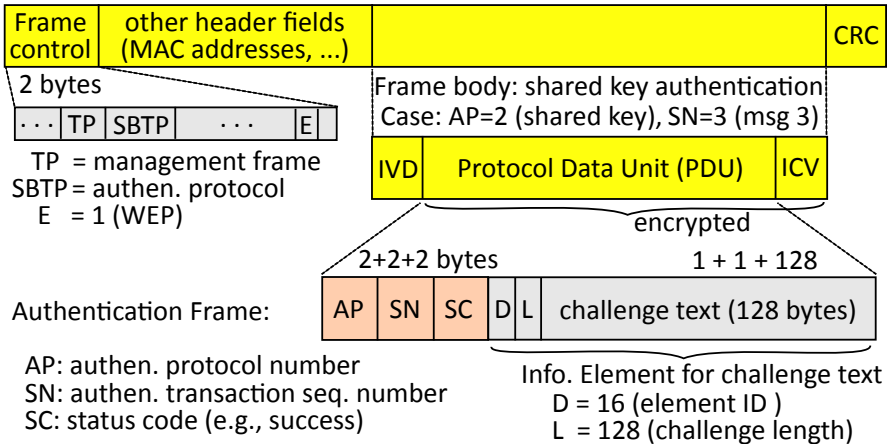

Figure 12.8: 802.11 frame for *Shared Key* authentication (message 3 of 4 messages).

viewed as worse than *Open System* authentication. This explains the otherwise confusing preference for *Open* authentication, and the deprecation of *Shared Key* authentication.

‡**COMMENT ON DESIGN FLAW.** The above attack is enabled by the 802.11 specification allowing, i.e., not preventing, deliberate IV reuse—which here implies keystream reuse. This flaw is not subtle—a basic rule for stream ciphers is that keystreams must not be reused. The consequent requirement is that key management be arranged such that keystream reuse cannot occur, or is extremely unlikely. (As Chapter 2 notes: if a keystream is truly random and not reused, the Vernam construction yields a *one-time pad* that is unbreakable with respect to confidentiality. In practice, a shortcut used to efficiently emulate a one-time pad is to use *pseudo-random* keystreams in place of truly random keystreams—but the essential rule remains: *never reuse keystreams*.)

As a second flaw here, in cryptographic protocol design it is widely recognized as unsafe to provide, or enable extraction of, plaintext-ciphertext pairs; for plaintext precisely as presented (without alteration), known-plaintext attacks become chosen-plaintext attacks. A common defensive heuristic in protocols is to reserve a plaintext field into which the responding party inserts an unpredictable value (*confounder*) of its own choosing.

‡**ADDITIONAL DETAIL ON ATTACK (AUTHENTICATION PROTOCOL).** The attacker knows that message 3 (Fig. 12.8) includes an encrypted 4-byte ICV immediately after the challenge text in the frame body. How is the keystream corresponding to this ICV itself recovered? An ICV does not appear in the cleartext message 2 (it is inserted only in encrypted frames). But this value is easily computed from the cleartext, and the known CRC algorithm. This allows recovery of the final 4 bytes of keystream (again simply by XOR), providing the full keystream needed to successfully forge the encrypted response for any later (different) challenge for fixed key K. Note that in such a response, the attacker reuses the same IV, in order for the same keystream to be valid; the attacker knows the necessary keystream, without knowing (or needing) the master key K itself.

KEYSTREAM REUSE: GENERAL CASE BEYOND WEP. The general issue of *keystream reuse* warrants further discussion, independent of WEP. Consider a byte-oriented Vernam stream cipher, with encryption by exclusive-or (XOR) of the keystream to the plaintext, and then again to the ciphertext to decrypt (Fig. 12.6). Suppose two messages are encrypted with the same keystream—what is the danger? If the two ciphertext streams are XOR'ed, the keystreams cancel out, leaving the XOR of the two plaintext strings: $P \oplus P'$. We mention two cases of interest.

Case 1: all or parts of P or P' are known. This then allows immediate recovery of the corresponding part of the other string by XOR (and then also recovery of the corresponding part of the keystream).[11] This case is not uncommon—many fixed-format messages, e.g., due to standard protocols, have constant fields, are known, or are easily predicted (e.g., a date field); plaintext P (corresponding to an encrypted message) may also be directly available from the source. *Case 2:* redundancy in the plaintext messages allows well-known statistical methods to recover the plaintext itself. As examples, ASCII-encoding

[11] As we have seen, in the case of 802.11 with WEP, reuse of a recovered keystream (for a fixed K and given IV) can be arranged by intentional reuse of the same IV, which is set by the frame originator.

provides redundancy in characters, and natural languages are highly redundant.

INTEGRITY MECHANISM FAILURE (OVERVIEW). While the overall 802.11 frame CRC detects transmission errors, WEP's encryption of the ICV (Fig. 12.7), a CRC based on message data but not any secret, was intended to detect malicious changes, i.e., for data integrity in the *data authentication* sense. However, it was already known (page 369) that encrypting an appended CRC fails at this. As we explain shortly, without knowing the secret keystream, the data can be altered undetectably by also making suitable changes to the ciphertext corresponding to the ICV. To gain an understanding of how the attack proceeds, we begin with an easy case: an attacker assumed to somehow know the underlying plaintext (i.e., a *known-plaintext* scenario). We will compute a string to XOR onto the ciphertext so as to insert any chosen message, with a correct (modified encrypted) ICV.

INTEGRITY MECHANISM FAILURE (DETAILS). Let $M = m_1 m_2 \cdots m_{t-1} m_t$ denote the string of bytes composing the MAC frame data (PDU of Fig. 12.7), and $M_r = f(M)$ the 32-bit ICV value from CRC function f. Let the $t+4$ corresponding keystream bytes (generated using RC4) be $S = k_1 k_2 \cdots k_{t-1} k_t$ with S_r denoting the last 4 bytes (to match notation). Then the WEP ciphertext for this frame can be denoted

$$C :: C_r \qquad \text{where } C = M \oplus S \qquad \text{and } C_r = M_r \oplus S_r \tag{12.1}$$

Let T be a substitute string (of identical length, for simplicity)[12] that an attacker wishes the recipient to recover on decryption (instead of M). The attacker will create a string D, then XOR D onto C to alter C. What value for D will work so that the intended recipient (unwittingly) recovers T rather than M? The value is simply: $D = M \oplus T$ (and critically, this requires knowing neither S nor S_r). The recipient now receives not C but $C \oplus D$, which is $C \oplus (M \oplus T) = (M \oplus S) \oplus (M \oplus T) = S \oplus T$. Now the usual WEP decryption XOR's S onto the received ciphertext, yielding $(S \oplus T) \oplus S = T$ as desired. This simple attack works because the result of XORing multiple items is unaffected by reordering terms.

However, we have yet to account for the ICV bytes, whose sole purpose is to detect changes, and trigger an error if the ICV check fails (page 352). What 4-byte value D_r should be XORed onto C_r, to avoid an ICV error? Analogously to the reasoning above, the value is: $D_r = f(M) \oplus f(T)$, i.e., $M_r \oplus T_r$, the XOR of CRC values from M and T. To see this, note that if this $D_r = M_r \oplus T_r$ is XOR'd onto the ICV bytes of the original ciphertext, since $C_r = M_r \oplus S_r$ from (12.1), the ICV bytes of the altered ciphertext become

$$C_r \oplus D_r = (M_r \oplus S_r) \oplus (M_r \oplus T_r) = S_r \oplus T_r \tag{12.2}$$

Now when the unsuspecting recipient uses the normally generated final keystream bytes S_r to XOR-decrypt the ICV, the result is T_r, which is $f(T)$, so the ICV check succeeds. We now recognize that the ciphertext alterations used to change the original $M :: f(M)$ to $T :: f(T)$ can be viewed as a simple XOR-off of $M :: f(M)$, and an XOR-on of $T :: f(T)$. Thus WEP's easily manipulated XOR-encryption fails also to protect the ICV value.

‡**FULL ATTACK EXPLOITING CRC USED AS ICV.** The attack as explained above relies on known-plaintext—to compute D and D_r requires knowing the (entire) original

[12] With only minor modifications to the attack, any T shorter than M could also be used.

plaintext M. However, the flaw is more serious due to the following generalized attack. An attacker not knowing *any* of the original plaintext M, can make *any* desired XOR-alteration D to M (perhaps informed by knowing parts of M or its underlying format), by easily computing a compensating value D_r to XOR onto the ICV position to avoid detection;[13] for any D, this is simply $D_r = f(D)$ where f is the CRC function used for the ICV. The following exercise explores supporting details.

‡**Exercise** (Full attack exploiting CRC linearity). (a) Explain why the generalized attack works, assuming the property: $f(M \oplus T) = f(M) \oplus f(T)$ (hint: [7]). (b) Explain how polynomials over binary finite fields of the form GF(2^t) are represented as bitstrings, and how in this representation, addition of polynomials is the same as bitwise XOR (hint: Example 2.231 [30, p.85]). (c) Look up and specify the degree-32 polynomial $G(x)$ used by the CRC function in question (hint: pages 40–41, 1999 version of [21]). (d) Using equations, explain how CRC values are represented as remainders of polynomial division, and explain how this and (b) support the property in (a) (hint: [30, p.363]).

‡**Exercise** (ICV requires keyed hash function). The known-plaintext ICV attack still works if the CRC function is replaced by any other function of the plaintext alone, including an (unkeyed) cryptographic hash function. Explain why.

‡**LACK OF INDEPENDENT MECHANISMS.** Note that this flawed ICV method also fails to detect fraudulent message insertion by an attacker holding a recovered encryption keystream—obtained, e.g., by attacking *Shared Key* authentication (page 353); any known-plaintext scenario also allows encryption keystream recovery. Thus rather than having two independent protection mechanisms, in the WEP design compromise of one (encryption) compromises the other (data authentication)—breaking a design principle. Also, from the viewpoint that WEP's data integrity mechanism was intended, or marketed, as a means to prevent use of the transmission channel by unauthorized users (whereas the usual goal of encryption is to prevent disclosure of confidential information), this data integrity failure may be viewed as an access control failure.

IVS IN WEP. The WEP encryption design specified the use of 24-bit initialization vectors (IVs), to vary the keystream across different frames—by the unorthodox means of simply concatenating an IV to a fixed shared master key K. It was widely known that reusing keystreams should be avoided, yet how to generate IVs was left as a homework assignment for manufacturers. As it turns out, regardless of the IV generation method, 24 bits was too few from the start. An understanding of why emerges from the next example, which explores: How long do we expect it takes before a 24-bit IV is repeated?

Example *(24-bit IV spaces).* For a network transmitting packets of a given (average) size at a given (average) rate, what time will elapse before an IV *collision* (two packets with the same IV)? The time is maximized if packets are assigned IV values in sequence, e.g., from 0 to $2^{24} - 1$. Thereafter, a collision is unavoidable. For illustration and ease of scaling, assume a network speed of 10 Mbps. This is close to the maximum 11 Mbps (raw data rate) from the 1999-ratified 802.11b amendment, but far less than new Wi-Fi

[13] For example, a D with a single 1-bit in one byte (other bytes all 0) will flip the corresponding bit in the recovered plaintext. This matches our Chapter 2 example on the one-time pad not providing data integrity.

networks which approach gigabits/s (Gbps). Assume also 1500-byte packets, to match the Ethernet MTU or *maximum transmission unit* (802.11 supports PDUs somewhat over 2300 bytes). Our model network then supports $(10^7 \text{ bits/s})/(1500 \times 8) \text{ bits/packet} = 833.33$ packets/s. Multiplying by 3600 s/hr yields 3 million packets/hr ($\approx 2^{21.52}$). To exhaust 2^{24} distinct IVs then takes $2^{24}/(3 \cdot 10^6) = 5.59$ hours. This time will decrease for a faster network that is fully loaded (or has smaller frames), e.g., to a few minutes for Gbps networks; and will increase for a slower network or one with lower traffic load.

INEVITABLE IV COLLISIONS. In summary, even for a slow model network, IV collisions occur in under a day (while the ideal period is never). The actual situation is worse: in practice, with more than one WLAN device transmitting, the packets of all devices sharing a fixed key K draw on the same IV space at once, without coordination. If IVs are selected randomly, collisions are expected well before 2^{24} frames—rather, on the order of 2^{12}, due to the *birthday paradox* (Chapter 2), which tells us to expect collisions in time proportional to the square root of the event space size—here, a reduction factor of 4000. As another reality, wireless network cards may reinitialize when systems reboot (e.g., as laptops are powered on), with common or predictable IV reset values and sequencing algorithms both within and across manufacturers. The obvious engineering choice of resetting IVs to 0, and incrementing sequentially, leads to an expectation of many collisions among IVs of low numerical value. Colliding 24-bit IVs in WEP are thus common.

We next summarize WEP security properties and key recovery, the strongest attack.

12.6 WEP security summary and full key recovery

Here we first highlight selected WEP design properties. Many of these are now recognized to be poor choices, enabling major security failures, which we review in turn. We also discuss in some detail the ultimate attack: recovery of the master key. We begin with six observations about WEP's design; notably, all involve key management.

D1: *IVs (in cleartext, as typically required) immediately precede* WEP*-encrypted frames.* While not a flaw itself, this allows immediate identification and matching of reused keystreams, and lookup of keystreams previously collected or compiled in *keystream dictionaries*.

D2: *The same key is used for three cryptographic mechanisms*—confidentiality, authentication (*Shared Key*), and data authentication (encrypted CRC). In general, such dependence admits new risks—a failure in one mechanism may compromise the others, and some attacks may exploit relationships between the mechanisms.

D3: *IVs are 24 bits and selected entirely at the option and control of the frame originator, with IV reuse allowed.* As consequences, in practice: IV reuse is unavoidable (due to the bitlength), standard-compliant receivers must accept all IVs (including those intentionally reused), and keystream reuse is unavoidable (in the common case of static master keys).

D4: *The concatenation of the IV and master key K is used directly as the* RC4 *seed.* Later designs would use hash functions to mix auxiliary parameters with a master key, to derive sets of secondary keys that differ from each other and whose reuse cannot be forced.

D5: *In practice, across a* WLAN*'s devices and* APs*, many users share a master key.* As consequences: (a) users can derive the frame keys protecting other users' data; (b) those sharing a master key all draw on the same IV space (increasing the number of unintentional IV collisions—resulting in reused keystreams); and (c) the master key is an attractive target.

D6: *The master key is often a human-chosen or memorable string.* Such keys are at risk to simple password-guessing attacks—even in implementations configured for 104-bit secret keys; meanwhile, the original 40-bit keys are entirely insufficient for serious protection.

Note: D5 and D6 are not design choices *per se*, but consequences of 802.11 declaring management and selection of keys to be beyond its scope. For D5, shared master keys can be avoided by infrastructure or policies providing distinct users with distinct master keys, but both are typically hard to implement or enforce in non-corporate environments.

‡**SECURITY DESIGN PRINCIPLES.** WEP design choice D2 fails to provide independence between cryptographic keys, in conflict with design principles P7 (MODULAR-DESIGN) and P13 (DEFENSE-IN-DEPTH). WEP encryption being disabled by default, per the specification, breaks P2 (SAFE-DEFAULTS). The WEP design was open to public scrutiny only after it was finalized, breaking principle P3 (OPEN-DESIGN).

FAILURES RELATED TO DESIGN CHOICES. We now list some of the most serious consequences of the preceding design choices combined with how WEP used RC4.

F1: *The Shared Key authentication method had severe flaws* (802.11 mode 2, now deprecated). Passive recording of one exchange not only allows subsequent fraudulent authentication, but immediately provides a secret keystream that can be reused by an unauthorized party to inject arbitrary (properly encrypted) 128-byte messages. This is contrary to expectations that only legitimate devices knowing the master secret can send encrypted traffic on the WLAN. This flaw is independent of the master secret being 40 or 104 bits.

F2: *The data authentication mechanism was easy to break* (encrypted CRC as ICV). Using the ICV mechanism attack (page 355), an eavesdropper can copy a legitimate message and inject an undetectably altered version without knowing the master secret.[14] In general, data authentication mechanisms should serve to both detect alteration and confirm message authenticity; here, the ICV attack allows undetected alteration, while keystream reuse enables injection of new forged messages, more directly defeating message authenticity. This flaw is likewise independent of the master secret being 40 or 104 bits.

F3: *Confidentiality intended by* RC4 *encryption was compromised* by a combination of design choices enabling identification and reuse of keystreams, including attacks that need not explicitly recover the master secret. Contributing factors here were design choices D1 and D3 (short, visible IVs) and a design encouraging shared static master keys (D5), combined with simple bytewise XOR-encryption. Keystream-type attacks are generally unimpeded by 104-bit keys, while 40-bit keys offer no serious protection.

F4: *Explicit recovery of the master secret is possible* by both guessing attacks (when weak) and (even for strong, 104-bit keys) by an efficient attack requiring only passive observation

[14] On-the-fly data alteration is a more sophisticated attack, but possible in conjunction with a middle-person attack (Fig. 12.3), itself possible because of the absence of a strong mutual authentication protocol in WEP.

of traffic with little computation. In both cases, master key compromise naturally results in total security failure of all mechanisms. We discuss these in turn.

WEP KEY RECOVERY: GUESSING ATTACKS. In practice, if a WEP master secret, whether 40 or 104 bits, originates from a human-chosen password, then password-guessing *dictionary attack* strategies often succeed (Chapter 4). For the 40-bit case, exhaustive guessing through the keyspace is easy even for truly random strings. (The next exercise pursues user-chosen passwords in general; it may be deferred until Section 12.8.)

‡**Exercise** (*Offline dictionary attack* on PSK). Explain how recording the first two messages of the four-pass handshake (page 365) allows an attack recovering the PSK for weak user-chosen passwords. (Hint: page 363, equation (12.4); or [32], [24].)

WEP KEY RECOVERY: FMS ATTACK. In 2001, Fluhrer, Mantin, and Shamir (FMS) described a method to recover WEP seed keys, thus fully breaking WEP. They found specific sets of IVs (*weak IVs*) that manipulate RC4's key setup such that the first keystream byte leaks information on one byte of the key; gathering this over enough IVs reveals the byte. A different set of IVs then reveals the next key byte, and so on. The attack relies on IVs being visible (D1) and directly used as part of the seed (D4) used for key setup. A master key shared across users (D5) makes collection of WLAN traffic both faster and simpler (as all frames correspond to a fixed base key with different IVs). The attack relies on seeing the first keystream byte for each IV used. While ciphertext reveals only the XOR of keystream with plaintext, WEP provides a known-plaintext scenario accommodating this requirement: the first plaintext byte of WEP-encrypted frames is the constant `0xAA`.[15] Thus an XOR of `0xAA` to the first encrypted payload byte yields the keystream byte as required.

The attack is iterative. At each step, given knowledge of the first i bytes of the RC4 key (`seed`), it finds the next byte. For a WEP key of 40 bits (8 bytes), bytes 0–2 of a seed $(IV_0, IV_1, IV_2)::K$ are known (visible) from the start, with bytes 3–7 to be found. Recall (page 351) that the `seed` bytes determine how the byte values 0...255 of the hidden state vector S are permuted during key setup. A weak IV leaves S in a state that potentially leaks information. To find the first unknown byte, `seed[3]`, IVs with byte pattern template $(3, 255, t)$ are used, with most values $0 \leq t \leq 255$ satisfying the requirements; frames with IVs of this form are found from captured traffic. To find key byte b, $3 \leq b \leq 7$, useful weak IVs include those of the form $(b, 255, t)$. An initial response of WEP product vendors was to disallow (filter out) IVs of this pattern; however, more advanced attacks soon emerged for which IV filtering was not possible.

‡**FMS ATTACK (DETAILS).** The first RC4 keystream byte is $z = S[S[1] + S[S[1]]]$. It depends on three entries of S at the instant key setup ends. FMS weak IVs aim to arrange that $S[1] = 0$ and $S[0] = b$, so that $z = S[0 + b] = S[b]$. This value z turns out to allow a direct calculation of `seed[b]`, using two main observations. (1) During key setup line 4 (page 351), when $i = b$, the unknown `seed[b]` is added into j along with known values; and (2) then in line 5, the updated value $j = j_i$ (for round $i = b$) determines the value swapped into index b as new value $S[b]$ from index j_i. From line 4 it now follows that, aside from `seed[b]`, j_i and $S[j_i]$ depend only on known or predictable values, including

[15] Due to standards-related issues, this is part of a known-plaintext LLC/SNAP header (page 360).

t (fixed for a given IV); here for $S[j_i]$, note that at this point $S[h] = h$ for most h (from line 1). This sketches out why seeing keystream byte $S[b]$ allows recovery of `seed[b]`—provided no further swaps (after round $i = b$) involve index b. This last condition is why for each b, multiple (e.g., 60) IVs of the desired form are collected. Based on modeling, the condition holds with probability somewhat over 0.05, so 60 instances are expected to yield at least 3 correct observations or "votes" for the correct value of `seed[b]`. Of 60 available votes, none of the other 255 (8-bit) values is expected (by model) to get even two votes, thus simple voting is expected to identify the correct key byte.

ATTACK TIME VS. KEY LENGTH. For a 40-bit WEP key, the FMS attack recovers 5 key bytes, each over one iteration step. For 104-bit keys, the 13 key bytes require 13 iterations, taking about 2.6 times longer—an increase only linear in the key length. In contrast, an attack time exponential in key length is expected (e.g., $0.5 \cdot 2^n$ for n-bit key trials) for cryptographic algorithms for which there are no known algorithmic weaknesses.

‡**Exercise** (Key recovery implementations). The FMS paper estimated requiring 4 million WEP frames. The first implementation used 5–6 million, but this was reduced to 1 million by optimizations. Independent public implementations of FMS key recovery soon appeared in tools AirSnort and WEPCrack.[16] (a) Summarize these optimizations (hint: [48]). (b) The *PTW attack*, a later statistical method not restricted to weak IVs, needed only 40,000 (85,000) frames for 50% (95%) probability of success in key recovery. In 2007 it recovered a 104-bit WEP key in 60 seconds. Explain how it acquired frames with different IVs using *active* techniques including re-injection of captured ARP requests[17] and *deauthenticate* management frames (hint: [50]; [49] for improvements). (c) Discuss the evolution of WEP key recovery attacks, from FMS to attacks utilizing arbitrary IVs; discuss passive vs. active attacks, the number of frames needed, attack times, and incorporation into automated key recovery tools such as Aircrack-ng (hint: [41]).

‡**Exercise** (Fragmentation attacks). IEEE 802.11 supports payload fragmentation and reassembly of MAC frames. An 8-byte known-plaintext LLC/SNAP (subnetwork access protocol) header commonly starts the WEP-encrypted body (Fig. 12.7 at '*' after IVD). Explain how this and one passively recorded encrypted frame can be used to: (a) recover an 8-byte keystream by simple XOR; (b) leverage 802.11 fragmentation to directly enable transmission of 64 bytes of encrypted data; (c) with further steps, recover 1500-byte keystreams, thereby allowing transmission (without fragmentation) of 1500 bytes of data; and (d) by inserting an IP header into frame fragments, get an AP to decrypt captured WLAN traffic and forward the result to an address of choice. (Hint for all parts: [6].)

‡**Exercise** (RC4 initial keystream bytes). When the 2001 FMS attack appeared, a suggested work-around was that when using RC4, an agreed-upon number of output bytes (at least 256) be discarded before using the keystream, to avoid biases in the initial keystream bytes. Why was this solution not promoted by the Wi-Fi Alliance? (Hint: [11]; cf. [27].)

SUMMARY COMMENT: KEY MANAGEMENT. In distributed networks using open standards and public communications channels, cryptographic protection is recognized

[16] Free cracking tools (illegal in some countries) rarely bring joy to product vendors, but are hard to ignore.
[17] The Address Resolution Protocol (ARP) is discussed in Chapter 11.

as the primary means for security. Aside from trustworthy algorithms, cryptography itself depends on one major assumption among all others: that secret keys remain secrets associated with (only) the authorized parties. Sound key management is absolutely essential, in all aspects. However, WEP's design failed here—its properties D2–D6 (page 357) would not have been approved by security experts. While it is understood that a standards committee has limited resources, the decision to declare management of the master key beyond the scope of WEP itself turned out badly for everyone but attackers. WEP's above-noted spectacular failures are strongly related to this void; the lack of built-in key management, with no requirement or support for easy key update, discouraged changing master keys, commonly resulting in a shared, long-term static key—increasing both its attractiveness as a target, and the consequences of its compromise. Had 802.11 failed to gain popularity, these issues would have drawn less attention and had little impact—but success attracts heavy scrutiny, and errors become both highly visible and costly to repair.

12.7 ‡AES-CCMP frame encryption and key hierarchy

The security of 802.11 networks improved with the introduction of the AES-CCM *protocol* (AES-CCMP), using the AES block cipher, as a replacement for WEP and its use of the RC4 stream cipher. Here we describe AES-CCMP frame encryption and how a set of session keys is derived from the pairwise master key. Section 12.8 then describes how group keys are managed, the four-pass handshake protocol used by AES-CCMP for mutual authentication between STA and AP, and further context on the evolution of 802.11 security from WEP to WPA3.

AES-CCMP FRAME ENCRYPTION. AES-CCMP uses the AES block cipher in the CCM *mode* (*counter mode* of operation for encryption, *with CBC-MAC* authentication).[18] Here the AES algorithm, which processes data in 128-bit blocks, is used with a 128-bit key as the basis for both frame encryption and frame integrity. CCM has two parts.

1. The CBC-MAC data authentication algorithm is used to compute a MAC (integrity tag), which is appended to the frame data.
2. The combined string is encrypted using *counter mode* (CTR), whereby a counter block is initialized (serving as an IV); then each 128-bit plaintext block is XOR'd to the AES-encryption of the current counter value (which itself increments after each plaintext block).

As discussed in Chapter 2, such algorithms combining authentication and encryption are called *authenticated encryption* (AE). In CCMP, some MAC frame header and CCMP header fields are also authenticated (but not encrypted), resulting in what is called *AE with associated data* (AEAD). Fields that are *mutable* (subject to change) are zeroed out for the purposes of MAC computation and verification.

CCMP HEADER, PN, CTR MODE NONCE. CCMP adds 16 bytes to an 802.11 frame body PDU (replacing WEP's 4+4 bytes, Fig. 12.7): 8 bytes of cleartext CCMP *header*

[18] The *counter mode of operation* (CTR), CBC-MAC and CCM are introduced in Chapter 2.

before the encrypted part, and 8 final bytes for the MAC integrity value.[19] Our main interest in this cleartext header is a 6-byte *packet number* (PN), and a `KeyID` field (cf. Fig. 12.7). The PN is used both to detect replay (below), and to avoid key reuse as explained next. A 48-bit PN is to be incremented with each frame (MPDU), yielding a PN that is unique for each MPDU (for a given PTK session key),[20] used to build a 128-bit *counter block* (Fig. 12.9) for CCM's CTR mode encryption. CTR mode requires, for its initial counter block, a *nonce* value that is never reused for the same key. A 104-bit nonce and a 16-bit rolling counter i are part of the full counter block (the rolling counter's 2^{16} values far exceed the number of 128-bit blocks to be counted, given the MPDU size limit). The value i is incremented (within the CTR mode) as each 128-bit block in the frame is encrypted, starting with $i = 1$ for the first 128-bit plaintext block. Here the PN (with extra assurance from the MAC source address) ensures that the 128-bit counter block of Fig. 12.9 starts at a value never used before for this PTK—as required by the CCM mode. Note that this value need not be unpredictable—uniqueness is the requirement.

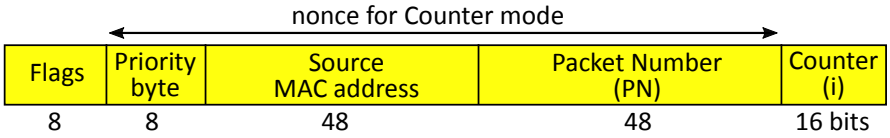

Figure 12.9: Counter block format for CTR mode in CCMP. The initial block uses $i = 1$. Flags and the Priority byte are omitted from our discussion. The nonce here need only be unique; in other cases, nonces are specified to be unpredictable random numbers.

Exercise (AES-CCMP details). (a) Describe full details for data integrity and encryption of frames in AES-CCMP, including padding fields and the frame format, how CCMP header fields are used, and the special first block in CCMP's CBC-MAC. (b) Discuss the advantages and disadvantages of AES-CCMP keeping only 8 bytes of 16-byte CBC-MAC. (Hint: [22]; [11, Ch.12]. CCMP follows RFC 3610 [55] with $M = 8$, $L = 2$.)

NOTATION FOR PSEUDO-RANDOM BIT GENERATION. We now introduce notation for *pseudo-random functions* (PRFs). PRFs are used in 802.11 to derive a set of keys from a master key (*key derivation*), and to initialize random strings. To instantiate a PRF, the HMAC construction is used on the SHA-1 hash function, yielding the MAC algorithm called HMAC-SHA-1 (Chapter 2). It takes two inputs—a key, and a message string—and produces 160 bits of output. For notation, with "::" denoting concatenation, define:

$$x\_hmac\_sha1(K, A, B, i) = HMAC\_SHA\_1(K, A :: [0] :: B :: [i]) \qquad (12.3)$$

where $[0]$ and $[i]$ denote 1-byte representations of 0 and unsigned integer i. To get 160 random bits, we call this function with $i = 0$. If the number t of desired random bits exceeds 160, then we call it again for $i = 1$ and so on, until the combined output (appending each new output to the end) is at least t bits, and use the first t bits. To generate $t = 384$ bits,

[19] The *MAC protocol data unit* (MPDU) carried by an 802.11 frame body (Fig. 12.2) may result from the fragmentation of a *MAC service data unit* (MSDU) that exceeds the MPDU maximum size.

[20] A 48-bit counter, starting from 1 and incremented per frame, is not expected to roll over, for a given session key. If the device is restarted, a new session key is established, and PN is reset to 1 by default.

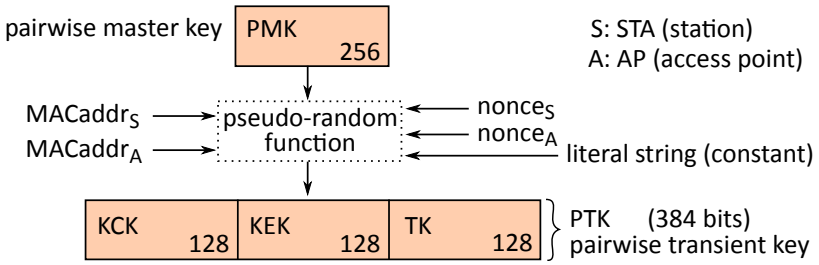

Figure 12.10: Derivation of session keys for AES-CCMP, also called a *key hierarchy*. PMK is agreed between STA and AS (then sent from AS to AP), or derived from a static password or PSK. Table 12.3 gives more information on acronyms KCK, KEK and TK.

we define $PRF\_384(K,A,B) = PRF(K,A,B,384)$ to mean calling $x\_hmac\_sha1(K,A,B,i)$ enough times to get 384 bits, i.e., using $i = 0, 1$ and 2.

SESSION KEYS FOR AES-CCMP. Three 128-bit keys comprise an AES-CCMP *pairwise transient key* (PTK); *transient* refers to a temporal/session key for the current association. One key is for frame encryption, the *temporal key* (TK). The other two protect messages in the four-pass handshake (page 365), as Table 12.3 notes. The PTK is generated from the *pairwise master key* (PMK) by the above pseudo-random function (PRF):

$$PTK = PRF\_384(PMK, \texttt{"Pairwise key expansion"}, A1 :: A2 :: n1 :: n2) \quad (12.4)$$

The PTK bitstring is partitioned to yield three keys $(KCK :: KEK :: TK)$, as Fig. 12.10 shows. Here $A1$ and $A2$, respectively, are the numerically lower and higher of the MAC addresses of the STA and AP, while $n1$ and $n2$, respectively, are the lower and higher of the 256-bit random number values $nonce_S$, $nonce_A$ from the four-pass handshake.

REPLAY PROTECTION. AES-CCMP has an anti-replay defense based on *packet number* (PN). A STA keeps, for each PTK, a 48-bit unsigned (received-PN) `ReplayCounter`, set to 0 when the PTK is established. A transmitting unit is expected to keep a (transmitted) `PNvalue` for each PTK, incremented by one before sending each encrypted frame. This allows a replay check by receivers: for each incoming frame with valid integrity, the frame is deemed a replay (and dropped) if the PN value extracted from its CCMP header is less than or equal to the `ReplayCounter` for this PTK; otherwise, the `ReplayCounter` is updated by the received PN value. Thus `ReplayCounter` monotonically increases.

Exercise (AES-CCMP improvements over WEP). Explain how the AES-CCMP design addresses WEP's security failures (listed in Section 12.6); summarize in a table.

Key name	Encryption	Integrity	Place of use
KCK (key confirmation key)		✓	four-pass handshake
KEK (key encryption key)	✓		four-pass handshake
TK (temporal key)	✓	✓	frame data (AES-CCM)

Table 12.3: Functions of the subkeys of the pairwise transient key (PTK).

12.8 Robust authentication, key establishment and WPA3

Wireless links introduce new risks not present in wired LANs. In light of this, an initial goal of 802.11 was to add security mechanisms to retain "wired equivalent privacy" (WEP). This unfortunately did not turn out to be the result. We recall again what went wrong with WEP, then follow the evolution of 802.11 security, with focus on the four-pass handshake, group key management, and additional authentication methods.

RECAP. The WEP design implicitly relied on each frame using a different keystream. Legitimate devices could aim for this by varying the IV—the design with per-packet IVs appears to support devices trying to avoid keystream reuse. But IV reuse was not prevented, and turned out to be impossible to avoid (for a static PSK), due to the IVs being short; and lack of support for automated key management resulted in static PSKs. The combination enabled a multitude of attacks (Sections 12.5–12.6). In retrospect, the design was naive and made without review by security experts—who plan for worst cases, based on experience and respect for attackers. The issue of IV and key(stream) reuse is related to message replay, which WEP also did not prevent. While detecting replay of application packets is often left to higher layers, at layer 2 we now expect a design to preclude reuse of keys across frames. This is one of many improvements delivered by the *robust network* redesign (below), which includes AES-CCMP and the four-pass handshake.

TKIP. In 2001, an immediately deployable WEP replacement was urgently needed. This constrained designs to those deliverable by software/firmware upgrade, with ongoing reliance on RC4 (including some aspects of its key setup, based on existing RC4 hardware-assist). The *Temporal Key Integrity Protocol* (TKIP) emerged as the interim solution. TKIP's design was complex yet understandably weak. It was deprecated by the Wi-Fi Alliance in 2015 (although deployed devices typically linger for many years). Thus our focus is on the longer-term AES-CCMP design (Section 12.7) that emerged alongside TKIP.

802.11 TIMELINE AND WPA. WEP, TKIP and AES-CCMP are part of the early history of 802.11 standards (Table 12.4). A related acronym is WPA (*Wi-Fi Protected Access*), the name for an 802.11 subset (including also minor variations and extensions) promoted by an industry association called the Wi-Fi Alliance (WFA). In late 2002, due to the perceived urgency to replace WEP, and with the redesign (802.11i) specifying TKIP and AES-CCMP still in progress, WPA was derived from the TKIP (RC4) part of a draft 802.11i. The final 802.11i specification (with both TKIP and AES-CCMP) was largely adopted by WFA and marketed as WPA2, supported by interoperability testing. WPA3 followed later (page 368).

MIXED ARCHITECTURES AND EVOLUTION. By the 802.11i plan,[21] once all WEP devices are retired or upgraded to TKIP, what remains to consider are devices that support (1) TKIP-only, or (2) both TKIP and AES-CCMP. A WLAN security policy could then allow TKIP associations in the short term, in the intermediate term allow interoperability between TKIP-only devices and newer devices capable of both TKIP and AES-CCMP, and in the longer term require that all associations use AES-CCMP. This plan salvaged investments in existing hardware, while moving in the longer term to a stronger design.

[21] This was circa 2002–2004, prior to AES-GCMP, WPA3 and other changes noted in Table 12.4.

Specification	Date	Notes
802.11–1997	1997	original version with WEP and RC4 (no AS, EAP, or AES)
WPA*	2002	TKIP-based subset of draft 802.11i (no AES)
802.11i	2004	security redesign: TKIP optional, AES-CCMP required
WPA2*	2004	interop spec based on full 802.11i (AES support required)
802.11w	2009	*protected management frames* (PMF): integrity for selected frames
802.11–2012	2012	SAE (Simultaneous Authentication of Equals) introduced
802.11ac	2013	new options: 256-bit AES keys, AES-GCMP
802.11–2016	2016	*SAE* (page 368) used to derive PMK (upgrades old PSK)
WPA3*	2018†	mandatory: PMF, SAE (if Personal), 192-bit (if Enterprise mode)
802.11b	1999	transmission rates up to 11 Mbps (from 1–2 Mbps in 1997)
802.11g	2003	transmission rates up to 54 Mbps (an improved 802.11a)
802.11n Wi-Fi 4	2009	150 Mbps × 4 streams maximum
802.11ac Wi-Fi 5	2013	from 433 Mbps to several Gbps (maximum 8 streams)
802.11ax Wi-Fi 6	2020	claims to approach 10 Gbps (maximum 8 streams)

Table 12.4: Selected 802.11 and *Wi-Fi Protected Access* (WPA) specs. Upper portion relates to security; lower portion relates to radio transmission. TKIP, AES-CCMP and AES-GCMP are 802.11 cipher suites for data confidentiality and integrity. Wi-Fi Alliance specifications (denoted *) promote WLAN interoperability certification (testing and branding) for both radio standards and security features. †(v2.0, Dec 2019; see Section 12.9)

‡**NONCE GENERATION.** The pseudo-random function (PRF) used to derive PTK (page 363, equation 12.4) is also used to initialize the counters for $nonce_S$, $nonce_A$. The nonces are inputs to this same PTK derivation, which occurs during the four-pass handshake (next). It is specified that these nonces should be randomly generated; 802.11 recommends that for RSNs (below), at device startup the nonce values be initialized using:

$$value = PRF\_256(RandomNum, \texttt{"Init Counter"}, LocalMACAddr :: Time) \quad (12.5)$$

RandomNum is a random number produced by the local device, itself perhaps 256 bits, depending on its cryptographic unpredictability. *Time* estimates current time (in Network Time Protocol format). The other arguments are a literal string and device MAC address.

FOUR-PASS HANDSHAKE (OVERVIEW). The four-pass handshake of 802.11i (2004) was a major step forward. It ties the 802.1X mutual authentication between STA and AS (which establishes a *pairwise master key*, PMK) to a session between STA and AP, deriving a new PTK (set of session keys) from the PMK. The exchange of fresh (random number) nonces allows derivation of a fresh PTK by equation (12.4), intended to be distinct from any previous association's session keys (even if from a static PSK or PMK). The main contents of the four messages are as follows (these so-called `EAPOL-Key` messages are unencrypted unless denoted; fields beyond our interest are omitted).

m1) STA ← AP: $nonce_A$ . . . . . a fresh 256-bit random number chosen by AP
m2) STA → AP:* $nonce_S$, IE_S . . . analogous nonce from STA + security option info
m3) STA ← AP:* $nonce_A$, IE_A, $E_{KEK}(GTK)$ . . . . . GTK is discussed below
m4) STA → AP:* `ACK` . . . . . *messages covered by an integrity MAC using KCK

IE_S, IE_A are `Information Element` (IE) fields related to the selected security options.

FOUR-PASS HANDSHAKE (NOTES). Observations on the handshake follow.

1. Per Fig. 12.10, the integrity MACs on messages m2–m4 demonstrate knowledge of KCK and therefore also PMK (as well as $nonce_A$, $nonce_S$, and the two MAC addresses).
2. The IEs allow a cross-check with the corresponding IEs (page 343) from the original association exchange, which is not integrity-protected. The integrity MAC here allows a check that the earlier elements were not manipulated. The $nonce_A$ in m3 is not needed for key derivation, but allows a sanity check that m3 belongs to this protocol run.
3. The `ACK` acknowledges that all checks succeeded, and that STA is ready to start authenticated encryption with the new temporal key (TK). Message m1 is not integrity protected, but if $nonce_A$ is tampered, the handshake will fail (a later integrity MAC check will fail).
4. The integrity MACs on m2–m3 not only protect the messages, but confirm knowledge of the new, freshly generated PTK (and thus the PMK it was derived from). This also confirms the "liveness" of the other party (i.e., that they participated in real time, since each message incorporates data that depends on a newly generated, unpredictable nonce), and absence of a middle-person or rogue AP (from the MAC on m3). While the STA–AS authentication generated PMK, this provides evidence to STA that this AP is trusted by that AS, and to the AP that this is the same STA that AS authenticated. *Aside:* it is for these reasons that $nonce_A$, $nonce_S$ should be fresh random numbers, rather than simply unique.
5. Since each party provides a nonce, neither controls the final value of PTK from equation (12.4), and the use of MAC addresses within this equation binds PTK values to specific devices. The PTK frame encryption key (TK) will be further varied on a per-frame basis by additional parameters in the CCM construction (e.g., packet number).
6. ‡In m3, GTK (sent with a key identifier) is for the case where the negotiated security options include a *group key* for encrypting multicast messages, as explained next.

‡**GROUP KEYS AND KEYID.** If multicast (including broadcast) messages from the AP are to be encrypted, then in practice a *group encryption key* is defined for the WLAN, allowing all STA devices to decrypt the same frame. To support this in AES-CCMP, beyond the 384-bit PTK, a 128-bit session key called a *group temporal key* (GTK) is used, derived from a random 256-bit AP-created *group master key* (GMK) as:

$$GTK = PRF\_128(GMK, \texttt{"Group key expansion"}, macAddr_A :: nonce_G) \qquad (12.6)$$

where $nonce_G$ is a random or pseudo-random number from the AP with address $macAddr_A$. The temporal key (for the group) is then (group) TK = GTK. *Aside:* as this TK is not pairwise, the data integrity (origin authenticity) aspect of CCMP authenticated encryption cannot be relied on to imply a unique source for multicast messages.

Now when the AP sends an encrypted multicast frame, it uses the CCMP header `KeyID` field (2 bits) to specify to STAs which TK to use for decryption. `KeyID` thus distinguishes between pairwise and group temporal keys. Allowing four choices (2 bits) enables real-time transitions when keys (pairwise or group) are updated, e.g., current-key vs. next-key.

‡**GTK KEY DISTRIBUTION.** The GTK is sent to each STA in its respective four-pass handshake, AES-encrypted using that STA's KEK (see message m3). This GTK distri-

bution process is simpler than establishing a PTK, since each STA's existing KEK—part of PTK—can be relied on. (Based on the PMK from each STA's 802.1X authentication, both STA and AP can generate the PTK upon completion of m2.) GTK is similarly updated as needed, by a customized two-pass *Group Key Handshake* (not discussed here); implementations may, e.g., update the GTK once an hour using this.

Exercise (Broadcast keys). In an 802.11 WLAN with all associated STA devices sharing pairwise master keys with their AP, why are group encryption keys needed for broadcast, i.e., why not encrypt broadcast messages for each STA, using individual pairwise keys? (*Aside:* independent of this, when a STA leaves a network, by convention it sends a disassociate message, at which point the AP updates the group key.)

RSNs. The 802.11i security amendment in 2004 (Table 12.4) defined the term *Robust Security Network* (RSN). Items introduced as part of this included:

1. defining an *RSN association* (RSNA) to be an association using the four-pass handshake, and defining an RSN as a WLAN whose policy requires that all associations be RSNAs;
2. making CCMP support mandatory for 802.11-compliant networks (while allowing TKIP as a non-mandatory option, and documenting WEP as a non-RSN option);
3. introducing the four-pass handshake with STA session keys derived per Fig. 12.10, tied into the 802.1X controlled-port design, and based on a PMK from either a PSK or the output of an 802.1X mutual authentication method;
4. providing WLAN access control via frame encryption, in that a temporal key is required in order to inject valid encrypted frames or extract plaintext from such frames;
5. generalizing the frame header WEP bit (Fig. 12.2) to a *protected* bit; and
6. disallowing the *Shared Key* protocol as a pre-association authentication alternative.

Despite consensus that RC4-based versions of 802.11 (WEP, TKIP) should no longer be used, fully eliminating such legacy systems is hard, even 20 years after the first attacks.

‡**Exercise** (Password-to-PSK mapping). Used when no external AS is involved, a PSK is often based on a pre-configured (secret) password or passphrase. The required 256-bit PSK is typically derived using a deterministic function, whose output is then the PMK. A technique from 2004 recommended deriving the PSK from the SSID and user password (8 to 63 ASCII characters) via 4096 iterations of HMAC-SHA-1. Specify the full details. (Hint: [22, Appendix H.4], including a reference implementation in C.)

Exercise (Session keys of other shared-key users). In an AES-CCMP architecture, suppose a PSK is shared among multiple users or posted in public view (*shared and public*). (a) Explain how one user can compute the PTK of another; clearly indicate what information is required (hint: this greatly simplifies the *offline dictionary attack* from page 359). (b) Given this, discuss the risks of (shared and public) Wi-Fi passwords in coffee shops.

Opportunistic Wireless Encryption (OWE). Free airport and hotel Wi-Fi services often use an *open* AP (no encryption or authentication). *OWE* aims to improve on this by encrypting traffic, with no extra user actions (but without any authentication). The STA and AP carry out an unauthenticated DH exchange upon establishing 802.11 association (Table 12.1, page 343). The DH secret is used to derive a PMK, which is used in the four-pass handshake triggered by the AP, establishing the session encryption keys.

Exercise (OWE security). (a) What threats does OWE address relative to an *open* AP? Consider both passive and active attacks, e.g., rogue AP. (b) Some free Wi-Fi services are not *open*, but use a *shared and public* PSK, i.e., a communal passkey posted publicly. Would using OWE instead help or hurt security? (Hint: [18]. Note: OWE is not part of WPA3, but is promoted by the Wi-Fi Alliance under a separate *Enhanced Open* program.)

SAE (DRAGONFLY). 802.11–2012 (Table 12.4) added a core 802.11 authentication method based on a pre-shared secret: SAE (Simultaneous Authentication of Equals). SAE first uses Dragonfly, a password-based key establishment protocol similar to SPEKE (Chapter 4), offering *forward secrecy* and mitigating *offline dictionary attacks* (above), which plague WPA2-personal. By Dragonfly, a STA and AP mutually authenticate, demonstrating knowledge of the shared secret; a resulting fresh, high-entropy PMK is then used in the four-pass handshake to derive the usual session keys. (See also Section 12.9.)

WPA3 AND HOME NETWORKS. WPA3 specifies five suites of security requirements: two *personal modes* (for small or home networks using static secrets or passwords), three *enterprise modes* (using 802.1X EAP methods for upper-layer authentication). These two divisions each include a baseline *WPA3only* mode, and a *transition mode* to interoperate with WPA2 products. The enterprise division also has a "192-bit" *WPA3only* mode with longer keys (e.g., $192 \times 2 = 384$-bit elliptic curve parameters paired with AES-256). The *WPA3only-personal* mode requires support of SAE and integrity-protected management frames, while disallowing TKIP, WEP, and older PSK authentication (whereby predictably mapping PSK to PMK then PTK is subject to dictionary attack). For home-network products, labels such as *WPA2(AES)-personal* are used to signal WPA2 excluding TKIP, and *WPA2-WPA3-transitional* to allow WPA3 devices to interoperate with WPA2 (AES).

‡**Exercise** (Device setup, WPS and DPP). *Wi-Fi Protected Setup* (WPS) was promoted in 2007 to easily share a secret (PSK) between an AP and user device (STA); it is now deprecated. In one mode, a button-push on the AP released the PSK to the STA; another used an 8-digit PIN. (a) Summarize the technical details of these methods and the circa-2011 flaw in the PIN method (hint: [54]). (b) Explain known vulnerabilities in algorithms for generating default WPA2 passwords, often on printed labels on router-APs (hint: [26]). (c) The *Device Provisioning Protocol* (DPP), released in parallel with WPA3 and promoted as *Easy Connect* by the Wi-Fi Alliance, replaces WPS. It aims to provision public-key identities for Wi-Fi devices. Summarize the main technical details of DPP. (Hint: [56].)

EAP METHODS. 802.11 leaves open the choice of 802.1X authentication method (EAP *method*), and a wide variety is used. Table 12.5 gives examples, some themselves being frameworks allowing choice of inner methods. Recall the goal (Section 12.3): STA–AS mutual authentication, resulting also in a PMK (Fig. 12.10, page 363).

METHOD CATEGORIES. EAP methods may be categorized by whether they use (symmetric) pre-shared secrets, or credentials based on public-key cryptography; and whether the exchange is direct or occurs after a tunnel is first set up (e.g., using TLS) to facilitate an inner method. Such tunnels may protect static secrets from dictionary attacks or preserve privacy of user identities. Among the most popular enterprise EAP methods has been EAP-TLS; here only the TLS authentication handshake (Chapter 9) is used, and mainly with mutual authentication based on certificates for both AS and client (STA), thus

RFC no.	Short title	Notes
1334 [25]	PAP and CHAP authentication	PAP is cleartext userid-password
1661 [44]	PPP (Point-to-Point Protocol)	allows PAP and CHAP
1994 [45]	CHAP	hash of secret and challenge
2759 [60]	MS-CHAPv2	CHAP variant, mutual authentication
2865 [39]	RADIUS	Remote Authentication Dial In User Service
2869 [38]	EAP over RADIUS	updated in part by RFC 3579
3579 [2]	RADIUS Support for EAP	see Section 12.3 herein
3748 [1]	EAP (Extensible Authen. Prot.)	includes EAP-GTC (generic token card)
5080 [33]	RADIUS implementation issues	suggests mitigations
4962 [20]	key management best practices	tied in by RFC 5247 abstract
5247 [3]	EAP key management framework	specifies EAP key hierarchy
draft [37]	PEAP (Protected EAP)	TLS tunnel method (Microsoft)
4851 [8]	EAP-FAST	tunnel, may be based on pre-shared key
4764 [5]	EAP-PSK	AES-based challenge-response
5216 [43]	EAP-TLS	updates original RFC 2716
5281 [14]	EAP-TTLS	TLS tunnel secures other methods
6124 [42]	EAP-EKE	specified after expiry of EKE patent
8146 [17]	EAP-pwd (with salting)	based on the Dragonfly key exchange

Table 12.5: Selected early RFCs on 802.11 security and EAP. An RFC (Request For Comments) is an official document under the Internet Society. A *standards-track RFC* must be endorsed by the Internet Engineering Task Force (IETF); non-endorsed *independent submissions* may also be published, and categorized as *Informational* or *Experimental*.

requiring management of client-side certificates. EAP methods that avoid client-side certificates are generally simpler to deploy, but may be less resistant to attacks.

12.9 ‡End notes and further reading

For a comprehensive introduction to 802.11 networks and *802.11i* security, WEP, its flaws, and how it differs from WPA/WPA2, see Edney [11]; Lehembre [24] gives an overview. Mishra [31] gives a clear, informed summary of pre-WPA 802.11 security issues, including *session hijacking* and *rogue APs*; Borisov [7] explains fundamental cryptographic design flaws. For *war driving* see Skoudis [46]. For *DoS attacks* on 802.11, including *deauthentication* and *disassociation*, see Bellardo [4]. For the *FMS key recovery attack* on WEP, see Fluhrer [13]; Stubblefield's implementation [48] verified its viability and improved some details. More efficient approaches followed, e.g., by Tews [49]; Bittau [6] found further serious attacks on WEP. On the unsuitability of encrypted CRCs for data authentication, see Menezes [30, p.363] (cf. Stubblebine [47]). For automated recovery of plaintext when Vernam keystreams are reused (or from the XOR of two plaintexts), see Mason [28] (also Dawson [10]). McEliece [29] clearly explains computational aspects of *finite fields*.

The Wi-Fi Alliance specifies precise requirements for WPA3 compliance [57]. For 802.11 standards, see Table 12.4; Chapter 2 gives references on the *Galois Counter Mode*

Protocol (AES-GCMP) added in 802.11ac. NIST offers guidance on RSN*s* and 802.11i [34], Bluetooth [35], and *mobile device management* [36]. IEEE 802.1X [23] specifies *port-based network access control*. Youm [59] surveys EAP *methods* (cf. Table 12.5).

RADIUS and EAP were designed for use with the *Point-to-Point Protocol* (PPP) [44]. In early Internet service over phone lines, a user's computer modem dialed into the local modem pool of an ISP. PPP converted the phone line to a bytestream channel and then relayed bytes. PPP's link-layer services are not used by 802.11, but its authentication architecture matches. In PPP, a password was originally sent plaintext (*PAP* [25]) or used in the challenge-response *CHAP* [45] protocol. This relied on the physical security of telephone wires—from which accessing and extracting data is harder than for 802.11 radio. EAP [1] replaced PAP and CHAP by a framework allowing choice of methods (and thus evolution). RADIUS (Remote Authentication Dial In User Service) relayed data between local modem pools and a centralized (RADIUS) server for verification; this avoided replicating secrets across numerous local databases. RADIUS evolved over time, but its historical name remains along with its functionality: relaying data between service access points and a central authentication server. *Diameter* [12] is an Authentication, Authorization and Accounting (AAA) protocol intended to replace RADIUS. For analysis of the variants *MS-CHAP* and *MS-CHAPv2* [60] of CHAP authentication, see Schneier [40].

The *eduroam* network (Education Roaming) [58] is an example of using 802.1X-based authentication and RADIUS servers; member institutions provide WLAN access via their local networks, to visiting users from other eduroam-participating institutions. Regarding the practice of *MAC address randomization* to protect privacy, see Vanhoef [51].

SAE was first advanced as a *password-authenticated key exchange* protocol (Chapter 4) for peer-to-peer authentication in *mesh networks* under 802.11s (July 2011); it was renamed Dragonfly and has many variants (e.g., [15, 16]), one used in *EAP-pwd* [19]. For side-channel attacks on Dragonfly (as used per 802.11's SAE and EAP-pwd), see the *Dragonblood* [53] work. See also Vanhoef's attacks [52] on WPA2's four-pass handshake that reinstall temporal keys and reset packet numbers such that nonces are reused (defeating anti-replay defenses). For formal analysis of WPA2, see Cremers [9].

As a grammatical note to end with, we write "a STA" (orally: a STAY, for station) rather than "an STA" (an ESS-TEE-AY), as we prefer the oral efficiency of the former.

References (Chapter 12)

[1] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and H. Levkowetz. RFC 3748: Extensible Authentication Protocol (EAP), June 2004. IETF Proposed Standard; obsoletes RFC 2284, updated by RFC 5247.

[2] B. Aboba and P. Calhoun. RFC 3579: RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP), Sept. 2003. Informational RFC; updates RFC 2869.

[3] B. Aboba, D. Simon, and P. Eronen. RFC 5247: Extensible Authentication Protocol (EAP) Key Management Framework, Aug. 2008. IETF Proposed Standard; updates RFC 3748.

[4] J. Bellardo and S. Savage. 802.11 denial-of-service attacks: Real vulnerabilities and practical solutions. In *USENIX Security*, pages 15–27, 2003.

[5] F. Bersani and H. Tschofenig. RFC 4764: The EAP-PSK Protocol—A Pre-Shared Key Extensible Authentication Protocol (EAP) Method, Jan. 2007. Experimental RFC.

[6] A. Bittau, M. Handley, and J. Lackey. The final nail in WEP's coffin. In *IEEE Symp. Security and Privacy*, pages 386–400, 2006.

[7] N. Borisov, I. Goldberg, and D. A. Wagner. Intercepting mobile communications: The insecurity of 802.11. In *ACM MobiCom*, pages 180–188, 2001.

[8] N. Cam-Winget, D. McGrew, J. Salowey, and H. Zhou. RFC 4851: Flexible Authentication via Secure Tunneling Extensible Authentication Protocol Method (EAP-FAST), May 2007. Informational RFC.

[9] C. Cremers, B. Kiesl, and N. Mediner. A formal analysis of IEEE 802.11's WPA2: Countering the KRACKs caused by cracking the counters. In *USENIX Security*, 2020.

[10] E. Dawson and L. Nielsen. Automated cryptanalysis of XOR plaintext strings. *Cryptologia*, 20(2):165–181, 1996.

[11] J. Edney and W. A. Arbaugh. *Real 802.11 Security: Wi-Fi Protected Access and 802.11i*. Addison-Wesley, 2003.

[12] V. Fajardo, J. Arkko, J. Loughney, and G. Zorn. RFC 6733: Diameter Base Protocol, Oct. 2012. IETF Proposed Standard; updated by RFCs 7075 and 8553, obsoletes RFCs 3588 and 5719.

[13] S. R. Fluhrer, I. Mantin, and A. Shamir. Weaknesses in the key scheduling algorithm of RC4. In *Workshop on Selected Areas in Cryptography (SAC)*, pages 1–24, 2001.

[14] P. Funk and S. Blake-Wilson. RFC 5281: Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0), Aug. 2008. Informational RFC.

[15] D. Harkins. Simultaneous Authentication of Equals: A secure, password-based key exchange for mesh networks. In *Sensor Tech. and Applications (SensorComm)*, pages 839–844, 2008. See warning in [19].

[16] D. Harkins. RFC 7664: Dragonfly Key Exchange, Nov. 2015. Informational RFC.

[17] D. Harkins. RFC 8146: Adding Support for Salted Password Databases to EAP-pwd, Apr. 2017. Informational RFC; updates RFC 5931. Note: EAP-pwd is based on the Dragonfly key exchange.

[18] D. Harkins and W. Kumari. RFC 8110: Opportunistic Wireless Encryption, Mar. 2017. Informational.

[19] D. Harkins and G. Zorn. RFC 5931: Extensible Authentication Protocol (EAP) Authentication Using Only a Password, Aug. 2010. Informational; updated by RFC 8146 [17]. RFC 5931's official Errata notes that the EAP-pwd key exchange (Dragonfly) of RFC 7664 [16] addresses a side-channel attack on the method in 5931, and that consequently the method in 7664 should be used instead.

[20] R. Housley and B. Aboba. RFC 4962: Guidance for Authentication, Authorization, and Accounting (AAA) Key Management, July 2007. IETF Best Current Practice.

[21] IEEE Computer Society. IEEE Std 802.11–2007, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. June 2007 (1184 pages), incorporating the 8 amendments since 802.11–1999; superseded by 802.11–2012 (2695 pages) and 802.11–2016. The IEEE 802 group addresses Local and Metropolitan Area Networks (LANs and MANs).

[22] IEEE Computer Society. IEEE Std 802.11i–2004, Amendment 6: Medium Access Control (MAC) Security Enhancements, Jul 2004. 175 pages. Provides security enhancements for 802.11–1999 [21].

[23] IEEE Computer Society. IEEE Std 802.1X–2010: Port-Based Network Access Control, Feb 2010. 205 pages. Revises 802.1X–2004; superseded by 802.1X–2020. The IEEE 802 group addresses Local and Metropolitan Area Networks (LANs and MANs).

[24] G. Lehembre. Wi-Fi security—WEP, WPA and WPA2. *Hakin9* (magazine), pages 2–15, Jun 2005. `https://hakin9.org/`.

[25] B. Lloyd and W. Simpson. RFC 1334: PPP Authentication Protocols, Oct. 1992. IETF Proposed Standard; obsoleted by RFC 1994 (PPP CHAP [45]).

[26] E. N. Lorente, C. Meijer, and R. Verdult. Scrutinizing WPA2 password generating algorithms in wireless routers. In *USENIX Workshop on Offensive Technologies (WOOT)*, 2015.

[27] I. Mantin. A practical attack on the fixed RC4 in the WEP mode. In *ASIACRYPT*, pages 395–411, 2005.

[28] J. Mason, K. Watkins, J. Eisner, and A. Stubblefield. A natural language approach to automated cryptanalysis of two-time pads. In *ACM Comp. & Comm. Security (CCS)*, pages 235–244, 2006.

[29] R. McEliece. *Finite Fields for Computer Scientists and Engineers*. Kluwer, 1987.

[30] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996. Openly available, `http://cacr.uwaterloo.ca/hac/`.

[31] A. Mishra, N. L. Petroni Jr., W. A. Arbaugh, and T. Fraser. Security issues in IEEE 802.11 wireless local area networks: A survey. *Wireless Communications and Mobile Computing*, 4(8):821–833, 2004.

[32] R. Moskowitz. Weakness in passphrase choice in WPA interface. *WNN Wi-Fi Net News*. 4 Nov 2003.

[33] D. Nelson and A. DeKok. RFC 5080: Common Remote Authentication Dial In User Service (RADIUS) Implementation Issues and Suggested Fixes, Dec. 2007. IETF Proposed Standard; updates RFCs 2865, 2866, 2869, 3579.

[34] NIST. Special Pub 800-97, Establishing Wireless Robust Security Networks: A Guide to IEEE 802.11i. U.S. Dept. of Commerce, Feb 2007.

[35] NIST. Special Pub 800-121 r2: Guide to Bluetooth Security. U.S. Dept. of Commerce, May 2017.

[36] NIST. (Draft) Special Pub 800-124 r2: Guidelines for Managing the Security of Mobile Devices in the Enterprise. U.S. Dept. of Commerce, Mar 2020.

[37] A. Palekar, D. Simon, J. Salowey, H. Zhou, G. Zorn, and S. Josefsson. Protected EAP Protocol (PEAP) Version 2. Internet-Draft (Category: Informational, EAP Working Group), 15 October 2004, `draft-josefsson-pppext-eap-tls-eap-10.txt`. See also Chapter 9 in [11].

[38] C. Rigney, W. Willats, and P. Calhoun. RFC 2869: RADIUS Extensions, June 2000. Informational RFC; updated by RFC 3579, see also RFC 5080.

[39] C. Rigney, S. Willens, A. Rubens, and W. Simpson. RFC 2865: Remote Authentication Dial In User Service (RADIUS), June 2000. IETF Draft Standard. Obsoletes RFC 2138, which obsoleted 2058; updated by RFCs 2868, 3575, 5080, 6929, and 8044. See also RFC 5176.

[40] B. Schneier, Mudge, and D. A. Wagner. Cryptanalysis of Microsoft's PPTP authentication extensions (MS-CHAPv2). In *Secure Networking—CQRE (Secure)*, pages 192–203. Springer LNCS 1740, 1999.

[41] P. Sepehrdad, P. Susil, S. Vaudenay, and M. Vuagnoux. Smashing WEP in a passive attack. In *Fast Software Encryption*, pages 155–178, 2013. Extended version (2015, 65 pages): "Tornado attack on RC4 with applications to WEP and WPA".

[42] Y. Sheffer, G. Zorn, H. Tschofenig, and S. Fluhrer. RFC 6124: An EAP Authentication Method Based on the Encrypted Key Exchange (EKE) Protocol, Feb. 2011. Informational RFC.

[43] D. Simon, B. Aboba, and R. Hurst. RFC 5216: The EAP-TLS Authentication Protocol, Mar. 2008. IETF Proposed Standard; obsoletes RFC 2716.

[44] W. Simpson. RFC 1661: The Point-to-Point Protocol (PPP), July 1994. IETF Internet Standard.

[45] W. Simpson. RFC 1994: PPP Challenge Handshake Authentication Protocol (CHAP), Aug. 1996. IETF Draft Standard; obsoletes RFC 1334.

[46] E. Skoudis and T. Liston. *Counter Hack Reloaded: A Step-by-Step Guide to Computer Attacks and Effective Defenses (2nd edition)*. Prentice Hall, 2006 (first edition: 2001).

[47] S. G. Stubblebine and V. D. Gligor. On message integrity in cryptographic protocols. In *IEEE Symp. Security and Privacy*, pages 85–104, 1992.

[48] A. Stubblefield, J. Ioannidis, and A. D. Rubin. Key recovery attack on the 802.11b wired equivalent privacy protocol (WEP). *ACM Trans. Inf. Systems and Security*, 7(2):319–332, 2004. Extends *NDSS* 2002 paper.

[49] E. Tews and M. Beck. Practical attacks against WEP and WPA. In *ACM WiSec*, pages 79–86, 2009.

[50] E. Tews, R. Weinmann, and A. Pyshkin. Breaking 104 bit WEP in less than 60 seconds. In *Workshop on Information Security Applications (WISA)*, pages 188–202, 2007.

[51] M. Vanhoef, C. Matte, M. Cunche, L. S. Cardoso, and F. Piessens. Why MAC address randomization is not enough: An analysis of Wi-Fi network discovery mechanisms. In *AsiaCCS*, pages 413–424, 2016.

[52] M. Vanhoef and F. Piessens. Key reinstallation attacks: Forcing nonce reuse in WPA2. In *ACM Comp. & Comm. Security (CCS)*, pages 1313–1328, 2017. See also `https://www.krackattacks.com/`, and the authors' *CCS* 2018 follow-up, "Release the Kraken: New KRACKs in the 802.11 standard".

[53] M. Vanhoef and E. Ronen. Dragonblood: A security analysis of WPA3's SAE handshake. In *IEEE Symp. Security and Privacy*, 2020.

[54] S. Viehböck. Brute forcing Wi-Fi Protected Setup. Technical report, 26 Dec 2011 (version 3).

[55] D. Whiting, R. Housley, and N. Ferguson. RFC 3610: Counter with CBC-MAC (CCM), Sept. 2003. Informational RFC.

[56] Wi-Fi Alliance. Wi-Fi Easy Connect Specification (Version 2.0). 14 Dec 2020 (revises: Version 1.0, Device Provisioning Protocol Specification, 9 Apr 2018), `https://www.wi-fi.org`.

[57] Wi-Fi Alliance. WPA3 Specification (Version 2.0). 20 Dec 2019, `https://www.wi-fi.org`.

[58] K. Wierenga and L. Florio. Eduroam: past, present and future. *Computational Methods in Science and Technology*, 11(2):169–173, 2005. See also: `https://www.eduroam.org`.

[59] H. Y. Youm. Extensible Authentication Protocol overview and its applications. *IEICE Trans. Inf. Syst.*, 92-D(5):766–776, 2009.

[60] G. Zorn. RFC 2759: Microsoft PPP CHAP Extensions, Version 2, Jan. 2000. Informational RFC; improves on MS-CHAPv1 (RFC 2433).