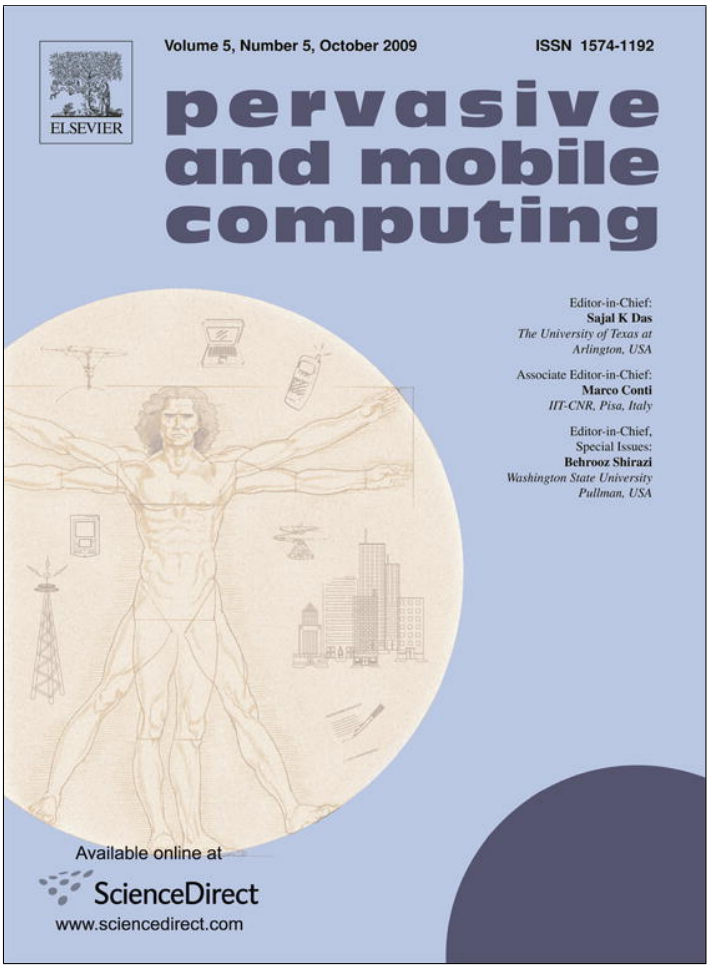


Pervasive and Mobile Computing 5 (2009) 510–525

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

journal homepage: www.elsevier.com/locate/pmc

Fast track article

Bringing law and order to IEEE 802.11 networks—A case for DiscoSec☆

Ivan Martinovic*, Paul Pichota, Matthias Wilhelm, Frank A. Zdarsky, Jens B. Schmitt

Distributed Computer Systems Lab (disco), University of Kaiserslautern, Germany

ARTICLE INFO

Article history:
Received 30 September 2008
Received in revised form 19 January 2009
Accepted 5 March 2009
Available online 24 March 2009

Keywords:
IEEE 802.11 security
Frame authentication
DoS protection
Implementation
Measurements

ABSTRACT

To improve the tarnished reputation of WLAN security, the new IEEE 802.11i standard provides means for mutual user authentication and assures confidentiality of user data. However, the IEEE 802.11 link-layer is still highly vulnerable to a plethora of simple, yet effective attacks which further jeopardize the already fragile security of wireless communications.

Some of these vulnerabilities are related to limited hardware capabilities of access points and their abuse may result in serious degradation of control over the wireless connection, which, especially in the case of broadcast communication, allows for client hijacking attacks. Although these issues are known and their impact is expected to be less prevalent on modern equipment, this work demonstrates the opposite. In our experimental analysis, we tested frequently used access points, and by forcing them to operate on their performance limits, we identified significant operational anomalies and demonstrated their impact on security by implementing a novel version of the Man-In-The-Middle attack, to which we refer as the *Muzzle attack*.

Secondly, this work describes DiscoSec, a solution for "patching" WLANs against a variety of such link-layer attacks. DiscoSec provides DoS-resilient key exchange, an efficient frame authentication, and a performance-oriented implementation. By means of extensive real-world measurements DiscoSec is evaluated, showing that even on very resource-limited devices the network throughput is decreased by only 22% compared to the throughput without any authentication, and by 6% on more performance-capable hardware. To demonstrate its effectiveness, DiscoSec is available as an open-source IEEE 802.11 device driver utilizing well-established cryptographic primitives provided by the Linux Crypto API and OpenSSL library.

1. Introduction

The various confidentiality and integrity vulnerabilities of Wired Equivalent Privacy (WEP) and the simplicity of launching impersonation attacks by MAC address manipulation caused the bad reputation of IEEE 802.11 security. To regain trust in this widespread technology the IEEE Task Group *i* successfully finalized the new security standard *802.11i*. The new standard provides a security framework composed of several known and approved protocols to ensure robust protection of wireless communication. An enhanced user authentication, a new underlying cipher, and a reliable integrity verification finally enabled the protection of data equivalent to the security in wired networks.

☆ This article is an extended version of work given in [I. Martinovic, P. Pichota, M. Wilhelm, F.A. Zdarsky, J.B. Schmitt, Design, implementation, and performance analysis of discosec–service pack for securing WLANs, in: 9th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WOWMOM 2008), Newport Beach, CA, USA, June 2008].

* Corresponding author.
E-mail address: martinovic@informatik.uni-kl.de (I. Martinovic).

1574-1192/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.pmcj.2009.03.002

Table 1
IEEE 802.11 frame types and connection states within they are allowed to be transmitted. Bold frames are authenticated by DiscoSec.

	Management frames	Data frames
State 1	Beacon, Probe Req./Resp., Traffic Indication Message, **Authentication Req./Resp., Deauthentication.**	None (infrastructure BSS)
State 2	**Association Req./Resp., Reassociation Req./Resp., Deauthentication Req.**	None
State 3	**Deauthentication Req. Disassociation Req.**	**All frames**

However, IEEE 802.11i [2] focuses only on securing higher layer protocol data transferred within *data frames*, leaving all other IEEE 802.11 link-layer services unprotected. The reason seems to be twofold. First, the tragic end of WEP left wireless clients without standardized protection for user's data, giving rise to a dispersion of proprietary solutions. Clearly, the interoperability certification program (i.e., Wi-Fi Protected Access) was impatiently awaited by both the industry and the users, leaving less time to solve all security concerns of the highly exposed IEEE 802.11 technology.

Secondly, attacks on the link-layer impact the *availability* of the IEEE 802.11 network, which, especially in wireless networks, is the most vulnerable among all security goals. The reason lies in an indigenous property of wireless communication which is unable to protect itself from channel capacity attacks (e.g., frequency jamming) and thus availability of the link-layer is often downgraded or even sacrificed to avoid more complex radio technologies. Nevertheless, there is a significant difference between physical layer attacks, like frequency jamming which denies *any* communication on the wireless channel, and link-layer attacks selectively, affecting the *services* provided by an access point and the connection states of wireless stations.

There are two major security vulnerabilities which are a frequent source of various sophisticated attacks against all IEEE 802.11 link-layer services. The first one concerns the *management frames* which are used for state transitions within the IEEE 802.11 state machine. These frames are not protected by the IEEE 802.11i security standard and the only authentication mechanism is still based on a receiver's blind trust in a sender's MAC address. Hence, simple address manipulation offers a plethora of well-known impersonation attacks, such as the popular Deauthentication and Disassociation attacks (for most prominent attacks abusing unauthenticated management frames see, e.g., [3]).

The second vulnerability is based on simple but very effective *resource exhaustion* of performance-limited devices. The association procedure of an IEEE 802.11 network utilizes a stateful protocol execution which is prone to DoS attacks, especially to a memory-depletion attack of wireless access points (APs). After receiving an authentication request, which is the first frame sent to an AP after the network discovery, an AP must reserve memory for a client's connection state. Considering the simplicity of previously mentioned IEEE 802.11 link-layer address manipulation and taking advantage of broadcast communication, flooding an AP with a high number of fake authentications offers a low-cost attack that can easily exhaust the AP's memory. The outcome of such an attack, which is often a full crash of a device where only manual reset helps, makes it even more attractive as it can be used as starting point for creating more sophisticated attacks against other security objectives.

Since 2003/2004 when such attacks were first identified, various tools have become freely available to facilitate their execution. However, one would assume that today's modern equipment provides better resource-protection and does not end up with such devastating results. Yet, the popularity and frequency of these attacks is increasing rapidly. The reason is a fast deployment of commercial wireless hotspots within dynamic environments such as airports, coffee shops, public parks, hotels, etc. They offer an instant Internet access by simple web-based payments. Nevertheless, such simplicity has a much higher price, especially if the link-layer is lacking a mutual authentication and if users are unable to distinguish between legitimate and rogue APs.

Additionally, many Internet Service Providers (ISPs) deliver APs as a part of their Cable or DSL contracts. Such widespread wireless network deployment further increases a user's dependability on IEEE 802.11 technology as it becomes an integral part of their living spaces.

1.1. Our contribution

The contribution of this article is twofold. First, we provide a detailed description of novel vulnerabilities based on abusing performance bottlenecks of modern state-of-the-art IEEE 802.11 devices. Such vulnerabilities, combined with unauthenticated management frames, easily result in low-cost, yet sophisticated rogue AP and Man-In-The-Middle attacks which are especially effective within popular hotspots. Secondly, this work offers a solution called DiscoSec. Without changing the IEEE 802.11 association procedure, DiscoSec enables efficient frame authentication and DoS-resilient key exchange. Table 1 lists frames protected by DiscoSec and the client's connection states within which they are allowed to be transmitted. From the moment the Authentication Request is successfully accepted, DiscoSec-enhanced stations and the AP share a 128-bit secret used for creating and verifying strong AES-based Message Authentication Codes (MACs). Due to its good performance

characteristics DiscoSec is able to efficiently compute MACs not only for management frames but also for all data frames transmitted after the station has been successfully authenticated. This is an important feature because all heavy-weight protocols such as IEEE 802.11i or IEEE 802.1X depend on correct exchange of data frames. As a result, using DiscoSec, wireless stations are immune even against newly identified IEEE 802.11i protocol blocking attack (as described in [4]).

The concept of DiscoSec has initially been described in [1]; however, in this paper we finalize our research by providing an overall picture of attacks, and extend DiscoSec's key-exchange to support both a Diffie–Hellman key exchange based on discrete logarithms (DH) and elliptic curves (ECDH). Furthermore, this work extensively analyzes the performance of DiscoSec's components, implementation issues, and discusses different lessons learned during the research. To demonstrate its effectiveness and to serve as a benchmark and prototype for future research the implementation of DiscoSec is made available as an open source IEEE 802.11 device driver.[1]

The rest of the paper is organized as follows. Section 2 covers related work, and Section 3 provides an analysis of performance-based vulnerabilities discovered after subjecting modern APs to resource-depletion attack. The protection objectives of DiscoSec are discussed in Section 4, while Section 5 describes the architecture, cryptographic primitives used for protection, and implementation issues. Section 6 introduces DoS protection against resource-depletion attacks aimed at DiscoSec's key-exchange. The performance evaluation providing insights into key-exchange computation, delays caused by frame authentication and impact on overall network throughput is given in Section 7. Section 8 discusses the choice of underlaying cipher and some performance properties of the ciphers' implementations offered by the Linux Crypto module. Finally, Section 9 concludes this work.

2. Related work

Since 2001, when IEEE 802.11 Wired Equivalent Privacy (WEP) enabled various attacks against all security objectives, the IEEE 802.11 networks have been going through hard times to improve their tarnished reputation. Most of the research, however, has been focused mainly on analyzing confidentiality and integrity problems caused by WEP [5–8]. Today, while the IEEE 802.11i standard seems to offer satisfying security for user's data, the vulnerable link-layer, i.e., unprotected management frames and effortlessness of launching resource-depletion attacks, still presents a significant security problem.

In [3], Bellardo et al. were among the first to describe how services offered by the IEEE 802.11 link-layer are vulnerable to impersonation attacks, and more importantly the authors demonstrate the great flexibility of such attack by being able to selectively choose among transmission rate throttling, fully denying associations, or selectively attacking single associated stations. The simplicity of such attacks made them popular, and tools such as void11 [9] and Airjack [10] were soon available to facilitate their execution. Based on the same vulnerabilities, in [4], He et al. successfully attacked even the new security standard IEEE 802.11i, which due to its complex message exchange and resource consumption further extended an attacker's toolbox. This attack also exemplarily showed that even sophisticated protection that depends on insecure building blocks often results in new vulnerabilities (e.g., execution of the IEEE 802.11i standard can easily be disrupted by management frame impersonation).

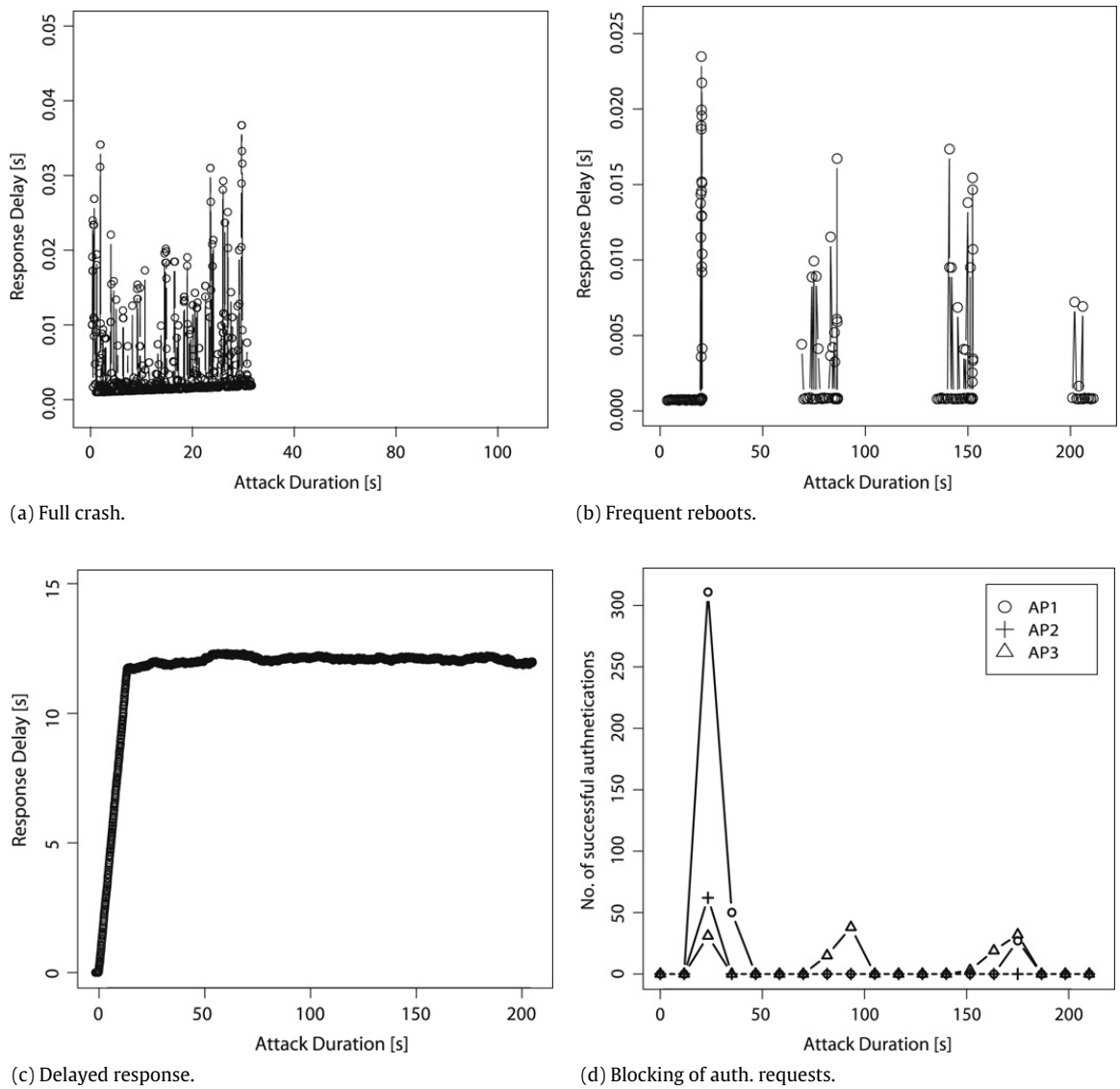
Considering research activities offering solutions to the aforementioned problems, Faria et al. [11] use a public key infrastructure to authenticate management frames. The authors compose a new security architecture by employing two protocols, SIAP and SLAP, to establish a secure association between wireless clients and an AP. While their solution offers frame encryption, it also completely modifies the IEEE 802.11 state machine and requires an additional SIAP server. Furthermore, the costly public key based on traditional RSA encryption seems only to be briefly evaluated on high-end devices, rather than on realistic, performance-limited access points. In our work, we take a different approach in which one of the main objectives is not to change the state machine, resulting in the support of both legacy and protected clients to be associated within the same AP.

Additionally, in our performance analysis we include new throughput-increasing features, such as Atheros' SuperG, which although proprietary, is based on frame aggregation technique and is currently being standardized by IEEE 802.11 Task Group N. This accounts for much longer frames transmitted over the wireless medium and may significantly increase computational stress caused by authentication on performance-limited APs.

Consequently, if not taken into account it may end up in new DoS vulnerabilities (more information on this topic is provided in DiscoSec's performance evaluation section).

Recently, the problem of rogue APs was also discussed in Ma et al. [12], where the authors describe a solution based on non-cryptographic methods. Their framework consists of various components which attempt to statistically differentiate between signals sent from a legitimate AP and those transmitted by an adversary AP. However, the framework requires controlled network deployment (i.e., planned placement of network components), additional hardware devices to enable monitoring of the channels, and wireless range extenders for a better radio coverage. A further disadvantage is the lack of a per-frame authentication, i.e., the solution does not offer a proactive protection against impersonation attacks. The statistical methods may be able to react to a rogue AP attack, but can hardly prevent it. Our work contributes to the prevention of rogue APs and session hijacks by allowing for strong cryptographic association to be efficiently established between an AP and wireless clients.

[1] DiscoSec's source code and installation guidelines are available under http://disco.informatik.uni-kl.de/content/downloads.



(a) Full crash.

(b) Frequent reboots.

(c) Delayed response.

(d) Blocking of auth. requests.

Fig. 1. Resource-depletion attacks against frequently used APs. From 12 analyzed APs, 7 exhibited faulty operation, such as full crash (a), frequent reboots (b), or increased response delay (c). Some APs implemented simple protection based on allowing only a certain number of association requests (d).

The importance of a cryptographic frame authentication is also recognized as an important issue by many IEEE 802.11 equipment vendors. For example, in certain product series, Cisco offers a proprietary security feature called Management Frame Protection (MFP). An AP extends the management frames by adding and verifying a message integrity check information element to each frame. Regrettably, there is no information (other than white paper [13]) offering more technical analysis of this proprietary solution. However, MFP does not seem to be a client-side supported feature. It only protects APs, while clients remain vulnerable to management frame attacks.

3. Abusing performance bottlenecks—Why hacking, when breaking is easier?

Even if the limited hardware capabilities of wireless APs do not present a significant security drawback under the anticipated network operation, bringing devices to the edge of their performance limits often results in unpredictable behavior. As a result, the services necessary to control a wireless channel and to manage wireless stations may no longer be assured. To analyze such *on-the-limit* behavior of wireless devices we gathered frequently used APs from well-known manufacturers such as Proxim, D-Link, Belkin, Linksys, Cisco, Siemens, and Lancon and monitored their operation after subjecting them to resource-depletion attacks. The analysis started by sending 100 fake authentication requests per second, while at the same time we measured the AP's response delay, frame loss, number of accepted/rejected associations, and other properties allowing us to fingerprint difficulties in AP's operation.

Even if we assumed that flooding would impact APs, the results of the depletion attack were devastating. Most APs we analyzed were evidently tarnished by the resource exhaustion and experienced highly impaired operations. We leave the exact model names of analyzed APs intentionally out; however, they are widely used and some of them are delivered as a part of broadband Internet access contracts. In Fig. 1 we provide an example of the most frequent outcomes of such an

attack. Each subfigure represents a group of APs which exhibited equal or similar operational anomalies. For example, two APs completely crashed, after which only a manual reset of the devices could help (depicted in Subfigure (a)). They did not respond to any further requests sent either by new or already associated clients. However, both APs still broadcasted *Beacon* frames used for network discovery, i.e., they announced the wireless network even if no link-layer service could be provided. As such, abusing them to create an attack against wireless clients is a trivial task. Similar problems were experienced by another two APs (shown in Subfigure (b)). Although these APs did not crash permanently, their operation was affected by frequent reboots, during which no channel control could be maintained. Clearly, such outcome affected both new and already associated clients.

Further analysis identified three APs which increased their response delays to the magnitudes of seconds (see Subfigure 1(c)). Such high delays rapidly filled up the AP's transmission queue, and thus resulted in overflows producing frame loss higher than $\geq$ 70%. This behavior, as we discuss in more detail in the next section, can also be used to successfully hijack wireless clients. Three tested APs seem to implement a simple resource-protection mechanism based on allowing only a limited number of client associations. After reaching a threshold of accepted requests, the APs would block any further request for a certain period of time (the frequency and the number of allowed associations is given in Subfigure (d)). Evidentially, the price for such protection is paid by the wireless clients. Flooding with the association rate, the attacker succeeds in instantly reaching the allowed threshold, and thus leaves no opportunity for legitimate clients to successfully associate.

3.1. An implementation of the Muzzle attack

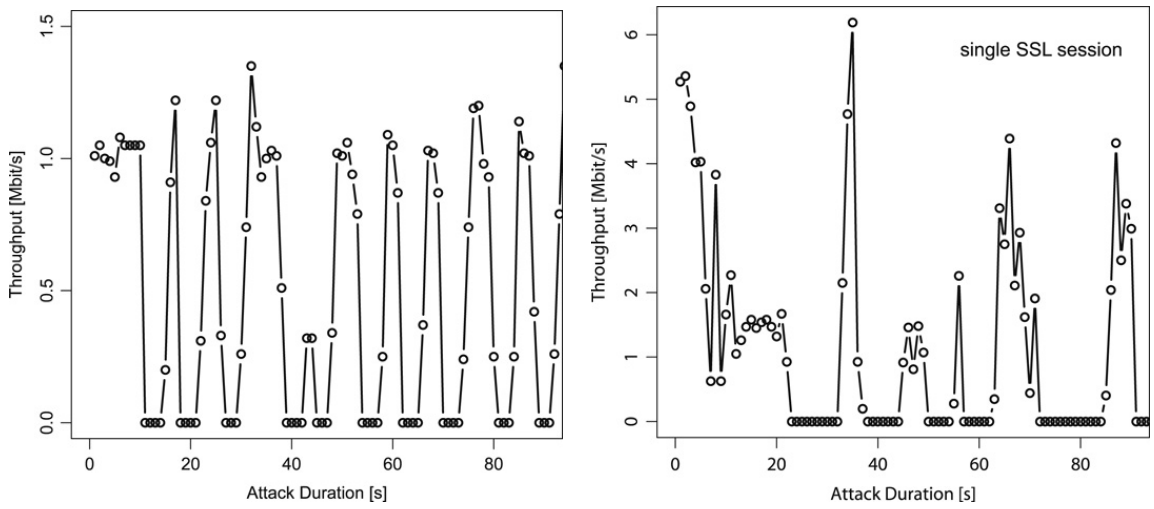
Before describing the Muzzle attack, we briefly recall the rogue AP attack and currently available countermeasures. The rogue AP attack is a simple, yet highly effective impersonation attack that abuses the default configuration of wireless devices which choose to associate with the AP providing the strongest received signal strength (RSS). Selecting the AP with the strongest RSS is an understandable decision since such an AP can support the highest available data rates. Hence, by installing the rogue AP on a different wireless channel, disguising it by setting the network identifier as the legitimate network, and having a stronger RSS, the rogue AP can easily attract new clients from the beginning of the network discovery. After the client is associated with the rogue AP, data frames can be transmitted and the adversary can launch sophisticated higher-layer attacks. Today, to avoid such attacks any high-end AP has a built-in rogue AP detection. Moreover, since rogue AP attack is considered as one of the most serious WLAN threats, there is a growing market of different Wireless Intrusion Detection Systems (WIDS) such as AirDefense [14], AirMagnet [15], 3Com AirProtect [16], etc. Common to all these and similar protections is the way of detecting rogue transmissions. It is based on background scanning for untrusted APs which are installed on different wireless channels. Such tools have powerful radio capabilities and can easily detect transmissions which are not defined as legitimate, i.e., they operate on different wireless channels using the network identifiers of legitimate networks. The rogue AP is forced to use another channel, otherwise the fake transmission with the client would be immediately detected by the legitimate AP which would disrupt it by sending deauthentication frames. These frames reset the client's connection and disconnect the client from the rogue AP. Clearly, such protection assumes that the AP is always able to (i) control the wireless channel through traffic monitoring and (ii) promptly react in case of detected transmissions from not associated clients (i.e., clients associated with the rogue AP). Since a link-layer address and network identifier can be easily cloned, the wireless channel is the only unique feature used to distinguish between a rogue AP and a legitimate AP. This leads us to a new attack in which the adversary attempts to hinder the reaction of the legitimate AP, and hence to muzzle it, while at the same time offering fake link-layer services on the same wireless channel. In such a case all existing countermeasures are not effective because the link-layer transmissions are completely equal.

While the Muzzle attack may be considered as a version of impersonation attack, the difference is that during the Muzzle attack the legitimate AP is still able to observe the fake transmission on its own channel, but due to the resource-exhaustion, the rogue AP is the first to answer the client requests.

Using the same hardware as in our previous analysis, the next few steps describe an implementation of the Muzzle attack for which we developed a tool called *smartspoof* [2] to facilitate its execution:

(i) An attacker consists of a laptop with two wireless interfaces NIC_1, NIC_2, respectively. NIC_1 is set to master mode to intercept clients' association requests (rogue AP). It completely clones the legitimate AP, i.e., it operates on the same channel with the same link-layer address, and the same Basic Service Set ID (BSSID). Using NIC_2, the attacker floods the legitimate AP to reach its performance limits.
(ii) After the legitimate AP experiences operational anomalies, which are detected by monitoring its responses, the fake AP sends an impersonated deauthentication frame to force already associated clients to re-connect. While the network discovery procedure is equal to the one used by the legitimate AP, the association procedure and transmission of data frames is controlled by NIC_1.
(iii) NIC_1 intercepts all further requests (such as DHCP/DNS/ARP data) and responds with fake data to the client's network configuration queries. From this step onward, the Man-In-The-Middle attack can be launched against all security objectives.

[2] The source code used for this demonstration can be requested from the author.



(a) UDP throughput with fake AP. (b) TCP throughput with fake AP.

Fig. 2. Attack Implementation: UDP and TCP (carrying SSL protocol) traffic exchanged between a rogue AP and the client during hijacked association. Even if occasional deauthentications sent from a legitimate AP reset the hijacked link-layer connection, the client is promptly re-associated with the rogue AP and its TCP connection remains established (i.e., the application layer is unaware of disassociations) and the attack continues.

Since this attack is running on the same wireless channel, the questions that immediately come up are: how good is the impersonated association and can the legitimate AP be successfully sedated during the Man-In-The-Middle attack? To answer these questions we captured a client's communication with the rogue AP which is depicted in Fig. 2. Subfigure 2(a) shows UDP traffic, which serves as sampling method of the wireless channel, i.e., it shows how frequent the reactions from the legitimate AP are. As can be seen, the legitimate AP occasionally detects the impersonation and responds with a deauthentication frame which resets the client's association with the rogue AP. However, after step (2), the legitimate AP is kept under resource-exhaustion attack and the rogue AP is the first to answer all further re-association requests; thus attack continues. Moreover, the attack is taking advantage of the layered network design which requires abstraction and independent functionalities at each layer. Hence, the higher layer protocols, especially TCP, absorbs the frequent lower-layer re-connects and makes them invisible to the SSL. Once established, an SSL connection with a fake web server persists. This is shown in Subfigure 2(b), which depicts a single SSL session transferring user data between the user and the fake web server (we implemented a typical Man-In-The-Middle attack using fake certificates for a mutual SSL authentication). Actually, the behavior of the network is similar to an "ordinary" wireless connection with frequent signal losses which are not perceivable at the application layer. We have tested this scenario in our controlled environment, and testers were unable to detect any difference during their web activities such as using online banking or reading their emails. Using an intrusion detection system (in this case we tested the popular SNORT/WIDS) could not further help since no intrusion rule could be defined to identify the attack (the single effective rule would be to define the legitimate AP as rogue and hence deny its own wireless network).

This attack, although simple, can be launched against any IEEE 802.11 network. In contrast to the attack, the countermeasure is not that simple. Monitoring the wireless channel cannot reveal any suspicious activities. A single attack symptom is flooding with authentication requests. This can be detected, though except for turning down the complete wireless networks, there is no effective countermeasure that can protect legitimate clients and provide a legitimate service. Another approach would be to deploy the wireless network with APs containing higher hardware capabilities which are more resilient against such resource-depletion attacks. Clearly, this does not completely solve the problem and results in higher network deployment costs. In our opinion, this is an important fact because the low price of IEEE 802.11 technology is often considered to be one of its most mentioned advantages.

4. Design objectives of DiscoSec

To sustain the fast deployment of the IEEE 802.11 technology there is a long-term need for protection against various attacks resulting from unauthenticated management frames and resource depletion weakness, as shown in the previous section. While the resource-protection of APs can be provided by careful design of protocol execution, for the protection of management frames cryptography-based methods are required. However, adding key exchange and frame authentication to already performance-limited devices introduces a further computational burden. For this reason, care must be taken to avoid new vulnerabilities, and therefore we identified the following design objectives on which DiscoSec's protection should be based:

(i) *Simple and lightweight authentication protocol*
(ii) *DoS-resilient protocol execution*
(iii) *No alterations to the current IEEE 802.11 state machine.*

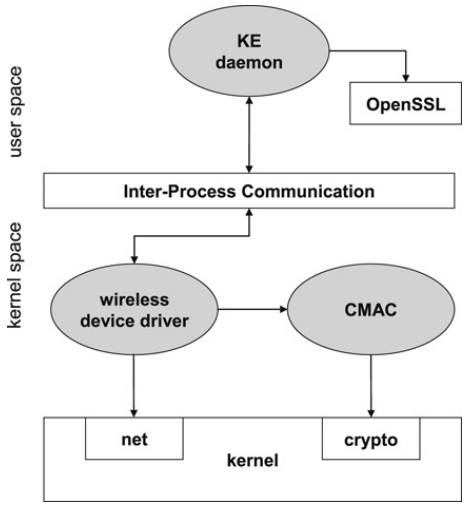


Fig. 3. Architecture of DiscoSec.

Simple and lightweight authentication is necessary for several reasons. Simplicity is a property affecting not only protocol design but also its implementation. In wireless communication where no assumption on a reliable channel should be made, protocols consisting of many round-trips (e.g., many message exchanges) often create deadlock vulnerabilities. The simplicity of authentication also assists us in reusing well-established cryptographic primitives available within the standard Linux (kernel) Crypto API and the OpenSSL library, thereby minimizing the potential for faulty implementation.

The *lightweight* property of an authentication protocol focuses on the key exchange phase where public key cryptography is used. To avoid many message round-trips we abandon the negotiation of security properties and rather utilize an anonymous Diffie–Hellman (DH) key exchange. The idea behind this decision is to shift the key exchange phase to the very beginning of the communication so that no resource reservation is made before the key exchange is finalized. At this stage no user identities are known but only their link-layer addresses, and therefore DiscoSec *binds* the sender's and receiver's link-layer addresses for the remainder of the session. Being protected from frame injection, devices can utilize identity authentication within the later stages of communication, relying upon more heavy-weight protocols. As a result, the key exchange is executed within only *one round-trip* (i.e. two messages) whilst supporting the second important design property—DoS-resilient protocol execution.

DoS resilience of DiscoSec concerns both computation- and memory-depletion attacks. The key exchange is the most vulnerable part of the authentication protocol and its arbitrary initiation should be avoided. For this reason we implement a rate limitation of key exchange requests which takes advantage of broadcast communication to provide fairness in the association process. The DoS protection is provided as a configuration parameter and its dimensioning can be adapted to comply with the performance characteristics of the dedicated AP.

To support legacy and DiscoSec protected stations within the same BSS, we designed and implemented DiscoSec *without changing the current IEEE 802.11 state machine*. All required information is embedded within the existing frames as *Information Elements* (IEs), i.e., the key-value data structure reserved by the IEEE 802.11 standard for the transmission of custom data. The authentication data is therefore only processed if DiscoSec is implemented; otherwise, it is simply discarded by the legacy driver. The frame structure remains unchanged, and a DiscoSec-enhanced AP has no impact on the association procedure for legacy stations.

Various other properties identified as performance vs. security tradeoffs are offered as configuration parameters of DiscoSec's implementation and can be adapted to the network requirements.

5. DiscoSec's design and implementation

The design of DiscoSec followed the requirements discussed in the previous section. The cryptographic primitives used as building blocks for the implementation of DiscoSec were chosen based on their performance properties. Decisions such as the choice of the underlying cipher, the use of a block cipher-based message authentication code (CMAC) and elliptic curve cryptography (ECC) for the key exchange were based on extensive measurements on dedicated APs (a more detailed discussion is given in Section 7).

5.1. Architecture of DiscoSec

The architecture of DiscoSec is depicted in Fig. 3. The functionality is split into modules and logically divided into three functional units: (i) the wireless LAN device driver that controls the WLAN hardware and contains the 802.11 network stack; (ii) the Key Exchange (KE) daemon which provides public key cryptography features utilizing primitives offered by the OpenSSL library: it processes the key exchange requests issued by the wireless device driver via Inter-Process

Table 2
Notation used in DiscoSec.

AP	Link-layer address of access point
STA	Link-layer address of wireless station
EC_{Param}	Elliptic curve parameters
p	Large prime
g	Primitive root modulo p
PK_{AP}	Access point's public-key
PK_{STA}	Station's public-key
MK	Master key computed from DH or ECDH
SK	Session key used for authentication
$MAC_{SK}(m)$	Message authentication code
AT	Association Token

Communication; and (iii) the CMAC kernel module which provides functions for calculating Message Authentication Codes (MACs) using the kernel's standard Crypto API.

The calculation of the MACs is a time-critical operation and is thus implemented in *kernel space*. In contrast to CMAC, the Key Exchange daemon runs in *user space* with a lower priority. In the case of a high CPU load, the Key Exchange daemon is scheduled less often, so the already associated stations are not influenced by expensive computations and their data throughput remains stable, as shown later in the performance section of this work.

5.2. Terminology and cryptographic primitives used

Table 2 shows the notation used in DiscoSec's key exchange and frame authentication. All exchanged variables are defined as custom Information Elements appended to existing IEEE 802.11 frames.

The DiscoSec key agreement protocol is based on Diffie–Hellman key exchange supporting both variants, the discrete logarithm, and the elliptic curve Diffie–Hellman (ECDH). The ECDH enjoys the advantage of much smaller key sizes, and hence is made as DiscoSec's default key exchange. However, because the DiscoSec key exchange scheme is kept as general as possible, the ECDH parameters can simply be replaced with a discrete logarithm DH. This way, the key exchange supports stations using earlier versions of the OpenSSL library which lack elliptic curve implementation. The choice of the DH version is offered to a user as DiscoSec's configuration parameter. The key lengths correspond to approximately equal security of both variants, but differ greatly in terms of performance (refer to Section 7 for performance measurements using both variants and different key lengths).

In the following, we discuss the default version of DiscoSec, i.e., the ECDH key exchange, and only briefly mention the discrete logarithm version. The public keys PK_{AP} and PK_{STA} are available in 128-bit length and are expandable to 256 bit. EC_{Param} defines an elliptic curve over a finite field supported by the OpenSSL library and is changeable through DiscoSec's configuration parameters.

The shared secret MK is computed from the ECDH key exchange using PK_{AP}, PK_{STA} and the station's and AP's private keys. It serves as a *master key* to derive key material for authentication.

The association token AT is used as *nonce* for computing a fresh session key SK for frame authentication, and additionally as DoS protection to control the rate of association requests (more information on tokens and DoS protection is given in Section 6).

The MAC_{SK} is a cipher-based message authentication code utilizing AES. We selected AES as an underlying cipher due to its availability within the IEEE 802.11i standard and good performance characteristics. It is provided within the Linux Crypto API (ver. 2.6+). The secure CMAC based on AES for authentication of messages with variable length was not available during the development of DiscoSec. Therefore we implemented RFC 4493 [17], which defines AES-CMAC and serves as a NIST recommendation for CMAC message authentication using the AES block cipher [18].

5.3. Association procedure—Key derivation

We omit the detailed description of a public/private key initiation and an ECDH shared secret computation. Both methods are standardized and their implementations are given by OpenSSL ver. 0.9.8+ [19,20]. In the following we describe the DiscoSec specific parameter exchange and the derivation of the authentication key.

The key exchange is accomplished within a single round-trip:

(i) $AP \rightarrow STA$: $\{PK_{AP}, EC_{Param}, AT\}$
(ii) $AP \leftarrow STA$: $\{PK_{STA}, AT\}$.

During start-up the AP initializes its key pair based on the elliptic curve parameters EC_{Param} (the AP's key pair can also be precomputed and loaded during start-up). The resulting PK_{AP} and EC_{Param} are sent to the stations via periodically emitted Beacon frames or a triggered Probe Response frame depending on either active or passive network discovery.

The wireless station extracts the supplied values, generates its key pair based on EC_{Param}, and computes the master key MK using the ECDH method. The session key SK is created by applying the cryptographic hash function *SHA-256* on the

MK and a previously received association token *AT*. The 128 least-significant bits of the hash are selected to provide the authentication key.

The reason for deriving the authentication key from the master key is to support a *key-caching* technique similar to the IEEE 802.11i standard. If the same wireless station decides to associate with the same AP and uses the static key pair, the computationally expensive *ECDH* key exchange can be omitted. The fresh *SK* is then derived by applying a single hash computation on the new association token and the cached *MK*.

The PK_{STA} and the *AT* are returned within the Authentication Request to the AP, which computes the *MK* and *SK* analogously to the station side. This finalizes the key exchange and both participants use their session key *SK* as secret key for AES frame authentication ($MAC_{SK}(m)$).

For the version of key exchange using the *discrete logarithm problem*, the parameters consist of the safe prime *p*, and the primitive root *g* which generates the finite field. *PrivKey* is a randomly chosen number within the finite field yielding *PubKey* after applying it as an exponent to *g*. Similar to its ECDH version, the public parameters broadcast within the Beacon frame are *p* and *g*, which are sent from the AP to the surrounding stations. The joining station extracts these parameters, generates its own PK_{STA}, and appends it to the Authentication Request. This is also the single difference between both variants used by DiscoSec.

However, key exchange is also the most critical part of the association procedure from the computational perspective. While the AP's key pair can be calculated offline and loaded during start-up, the session key derivation is triggered by an Authentication Request and its computation depends on PK_{STA} and *AT*. This opens a new vulnerability because the Authentication Request can easily be faked, and validation is only possible after the AP has derived *MK* by performing complex modular computations. If Authentication Request frames are received faster than the AP's transmission queue is processed, the AP can suffer from high frame loss and various operational anomalies, as demonstrated previously in this work. To prevent this kind of resource-depletion attack, DiscoSec provides a countermeasure based on an association rate control which is described in Section 6.

5.4. Frame authentication—Variable vs. fixed frame length

The challenge of frame authentication lies in its performance. While the low number of transmitted management frames only marginally increases the computational load, the authentication of each data frame significantly stresses performance-limited APs. Consequently, data frame authentication directly impacts the throughput of the wireless connection. For efficient frame authentication the most influential parameter is the frame's *size*. The Maximum Transmission Unit (MTU) of expected 1500 bytes is overrun in IEEE 802.11 networks, and currently frames up to 3000 bytes are transmitted over the wireless channel. The reason lies in proprietary features of various WLAN cards whose purpose is to increase throughput by frame aggregation. For example the *SuperG* [21] extensions of Atheros utilize the so-called *FastFrame* and *Bursting* techniques. FastFrame exploits the wireless channel more efficiently by increasing the amount of data contained within a single frame, i.e., it minimizes the frame overhead, while Bursting increases throughput by sending multiple frames within a single transmission opportunity. Although not standardized, these properties are common to various IEEE 802.11 vendors (under different names such as *108G Technology*, *Xtreme G*, *Turbo*) and their standardization should be finalized within the 802.11n standard.

The authentication of such frames can often present a computational burden for performance-limited APs. To be able to support these extensions DiscoSec provides two modes of authentication—the *full_auth* mode for APs with sufficient computational capabilities and the *fast_auth* mode for APs where full authentication of frames would result in a new performance bottleneck. Both modes are based on CMAC-AES authentication.

The *full_auth* mode supports authentication of frames with variable lengths, while the *fast_auth* mode limits the data included in the MAC to a fixed amount of 128 bits (both modes and authenticated frame fields are depicted in Fig. 4). In *fast_auth* mode only certain header fields are authenticated and the frame's payload is omitted from the computation. The authenticated fraction of *fast_auth* mode matches the block size of the AES cipher and can be authenticated within a single AES block computation (the *AT* of 4 bytes is included into the calculation, although not depicted in the figure). Accordingly, the authentication is more lightweight and may be performed much faster.

On the other hand, this presents a tradeoff between security and performance, i.e., a complete authentication vs. higher throughput. While in our opinion a meaningful on-the-fly manipulation of single bits during wireless transmission is hard to achieve and therefore *fast_auth* is sufficient for wireless communication, the decision on which mode to use is left to the user as a configuration parameter.

5.5. Replay protection

Frames transmitted over the wireless channel can easily be intercepted and used for replay attacks. To detect the resending of such frames, DiscoSec implements a replay protection by authenticating the frame's sequence numbers. The IEEE 802.11 generic frame format contains a sequence number field which is 16 bits long, out of which 4 bits are used for the fragment count and only 12 bits as the frame sequence.

In order to provide a frame counter sufficient for long sessions and to avoid resynchronization problems, DiscoSec implements an independent 32-bit sequence number (SEQ) field. It is included as an IE in each frame and used for MAC

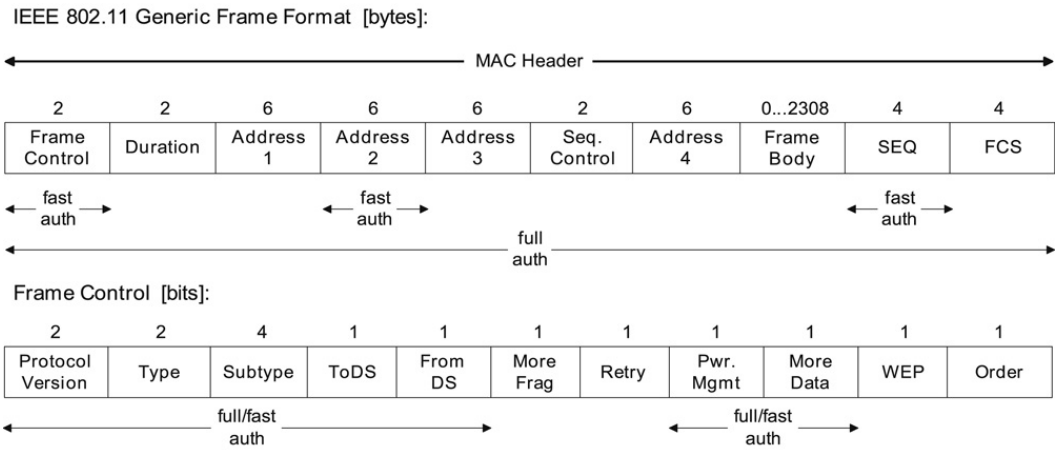


Fig. 4. DiscoSec authentication modes: authenticated frame fields in *full_auth* and *fast_auth*.

computation within both authentication modes (as shown in Fig. 4). The semantics of the legacy sequence number field (Sequence Control) remains therefore unchanged.

The verification is based on accepting a received sequence numbers within a *window:*

$$seq_{previous} < seq_{current} < seq_{previous} + window + 1$$

This way, false positive rejections of frames that are retransmitted due to loss or corruption are minimized. Frames containing a value less than the current sequence number are rejected. The magic number for the window length is usually selected around 10, which we also verified by real-world measurements. Since it obviously depends on the wireless environment, it can be changed in DiscoSec's configuration.

6. Attacks against DiscoSec

As shown in Section 3, performance-limited devices are prone to various resource-depletion vulnerabilities which can be used as a starting point against all security objectives. If there is no resource protection mechanism, the most severe attack against DiscoSec is achieved by flooding an AP with authentication requests containing a fake public key (i.e., an authentication request carrying randomly chosen values given as a station's public key). Each of such requests initiates an expensive shared secret computation on the AP. During the computation, the AP stores all further requests received but not yet processed; hence a performance-weak AP may easily exhaust its memory if the flooding rates are high. For example, one of the devices used for DiscoSec evaluation, a 4G-Systems AccessCube with CPU of 324 MHz and 64 MB RAM was successfully "broken" after only 2 min of flooding with a rate of 100 authentication requests per second. Clearly, the duration of the attack until the memory is exhausted depends on the hardware capabilities of the AP. Using a more powerful AP such as a laptop with 1400 MHz, the same attack showed no impact even after 1 h of duration. Nevertheless, similar to devices analyzed at the beginning of this work, if no resource protection is implemented even the security offered by DiscoSec can easily be used against them.

6.1. DiscoSec's resource protection

A straightforward protection against aforementioned attacks is to allow only a certain number of authentication requests and to block all further requests until the computational resources are available again. This solution would protect the AP, although it would leave less chance for a legitimate client to associate during an attack. Hence, we are interested in a solution which supports a fair chance of association among all clients.

The protection is based on using *association tokens*. The AP generates a set Θ_t containing association tokens which are 32-bit randomly chosen numbers. They are published at discrete time t and applied using the following concurrent phases:

- *publication phase* P_t—the AP broadcasts Θ_t to surrounding stations,
- *acceptance phase* A_t—the AP allows associations containing unused tokens from Θ_{t-1}.

Set Θ_t is *repeatedly* sent within Beacon frames for the duration of the time interval $[t, t+1[$. The Beacon frame additionally contains a *Counter Field* which reports the number of Beacon frames until $t + 1$.

The association proceeds as follows. After receiving a Beacon frame, the wireless station *randomly* chooses one token from Θ_t. It waits until the Beacon frame signals the beginning of $t + 1$ and then sends the authentication frame containing the chosen token. If the token has not been used the AP accepts the station's request and initiates the key exchange.

Using this mechanism, a successful authentication is independent of the time at which a station discovers the tokens. Each station knows the beginning of A_{t+1} and possesses Θ_t; therefore the authentication success probability is equal to the roughly fair medium access mechanism of IEEE 802.11. Choosing a random token helps legitimate STAs to increase their

Table 3
Evaluated platforms.

Device	CPU (MHz)	RAM (MB)	Kernel
Cube	MIPS, 324	64	2.6.14
Routerboard	Geode, 266	256	2.6.17
Laptop	Pentium 3, 1400	1024	2.6.17

chance of successful authentication and increases the cost of a successful attack. To ensure that no legitimate station can authenticate, an attacker would have to send all the tokens before the legitimate station sends its requests, and even then, the attacker must succeed for each published Θ_t.

The implementation of this protection is simple as it only requires Θ_{t-1} and Θ_t to be saved at the AP. The length of the Counter Field is 1 byte and Beacons are per default broadcast every 100 ms. The number of tokens within Θ_t depends on the performance characteristics of the dedicated AP.

During our measurements the performance-weakest AP could afford 10 authentications per second, i.e., every second the AP publishes 10 new tokens and accepts 10 tokens. It is important to mention that the tokens are only verified by the AP if the DoS Protection is enabled. On the other hand, when running with DoS Protection, only stations supporting the token mechanism can associate. This tradeoff is the unavoidable consequence of extending the AP's protection functionality.

While the primary objective to protect an AP's resources and assure its operational stability is fulfilled within this version of DiscoSec, more sophisticated techniques to differentiate between legitimate stations and attacker's requests are part of our current research [22].

7. Performance analysis of DiscoSec

The selection of platforms for testing DiscoSec's performance focused on hardware discrepancies in order to represent the computational capabilities of broadly available devices. Their hardware characteristics are shown in Table 3.

The performance-weakest device is a 4G AccessCube.[3] The device is from the year 2004 and runs on an architecture other than x86, thus making cross-compiling necessary. The other two devices are a Routerboard[4] 230 using Voyage Linux[5] 0.3 and a medium-class laptop operating in master mode of the wireless device driver, i.e., offering authentication and association procedures.

To provide insight into all authentication-related delays, DiscoSec was configured to protect both management and data frames. For throughput measurements we generated a continuous stream of UDP packets at various bit rates and under various AP utilizations. For measurements of key exchange and MAC computation we set the AP utilization to levels of 0%, 15%, 30% and 50% while monitoring the delay as a response variable. The utilization was increased either by using already associated clients sending with maximal throughput or artificially by additional CPU computations (if frame transmission did not result in high AP utilization). The measurements were repeated 10 times and the depicted results represent the mean with 0.95 confidence intervals.

In the remainder of the paper, the analysis of the key exchange and frame authentication is given for the performance-weakest device AP_{Cube}, while final throughput results of DiscoSec are provided for all three devices.

7.1. Costs of key exchange

Before going into details of the delays introduced by the key exchange, we briefly mention state-of-the-art delays imposed by the IEEE 802.11i security standard. For mutual identity authentication the security standard requires an Authentication Server that undertakes the shared secret computation instead of the AP. In [23] measurements show that the IEEE 802.11i delay imposed by the key exchange using mutual authentication (e.g., EAP-TLS) varies between ≈300 ms and 4 s, depending on different platforms and various implementations of the standard.

Concerning DiscoSec's key exchange, Fig. 5(a) and Fig. 5 depict the execution time using both variants of Diffie–Hellman under different key sizes and AP's utilizations.

Since the security of ECC keys (128 bit, 160 bit, 224 bit) complies with the strength of longer RSA keys (512 bit, 1024 bit, 2048 bit), we can directly compare the obtained measurement results.

The 128-bit elliptic curve key takes only ≈79 ms on the performance-weakest device. The varying AP utilization does not influence the key exchange much and at 50% utilization the 128-bit key exchange remains under 400 ms. Clearly, longer keys increase computational time, nevertheless even the key exchange using 224 bit keys (equivalent to a 2048-bit RSA public key) remains under 600 ms. Furthermore, the advantage of ECDH over the DH on the performance-weakest device becomes evidential. Using DH, the key exchange will take up to 1 s under 30% CPU utilization. Selecting stronger keys, such as a 2048-bit one, results in extremely high delays of 12 s.

[3] http://www.meshcube.org.
[4] http://www.routerboard.com.
[5] http://linux.voyage.hk/.

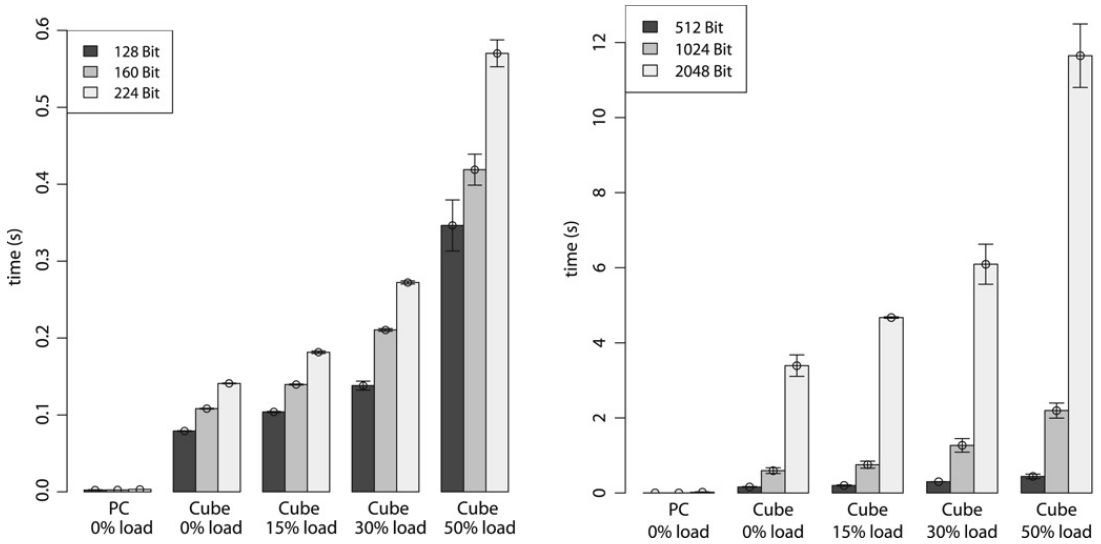


Fig. 5. Comparison of DiscoSec key exchange for different key sizes and under varying AP utilization: using ECDH (left) vs. the discrete logarithm version (right).

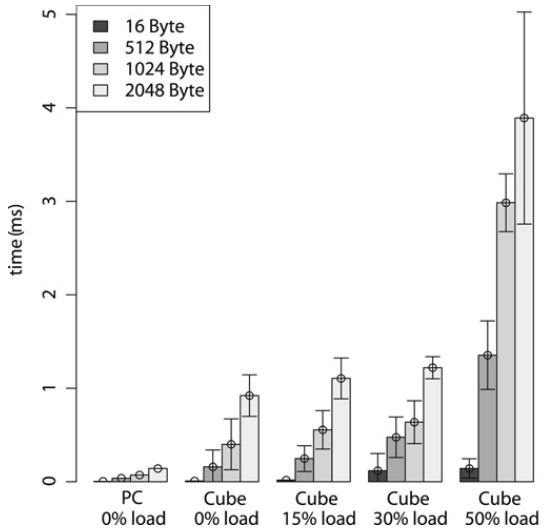


Fig. 6. Computational cost of implemented AES-CMAC.

Additionally, there are some initialization costs of both crypto systems. These costs are effective only at the start up of the AP when no stations are associated hence CPU utilization is at its minimum. So, for example, DH requires a large prime p and primitive root g. These static parameters are public and they can be pre-computed and simply loaded during the startup of the AP. Similarly, the parameters required for ECC are computed offline and loaded at startup. Concerning the initialization of ECC within the OpenSSL library, the process takes less then 6 ms even on the performance-weakest device.

7.2. Costs of frame authentication

In contrast to the shared secret which is generated once at the beginning of the client's session, the MAC is calculated for each transmitted management and data frame, implicitly influencing the connection throughput.

In order to evaluate the measurement results given in Fig. 6, it is important to consider the impact of MAC computation on the frame transmission. The maximum throughput of the AP_{Cube} is around 29 Mbit/s using the plain driver without any extensions. This means that every $\approx$433 μs a packet is transmitted. As a back-of-the-envelope calculation, if the MAC generation takes just as long, which includes the overhead imposed from the driver, the data rate will halve. On the tested hardware, processing 1024 bytes of data already takes $\approx$400 μs, hence not leaving much space. Nevertheless, the same figure shows that the computation time remains stable and less varying (given by the interval length of the confidence intervals) for all key sizes if the load is under 50%; otherwise the delay and its variance dramatically increase, exhibiting the device's computational limits. For this reason, the *fast_auth* mode becomes inevitable. Using *fast_auth* the computation of authenticated data equals 16 bytes, which complies with AES key length of 128 bits and does not exceed $\approx$150 μs even at

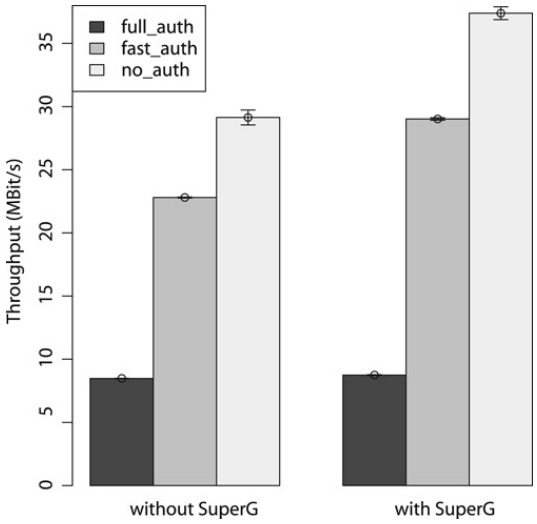


Fig. 7. Throughput using plan transmission vs. throughput-increasing features (SuperG).

50% CPU load. This significantly relieves the computational burden, resulting in much higher throughput, as shown in the next subsection.

7.3. DiscoSec featuring SuperG

The throughput comparison between plain and *SuperG*-enhanced transmission is depicted in Fig. 7 for three different configurations: *no_auth*, *fast_auth*, *full_auth*.

Considering the frame transmission without SuperG extensions, the *no_auth* bar denotes the maximum possible unauthenticated throughput of 29 Mbit/s equal to transmission without DiscoSec which serves as reference. Using *full_auth*, the complete IEEE 802.11 frame is authenticated. This security feature is computationally the most demanding and AP_{Cube} offers only 8 Mbit/s throughput.

Using *fast_auth* mode the AP relaxes the computational requirements and achieves data rates of ≈23 Mbit/s (78% of unauthenticated throughput). This scenario shows the importance of providing *fast_auth* as a tradeoff parameter for performance-limited APs.

Enabling the SuperG extensions (FastFrame and Bursting) leads to shorter transmission delays and larger frames, increasing the *no_auth* throughput to 37 Mbit/s. But more importantly, since SuperG does not impact the IEEE 802.11 header, the computational effort of *fast_auth* mode is equal to a transmission without SuperG features although more data is being transmitted. Therefore, even on a very performance-limited device like AP_{Cube}, using SuperG with *fast_auth* authenticated transmission is at ≈29 Mbit/s (78%).

7.4. Overall throughput

Until now, the presented analysis has focused only on the weakest measured device. By using the *fast_auth* mode, performance degradation can be mitigated, though not eliminated. The trend of modern APs aims at offloading computations of cryptographic primitives, especially symmetric ciphers, to specialized hardware. For example, the new generation of Geode CPUs features a hardware implementation of the 128-bit AES cipher and a true random number generator.

In our measurements we analyzed an older version of the Geode CPU within a *Routerboard 230*. It is a multifunctional device running at only 266 MHz (less than AP_{Cube}), without any special-purpose hardware for faster computations. It uses a Geode x86 SC1100 processor, equivalent to the Intel Pentium MMX architecture. Although its low CPU clock frequency does not allow for much faster computation, in *fast_auth* it achieves 97% of the possible throughput (see Fig. 8). The modest-looking 2%–3% throughput increase of *full_auth* mode compared with AP_{Cube} implies that Routerboard succeeds in authenticating ≈1 Mbit/s more data. Using hardware capabilities of an older Notebook running as an AP, it shows an exemplary authentication performance. The throughput of both *full_auth* and *fast_auth* mode is at 89% and 94%, respectively.

To summarize, this section provided an overview of what to expect from the network throughput using computationally-limited devices. It demonstrates the importance of identifying security vs. performance tradeoffs which in turn may smoothen throughput differences among heterogeneous hardware platforms.

8. DiscoSec's decision insights

Without using the *fast_auth* mode, the throughput of stations with limited computational power would evidentially suffer during complete per frame authentication. Similarly, the choice of the underlying cipher used for frame authentication played an important role for performance-oriented implementation. Hence, in this section we provide a brief overview of

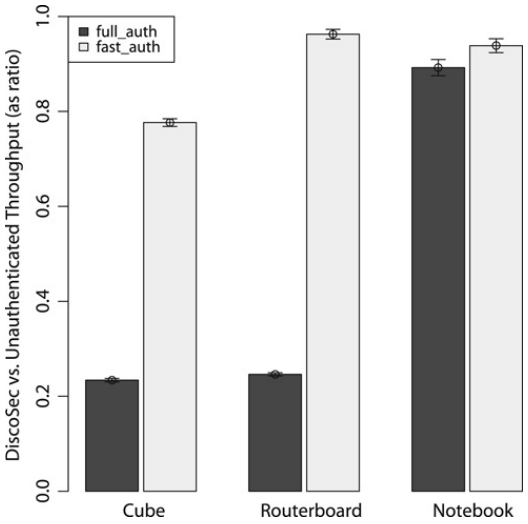


Fig. 8. Throughput analysis of all tested platforms.

Table 4
HMAC benchmarks. The values are in CPU cycles per byte for different plain text lengths, maximum standard deviation of 2 cycles per byte.

HMAC	16 bytes	256 bytes	1024 bytes	2048 bytes
MD5	948	33	16	14
SHA1	310	83	54	48
SHA2-512	1707	189	140	129
Tiger-160	477	123	60	56
Whirlpool-256	1109	199	168	161

Table 5
Cipher-based MAC benchmarks. Commonly used block ciphers implemented by Linux kernel Crypto module and their performance characteristics.

Cipher	Keysize	Blocksize	Cycles	CPB
Blowfish	64	16	2 499	157
	64	1024	89 845	87
	256	16	1 850	115
	256	1024	89 878	88
DES	64	16	2 177	137
	64	1024	64 678	64
3DES	192	16	6 926	433
	192	1024	187 613	184
Twofish	128	16	1 436	90
	128	1024	38 730	38
	256	16	1 501	94
	256	1024	41 241	41
AES	128	16	1 344	84
	128	1024	39 609	39
	256	16	1 530	96
	256	1024	44 290	44

tested ciphers selected from the Linux Crypto API (from ver. 2.6+) and the performance achieved by their implementations. The ciphers listed in Tables 4 and 5 were tested on the same Linux version as the one used for DiscoSec throughput analysis. The Tables describe the number of CPU cycles required for authentication, the two different frame sizes (used for *full_auth* and *fast_auth* mode), and resulting CPU cycles per byte (CPB) for the comparison of their relative costs.

After the key exchange both parties are in possession of a shared secret and are able to construct and verify Message Authentication Codes. A MAC can be computed using a stream cipher, keyed cryptographic hash functions also called HMACs, or block ciphers. Although the most flexible and efficient MAC computation is given by stream ciphers (i.e., they are simple and easily implemented in software), they are also the most sensitive ones. If the internal state of the stream cipher is repeated, serious security attacks become feasible. The most (in-)famous examples were attacks on confidentiality and integrity of Wired Equivalent Privacy (WEP), where default initialization of the RC4 stream cipher and long-term application of the secret key produced highly correlated values [7,6].

Another alternative to stream ciphers is the cryptographically strong (collision-resistant) hash functions (see Table 4). They are already in widespread use, especially for secure key derivation, i.e., they are usually applied on a master key or

shared secret to create various special-purpose keys. However, as Table 4 shows, their performance is less visible if short messages are used. The reason is their costly initialization, which must be done for each message, and thus applying them on short IEEE 802.11 management frames ($\approx$16 bytes) results in a high overhead and may significantly influence the network throughput. A further disadvantage of using HMACs, especially the MD5, is the recent security concerns [24]. Clearly, security can be enforced by increasing the hash value, such as in extended versions of HMACs collectively known as SHA-2 (e.g., SHA-512), although in this case the initialization costs are even higher and the performance further degrades.

By operating in Cipher-block chaining (CBC) mode, symmetric block ciphers produce ciphertext blocks dependent on all previous plaintext blocks, and thus the last block can serve as an MAC value. For comparison between HMACs and various available block ciphers, Table 4 lists results we collected by measuring their implementations offered by the Linux kernel. The Advanced Encryption Standard (AES) which is based on the Rijndael block cipher achieves high performance and offers simplicity within both hardware and software implementations. Although Twofish (which was selected as one of the final AES candidates) is also highly efficient, we decided to use AES because it is already part of IEEE 802.11i (WPA2) standard, and since an increased number of devices offers its support in their hardware.

Before a symmetric cipher can be utilized for MAC computation, its mode of operation must be defined. The Linux implementation is based on the CBC-MAC module. It is highly efficient due to a zero-copy network stack, i.e., it allows MAC calculation of received data without previously copying it. However, the CBC-MAC is secure only for fixed-length data. In the case of the authentication of complete IEEE 802.11 frames, varying sizes should be assumed, and therefore the *full_auth* mode offered by DiscoSec implements the recently published RFC 4493 (CMAC). This mode fixes the security deficiencies of CBC-MAC and also serves as a NIST recommendation for protecting authentication and integrity of data using symmetric ciphers.

In contrast to *full_auth*, the *fast_auth* mode uses a fixed size of 128 bits, which equals the AES block size. This saves the additional overhead such as checks of the data length, padding, or the data fragmentation, and results in higher throughput even using performance-limited devices.

9. Conclusion

The increasing number of commercial hotspots where web-based payments allow for a simple usage and an instant Internet access makes IEEE 802.11 networks attractive not only for sweet-tempered users. The broadcast nature of wireless communication and the unauthenticated IEEE 802.11 link-layer create fertile environments for launching various sophisticated attacks against all security objectives. This work addresses significant vulnerabilities that, although identified as purely performance-related problems, can easily serve to attack the already fragile wireless security. Using empirical analysis of various modern APs, we demonstrate the importance of maintaining the control over the wireless channel and the impact of simple resource-depletion attacks against widely deployed APs.

Secondly, this work describes and evaluates DiscoSec, a step toward secure and more dependable WLAN networks. The concept of DiscoSec followed the idea of "patching", i.e., providing a small, effective and easily applicable solution to a variety of devices. Since applying cryptographic methods may result in significant throughput problems or even new resource-depletion attacks, DiscoSec allows for parameter adjustment according to particular hardware capabilities of every device. As a result, its design supports balancing between security and performance tradeoffs.

Taking advantage of DiscoSec's protection, link-layer addresses of stations and APs are authenticated by lightweight ECC Diffie–Hellman key-exchange, adapted to the peculiarities of wireless communication. Using strong AES-based frame authentication which becomes effective during the association procedure, all further frame exchanges are immunized against popular injection attacks. To analyze its performance, especially the price paid in terms of overall network throughput, DiscoSec was implemented as an open-source device driver and was tested using different hardware platforms. Real-world measurements demonstrated that even a performance-limited device achieves 78% of the maximum throughput, while using more powerful hardware the throughput decrease is only 11% and 9% for full and fast authentication, respectively.

Acknowledgement

We gratefully acknowledge the *madwifi.org* project which was used as the basis for implementing DiscoSec.

References

[1] I. Martinovic, P. Pichota, M. Wilhelm, F.A. Zdarsky, J.B. Schmitt, Design, implementation, and performance analysis of discosec—service pack for securing WLANs, in: 9th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WOWMOM 2008), Newport Beach, CA, USA, June 2008.
[2] IEEE 802.11i/D10.0, Security enhancements, amendment 6 to IEEE Standard for Information Technology, IEEE Standard, April 2004.
[3] J. Bellardo, S. Savage, 802.11 Denial-of-Service Attacks: Real Vulnerabilities and Practical Solutions, in: Proceedings of the USENIX Security Symposium, 2003, pp. 15–28.
[4] C. He, J.C. Mitchell, Security Analysis and Improvements for IEEE 802.11i, in: Proceedings of the 12th Annual Network and Distributed System Security Symposium (NDSS'05), February 2005, pp. 90–110.
[5] W.A. Arbaugh, S. Shankar, J. Wang, K. Zhang, Your 802.11 Network has No Clothes, in: Proceedings of the First IEEE International Conference on Wireless LANs and Home Networks, December 2001, pp. 15–28.

[6] N. Borisov, I. Goldberg, D. Wagner, Intercepting mobile communications: The insecurity of 802.11, in: MobiCom '01: Proceedings of the 7th Annual International Conference on Mobile Computing and Networking, July 2001, pp. 180–189.
[7] S. Fluhrer, I. Mantin, A. Shamir, Weaknesses in the key scheduling Algorithm of RC4, in: SAC'01: Revised Papers from the 8th Annual International Workshop on Selected Areas in Cryptography, August 2001, pp. 1–24.
[8] A. Bittau, M. Handley, J. Lackey, The final nail in WEP's coffin, in: SP'06: Proceedings of the 2006 IEEE Symposium on Security and Privacy (S&P'06), IEEE Computer Society, Washington, DC, USA, 2006, pp. 386–400.
[9] R. Floeter, Wireless Lan security framework: void11, 2002. http://www.wlsec.net/void11.
[10] AirJack, 2003. http://sourceforge.net/projects/airjack/.
[11] D. Faria, D. Cheriton, DoS and authentication in wireless public access networks, in: Proceedings of the 2004 ACM Workshop on Wireless Security, September 2002, pp. 47–56.
[12] L. Ma, X. Cheng, A.Y. Teymorian, A hybrid rogue access point protection framework for commodity wi-fi networks, in: Proceedings of the 27th Annual IEEE Conference on Computer Communications (Infocom), April 2008.
[13] www.cisco.com Document ID: 82196 Infrastructure Management Frame Protection (MFP) with WLC and LAP configuration example (last accessed 11.02.08).
[14] AirDefense, www.airdefense.net (last accessed 06.08.08).
[15] AirMagnet, www.airmagnet.com (last accessed 06.08.08).
[16] 3Com AirProtect, www.3com.com (last accessed 06.08.08).
[17] J. Song, R. Poovendran, J. Lee, T. Iwata, The AES-CMAC Algorithm, RFC 4493 (Informational), June 2006.
[18] NIST Special Publication 800-38B, Recommendation for block cipher modes of operation: The CMAC mode for authentication, May 2005.
[19] IETF, Diffie–Hellman key agreement method, RFC 2631, 1999.
[20] SECG, Elliptic Curve Cryptography, Standards for Efficient Cryptographic Group, Available at: www.secg.org/collateral/sec2.pdf (last accessed 13.03.08).
[21] Atheros, SuperG—Maximizing wireless performance, Available at: www.super-g.com (last accessed 02.03.08).
[22] I. Martinovic, F.A. Zdarsky, M. Wilhelm, C. Wegmann, J.B. Schmitt, Wireless client puzzles in IEEE 802.11 networks: Security by wireless, in: Proc. ACM Conference on Wireless Network Security (WiSec 2008), Alexandria, VA, USA, March 2008, pp. 36–45.
[23] I. Martinovic, F. Zdarsky, A. Bachorek, J. Schmitt, Introduction of IEEE 802.11i and measuring its security vs. performance tradeoff, in: Proceedings of the 13th European Wireless Conference, France, April 2007.
[24] X. Wang, H. Yu, How to break MD5 and other hash functions, in: EUROCRYPT, 2005, pp. 19–35.