Virtual Prepaid Tokens for Wi-Fi Hotspot Access

Haidong Xia and José Brustoloni*
Department of Computer Science, University of Pittsburgh
210 S. Bouquet St. #6135, Pittsburgh, PA 15260, USA
{*hdxia, jcb*}*@cs.pitt.edu*

Abstract

We introduce virtual prepaid tokens (VPTs), a novel billing scheme that allows users to obtain access at Wi-Fi hotspots without having an account with a hotspot provider or a physical prepaid token (PPT). Upon arrival at a hotspot, a user buys a VPT online, using a third-party payment server with which the user already has an account. Experiments show that users can buy a VPT and gain full Internet connectivity in less than 15 seconds, i.e. much less time than it would take to create another account or to buy and activate a PPT. VPTs can be used in hotspots that use a captive portal or 802.1x for user authentication. The latter alternative enables better security. We also contribute a novel technique that allows a single access point to authenticate users by either method. Hotspots can use this solution for migrating to 802.1x without disrupting legacy captive-portal users. Experiments demonstrate that the proposed technique has little overhead and interoperates broadly.

1. Introduction

Wi-Fi hotspots are expected to have an important role in future provisioning of "anywhere, anytime" connectivity [1, 2]. They are quickly being deployed at locations that tend to attract nomadic users, such as cafés, airports, hotels, and conference centers. Although hotspots have limited range, they offer lower installation costs and higher bandwidth than do competing alternatives, such as 3G wireless.

However, many hotspots have low utilization and are unprofitable [3]. This low utilization is not due to incompatibility (many users' notebook computers and PDAs have a Wi-Fi interface) or other technologies' dominance (3G deployment has been slow in most markets). The observed unprofitability could limit growth in the deployment of Wi-Fi hotspots.

Billing is often cited as a problem area that contributes to low hotspot utilization [3]: Existing billing methods have drawbacks that turn away many potential users. Three of the most common methods are subscription, pay-per-use account, and prepaid token. *Subscriptions* give to the provider a steady revenue stream and to the user the convenience of a fixed price and single monthly payment. However, subscriptions are nontrivial commitments. Several concerns may militate against such a commitment, including user doubts about whether he or she will need access in a covered area often enough to justify the cost of a subscription. Users can also be concerned about provider reliability.

Instead of a subscription, users may set up a *pay-per-use* account with a provider. Pay-per-use accounts typically draw funds automatically from one of the user's bank or credit card accounts, when the user gains access. Pay-per-use accounts can be less wasteful than are subscriptions to sporadic users. However, many users hesitate to open such an account with a provider that is not perceived as reliable and well-established in areas frequented by the user. Many providers are startups that do not meet such criteria. Moreover, a user may occasionally need access in places that are not served (directly or by agreement) by any of the providers that serve areas more frequently visited by the user.

In the latter cases, users may prefer *prepaid tokens* (PPTs). PPTs contain an id and password that are typically revealed by scratching a card and are activated after first use for a limited time. A user does not need to set up any account to buy such a token; payment may be, e.g., by cash or credit card. Prepaid tokens offer little risk to users. However, such tokens can complicate access because they need to be physically obtained from a vendor. In many cases (e.g., at an airport), vendor location may be inconvenient or not obvious. Moreover, a vendor location may be closed when a token is needed.

*This project was funded in part by the Pittsburgh Digital Greenhouse through a grant from the Commonwealth of Pennsylvania, Department of Community and Economic Development.

This paper contributes an alternative billing architecture using *virtual prepaid tokens* (VPTs) and experimentally evaluates its performance. Users can buy a VPT from a provider without any relationship between them before or after a specific access session. Users buy VPTs at the point and time of access, using a third-party online payment server. Users can employ the same server also for making or receiving many other types of payment. Therefore, such an account is more flexible than is a conventional pay-per-use account, which can be used only to purchase access from a specific provider or set of providers. Like physical prepaid tokens, VPTs do not require users to maintain a possibly wasteful subscription with the access provider. However, because VPTs are bought online, they have several advantages relative to PPTs, including saved time and no need of staffing outlets for selling them.

The main difficulty in VPT implementation is that most current hotspot architectures authenticate a user *before* authorizing *any* Internet access by the user. VPT purchases require, however, that *unauthorized* users communicate with payment servers on the Internet. The VPT architecture accommodates such communication while blocking all other Internet access by unauthorized users. Communication between user and payment server is secured end-to-end by SSL. The payment server authenticates the user, debits the user's credit or bank account for the price of access, and credits that amount to the provider. After verifying payment, the provider authorizes full Internet access by the user.

VPTs involve more steps than do the password-based authentication schemes typically used for subscriptions and pay-per-use accounts. On the other hand, our experiments show that users arriving at a hotspot can buy a VPT and gain full Internet connectivity in less than 15 seconds, i.e. much less time than it would take to buy and activate a physical token.

VPTs can be used in hotspots that employ a captive portal [4] or 802.1x [6] to authenticate users. Most current hotspots use a captive portal, but 802.1x enables much better security. In particular, 802.1x enables mutual authentication between user and hotspot and per-session encryption keys at the link layer. Although the VPT architecture is secured end-to-end by SSL, 802.1x can provide a valuable additional line of defense at the link layer.

This paper also contributes a novel scheme that allows a single access point to support users authenticated by captive portal or 802.1x. Using this scheme, hotspots can introduce 802.1x without disrupting service to legacy users. The main difficulty for this integration is how to support broadcast packets, given that not all clients may be using link-layer encryption. Our solution consists in broadcasting such packets twice, once with encryption and once without it. Our experiments show that this solution has little overhead and interoperates with a broad range of Wi-Fi cards.

The rest of this paper is organized as follows. Section 2 describes how hotspots typically use a captive portal to authorize access by users with a subscription, pay-per-use account, or physical prepaid token. Section 3 explains how a captive portal can be modified so that it supports also VPTs. Section 4 discusses attacks against captive portals and summarizes our previous work on defenses against them. Section 5 reviews 802.1x operation. Section 6 introduces a new method that permits an access point to support captive portal and 802.1x authentication at the same time. Section 7 explains how VPTs can be acquired by users authenticated by 802.1x. Section 8 presents our experimental results, and Section 9 concludes.

2. Captive portals for users with subscription, pay-per-use account, or physical prepaid token

This section describes how hotspots typically authenticate and authorize access by users who have a subscription or pay-per-use account with the provider, or an acceptable physical prepaid token.

Hotspots typically do not use WEP, Wi-Fi's original security scheme. WEP would require matching secret keys to be manually configured in access points and user computers. Such keys would be difficult to keep secret in public settings, such as hotspots.

Instead, hotspots usually employ *captive portals* for user authentication. Captive portals were first proposed in Stanford's SPINACH project [4] and are illustrated in Fig. 1. Captive portals do not require special configuration of user computers. A special gateway between the hotspot's LAN and the Internet enables only authorized users to communicate with the Internet. The gateway distinguishes authorized and unauthorized users by their MAC and IP addresses. The gateway allows unauthorized users' DHCP, ARP, and DNS query packets. Unauthorized users use DHCP to obtain networking configuration parameters. They are then expected to open a Web browser and send a Web request. The gateway redirects any Web requests from unauthorized users to a captive portal, and drops any other unauthorized packets. The captive portal returns to the user an SSL-secured login page that requests the user's id and password. The captive portal verifies the latter and, in case of success, sends the user's MAC and IP addresses to the gateway for authorizing the user's Internet access. The captive portal usually also sends the user a session

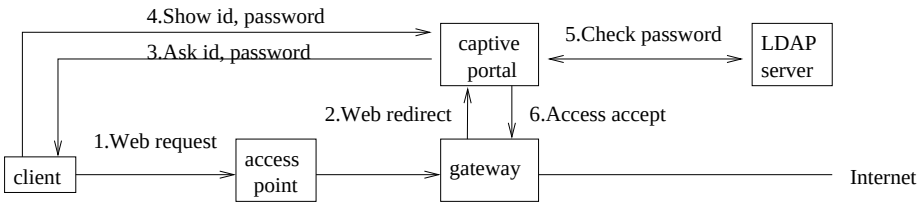

Figure 1. Captive portals provide an intuitive Web-based interface for user authentication. No special configuration is necessary in user computers.

management page with a button for logging off, on a small popup window that is not used for browsing. Finally, the captive portal redirects the user to the Web page that the user initially requested (note that the initial redirection by the gateway makes it unnecessary for the user to know the captive portal's URL).

Captive portals typically communicate with a remote account database for authenticating user passwords. Any of several protocols may be used for such communication, e.g., RADIUS, LDAP, or Kerberos. In the case of physical prepaid tokens, the database would have been previously populated with temporary accounts containing user ids and passwords that match those on the tokens. Upon first authentication of such an account's user, the database manager calculates and updates the respective account's expiration time.

3. Captive portals for users with virtual prepaid token

This section explains how captive portals can be modified so that they support not only the billing options discussed in the previous section, but also VPTs, as illustrated in Fig. 2.

The SSL-secured login page that the captive portal sends to the user is modified so that it contains an area where users who do not have a valid password can order a VPT. In the latter case, the user enters the respective user id and password and selects an expiration time and online payment server (OPS), possibly from among several alternatives displayed as buttons. The captive portal reserves in the account database the entered user id. In case of success, the captive portal sends the bill to the selected OPS and redirects the user to the OPS. The gateway is modified so that it allows unauthorized users to communicate with the supported OPSs. The selected OPS authenticates the user and asks the user to confirm payment of the provider's bill. After user confirmation, the OPS debits the bill's amount from the user's account and credits the same amount, minus OPS fees, to the provider's account. If the user's account does not carry enough balance, the OPS withdraws the bill's amount from the user's credit card or bank account. After crediting the provider, the OPS notifies the provider's captive portal. The captive portal establishes the user's account in the database and sends the user's MAC and IP addresses to the gateway for authorizing the user's Internet access.

4. Captive-portal vulnerabilities and defenses

This section discusses vulnerabilities of captive portals and alternatives for defense.

Although most current commercial hotspots use a captive portal for user authentication, the resulting security does not prevent theft of service. In particular, it is easy for an attacker to *hijack* a session of an authenticated user [5]. The hijacker periodically sends to an authorized user deauthentication or disassociation notifications purported to come from the access point. These notifications cause the authorized user to stop transmitting. The hijacker can then use the user's IP and MAC addresses to communicate with the Internet.

The increasing use of personal firewalls enables a simpler attack, *freeloading* [5]. Unlike hijacking, freeloading does not require special tools and can easily go undetected. A freeloader simply uses an authorized user's MAC and IP addresses while the user continues to communicate. Freeloading would ordinarily not be expected to work reliably, because the user receives packets actually destined to the freeloader and may respond in ways that disrupt the freeloader's communication. For example, the standard response to a TCP packet that belongs to a connection that, from the point of view of the receiver, is closed, is to send a RST segment to the sender, thereby possibly aborting freeloaders' connections. However, a personal firewall inhibits responses to stray packets, including RST segments (the firewall interprets stray packets as attempts to fingerprint the node's software or find vulnerable ports). When both the freeloader and authorized user

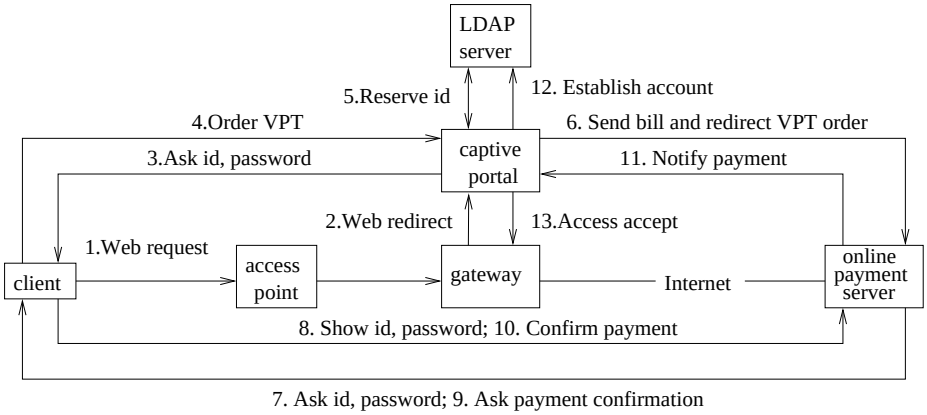

Figure 2. Users can employ an account with a third-party online payment server to buy a virtual prepaid token for hotspot access. User and hotspot need not have any relationship before or after access.

have personal firewalls, freeloading works reliably and can be used in collusion against the hotspot provider.

In another paper, we proposed novel defenses against hijacking and freeloading [5]. *Session id checking* secures the session management page with SSL, associates with the page a non-persistent cookie containing a cryptographically random session id, and tags the page with a refresh directive and some period. The directive periodically causes the user's browser to send to the captive portal a request to refresh the session management page. Such requests are automatically SSL-secured and accompanied by the cookie. The captive portal can thus detect hijacking by noticing that it has not received a correct refresh request from an authorized user for more than the refresh period. The captive portal then unauthorizes the user's addresses. *MAC sequence number tracking* detects freeloading by noticing that MAC-layer sequence numbers form more than one trend line, corresponding one to the authorized user and the remaining to freeloaders. The access point then causes the user's MAC and IP addresses to be unauthorized.

Although session id checking and MAC sequence number tracking improve the security of captive portals, 802.1x can provide a higher level of security. Unfortunately, 802.1x may require the installation and configuration of new hardware and software in user computers, and the respective standards are still in a state of flux. Because technical support at a level that most hotspots cannot provide would be necessary, hotspots cannot currently mandate use of 802.1x. This paper contributes a novel scheme, described in Section 6, that allows hotspots to enable 802.1x security while still supporting captive-portal users.

5. Wi-Fi security with 802.1x

This section reviews 802.1x-based Wi-Fi security. Readers familiar with this topic may skip this section.

IEEE 802.1x is an extensible protocol for authenticating clients before they are granted access to a local area network (LAN) [6]. Three parties are involved in 802.1x authentication: a *supplicant*, which is the client that desires access; an *authenticator*, which is the node that mediates the client's access to the LAN, and an *authentication server*, which authenticates supplicants. For example, in typical Wi-Fi LANs [1, 2] using 802.1x, mobile stations are supplicants, access points are authenticators, and RADIUS [7] servers are used as authentication servers.

IEEE 802.1x supports a variety of authentication schemes. For example, 802.1x can be used with EAP-TLS [8], whereby servers and supplicants authenticate each other using digital certificates. Alternatively, 802.1x can be used with PEAP [9], a recent proposal that uses digital certificates to authenticate servers to supplicants and can use passwords (e.g., via MS-CHAP-V2 [10]) to authenticate supplicants to servers. PEAP may be considerably simpler and less expensive to deploy than is EAP-TLS, because PEAP does not require supplicants to acquire and maintain certificates.

IEEE 802.1x also supports dynamic per-session keys for encrypting or authenticating packets sent between supplicants and authenticators. Dynamic per-session keys can make it significantly harder to break Wi-Fi's original packet encryption scheme, WEP (Wired Equivalent Privacy) [11, 12, 13, 14, 15]. Nonetheless, new Wi-Fi security standards combine 802.1x with stronger cryptography. The interim new standard, WPA (Wi-Fi

Protected Access) [1], uses frame encryption (TKIP) and authentication (Michael) algorithms that can be implemented as firmware upgrades on existing network interface cards (NICs) and access points. IEEE 802.11i [16], on the other hand, will use AES (Advanced Encryption Standard) [17] and therefore will need more powerful NIC and access point hardware.

In 802.1x, supplicants (i.e., clients) are by default considered unauthorized. Supplicants in this state can send or receive only EAPOL (EAP [18] over LAN) frames. After association, an 802.1x supplicant sends an EAPOL-Start frame to the authenticator (i.e., access point). The authenticator responds by requesting the supplicant's identity. The authenticator then translates and relays a sequence of responses and requests between the supplicant and the authentication server. Communication between the supplicant and authenticator uses the EAPOL protocol. On the other hand, communication between the authenticator and the authentication server uses the RADIUS protocol [7, 19]. RADIUS can be configured to secure packets using a secret key. The sequence of requests and responses in this phase will depend on the particular authentication scheme used (e.g., EAP-TLS or PEAP). In the case of PEAP, first a TLS [20] tunnel is established between the supplicant and authentication server. As is usual in TLS, the supplicant authenticates the server using the server's certificate, which the server sends to the supplicant via TLS. The TLS tunnel secures the subsequent authentication of the supplicant. A variety of methods can be used for authenticating the supplicant; the method currently supported by Microsoft is MS-CHAP-V2, which is password-based.

After the authentication server authenticates and accepts the supplicant, cryptographic keys are established and communicated. The precise steps involved in this phase are still being standardized. In Microsoft's existing implementation, both authentication server and supplicant derive from the TLS session key an MS-MPPE-Send-Key and an MS-MPPE-Recv-Key [21, 22]. The server includes these keys as vendor extensions in the RADIUS access accept packet that the server sends to the authenticator when the server accepts the supplicant [19]. After forwarding the corresponding EAP-Success frame to the supplicant, the authenticator sends two EAPOL-Key frames to the supplicant. The first frame contains the broadcast key that the authenticator uses to broadcast packets on the LAN. The second frame contains a unicast key for communication between the supplicant and the authenticator. In each EAPOL-Key frame, the unicast or broadcast key is encrypted using the MS-MPPE-Recv-Key, and the entire frame is signed using the MS-MPPE-Send-Key. The unicast and broadcast keys are used by cryptographic algorithms (e.g., WEP) for securing subsequent data frames. The authenticator can be configured to suppress unicast keys, in which case supplicants and authenticator use the broadcast key for all data frames. The authenticator can also be configured to refresh keys periodically.

After authentication and key exchange, the supplicant moves to the authorized state, and previous restrictions on supplicant communication are removed.

6. Supporting both users authenticated by captive portal and 802.1x

This section describes a novel method that allows a single access point to support both users authenticated by captive portal (who do not use link-layer encryption) and by 802.1x (who do use link-layer encryption).

To make this possible, the gateway functionality necessary for captive portals, as discussed in Sections 2 and 3, is integrated into the access points. The access points keep track of the authentication state of each associated supplicant (i.e., client). Supplicants are by default (e.g., right after association) considered unauthorized. In this state, supplicants can communicate only using EAPOL, DHCP, ARP, DNS queries, and HTTPS with a captive portal. A supplicant's state changes to authorized when the respective access point receives from an 802.1x authentication server or captive portal a corresponding access-accept packet. The latter packet specifies a session_timeout value. After that interval has elapsed, the supplicant's state reverts to unauthorized.

Communication between access points and 802.1x authentication servers is according to the RADIUS protocol, which includes access-accept and access-reject packets. Authentication using 802.1x proceeds as described in Section 5, i.e., there is no difference with respect to 802.1x-only LANs.

Communication between access points and captive portals, on the contrary, does not use RADIUS. However, captive portals send authenticators packets with the same format as RADIUS access-accept and access-reject packets.

If an unauthorized supplicant sends any HTTP or HTTPS request, the respective access point redirects the request to a captive portal. Captive portals are SSL-secured and ask for and verify the supplicant's credentials (e.g., id and password). Captive portals and 802.1x authentication servers may use the same account database (implemented, e.g., by an LDAP server). After authenticating a supplicant, a captive portal sends to the supplicant's access point an access-accept packet. Access-accept packets may specify different filter_id

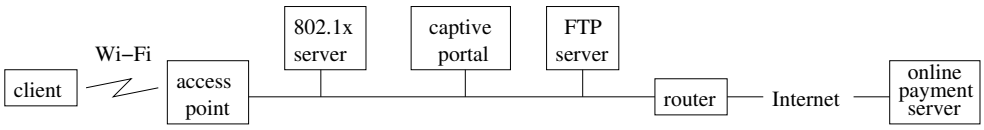

Figure 3. Testbed used in the experiments. The proposed techniques involve changes only in the access point and captive portal.

values according to supplicant class (as defined in the account database) and how the supplicant was authenticated (802.1x or captive portal). The access point can be configured to use different firewall rules or VLANs for forwarding packets from or to each authorized supplicant, according to the supplicant's filter_id.

The access point keeps in a record each associated supplicant's unicast key. In the case of an 802.1x supplicant, the access-accept packet also specifies an MS-MPPE-Send-Key and an MS-MPPE-Recv-Key, as discussed in Section 5. The access point uses these keys for communicating to the supplicant a unicast and broadcast key for securing subsequent data frames.

On the other hand, in the case of a captive-portal supplicant, the unicast key is null, indicating that the access point does not encrypt, decrypt, or authenticate packets that it forwards to or from the supplicant. This case achieves backward compatibility with the default configuration of most existing Wi-Fi NICs and access points.

The access point also keeps track of the number of associated 802.1x and captive-portal supplicants, respectively. When the access point needs to send to its associated supplicants a broadcast packet, the access point sends the packet secured with the broadcast key, if there are any associated 802.1x supplicants; and sends the packet without security if there are any associated captive-portal supplicants (if there are both 802.1x and captive-portal supplicants, the access point sends broadcast packets twice). Commonly, the only packets that may need to be broadcast twice are DHCP and ARP.

Finally, the access point's beacon frame is set so that the Capability field's Privacy bit is not set (some client Wi-Fi cards will not associate without static WEP if this bit is set, regardless of client configuration).

7. Using virtual prepaid tokens with 802.1x-based security

This section briefly explains how VPTs can be used with 802.1x-based security.

To enable 802.1x paying visitors, a special, well-known account may be defined in the account database (e.g., an account with id "visitor", null password, no expiration date, and a special filter_id with access restrictions similar to those of unauthorized clients). This account's dummy credentials allow an 802.1x client to obtain session keys for secure Wi-Fi communication. The client then starts a Web browser and sends a Web request. Given that the client's rights are still those of an unauthorized client, the access point redirects the request to the captive portal. The client can then purchase a VPT as described in Section 3, with the differences that 802.1x has authenticated the hotspot to the client and link-layer encryption is used between the client and access point. After the client completes the purchase, the captive portal sends a new access-accept packet to the respective access point. The access point then changes the client's status to authorized and upgrades the client's access rights. This upgrade is automatic and does not require the client to input the respective id and password.

8. Experimental results

This section presents experimental results from a prototype implementation of the proposed techniques. Fig. 3 shows the testbed used. Reported averages are the result of five tries.

We implemented on an IBM ThinkPad T30 1.8 GHz PC an access point with integrated gateway functionality, as described in Sections 2, 3, 6, and 7. This PC contains an integrated Wi-Fi interface based on the Intersil Prism 2.5 chipset. It ran Linux 2.4.20 and a modified version of the HostAP 0.0.3 Wi-Fi driver [23]. The modifications necessary for our techniques increased HostAP's code by roughly 32 KB. Additional memory is necessary for maintaining session status. For 50 sessions, the additional table memory needed is about 1 KB. These results suggest that the proposed techniques can be adopted in memory-constrained access points.

Testbed clients were two other T30 1.8 GHz PCs running Windows XP Professional SP 1 and Internet Explorer 6.0.28 configured to remember passwords and auto-fill forms (to reduce variations due to data entry). The 802.1x clients were configured to use the PEAP protocol and dynamic WEP. The 802.1x authentication

Table 1. Breakdown of VPT purchase latency

Payment step	Latency (s)
Visitor orders VPT from captive portal	0.6
Captive portal sends bill to online payment server (OPS) and redirects visitor to OPS	2.6
Visitor inputs id and password and sends to OPS	2.3
OPS verifies visitor id and password and asks payment confirmation	1.4
Visitor confirms payment	0.6
OPS processes payment and notifies captive portal	6.8
Total	14.3

server used was a Dell Dimension 4550 2.4 GHz PC running Windows 2000 Server SP 3 with 802.1x patch, IAS RADIUS server, and Active Directory. The captive portal and FTP server were two identical Dell 4550 PCs running Linux 2.4.20. The captive portal used Apache 2.0.40.

In the first experiment, we measured the time necessary for a visitor to purchase a VPT and gain full access. We used PayPal as a representative online payment server. We set up a consumer account for the client and a business account for the prototype hotspot. PayPal fees for these types of account are respectively none and $0.30 plus 2.9% per payment [24]. The business account was configured to send instant payment notifications (IPNs) to the captive portal, so as to avoid delays. The captive portal communicates with PayPal using SSL to verify IPNs. Table 1 breaks down the total latency into its constituent parts. The total latency measured was 14.3 s, which is probably acceptable for occasional users (frequent and highly mobile users may prefer a subscription, which allows faster authorization). Of this total, 10.8 s were spent by the online payment server.

In the next set of experiments, we measured the latency necessary for client authentication, and the TCP throughput and `ping` round-trip time (RTT) between client and FTP server, under the following conditions: captive-portal or 802.1x authentication with unmodified or modified HostAP. To obtain good resolution, we calculated latencies and RTT by examining `ethereal` traces of the relevant packets. We used the `ttcp` tool to measure throughput. The results are displayed in Table 2 and show that the proposed techniques have negligible effect on performance. (Captive portal corresponds to lower authentication latency and higher TCP throughput because, respectively, it does not authenticate the server and does not activate link-layer encryption.)

In the final set of experiments, we exhaustively tested the modified access point's interoperation with multiple clients and different client Wi-Fi interfaces. We configured both clients with captive-portal or 802.1x authentication or each client with a different authentication method. We also configured either client to use the integrated Prim 2.5 Wi-Fi interface while the other client uses a Netgear, Orinoco Gold, Cisco Aironet 350, or the integrated Prism 2.5 Wi-Fi interface. (The Cisco interface needed to be configured to "allow association to mixed cell;" the other interfaces used do not have an equivalent configuration option.) We verified that both clients operated correctly in all of the tested configurations.

9. Conclusions

Support for VPTs and simultaneous captive-portal and 802.1x authentication can be added to an access point with little extra memory, negligible overhead, and good interoperability with various clients.

For occasional hotspot users, VPTs offer several advantages relative to other billing options. Unlike a subscription, VPTs do not have a monthly carrying cost. Instead of a pay-per-use account with the provider or aggregator, which can be used only to purchase access at certain hotspots, VPTs use an account with a third-party online payment server, which allows the user to employ the account for any other payments as well. And unlike physical prepaid tokens, VPTs allow a user to order and obtain Internet access from a provider in less than 15 s, even if the user has no previous or subsequent relationship with that provider or that provider's aggregator.

Simultaneous support for captive-portal and 802.1x authentication allows hotspots to provide recent Wi-Fi security improvements to those clients whose configuration supports them, without disrupting legacy clients. Improvements include mutual authentication between client and hotspot (e.g. using EAP-TLS or PEAP) and link-layer packet encryption and authentication with dynamic per-session keys (e.g. using dynamic WEP, WPA, or 802.11i). These improvements can provide a valuable extra line of defense against attacks in the Wi-Fi network. However, hotspots typically can provide

Table 2. Access point modifications to support VPTs and simultaneous captive-portal and 802.1x authentication have negligible impact on performance (standard deviations represented between brackets)

Auth. method	Access point	Authentication latency (ms)	TCP throughput (KB/s)	Round-trip time (ms)
802.1x	unmodified	660 [120]	556 [9]	3.5 [0.5]
	modified	700 [110]	555 [6]	3.3 [0.5]
Captive portal	unmodified	48.3 [0.4]	598 [8]	2.9 [0.2]
	modified	48.7 [0.6]	593 [8]	3.0 [0.2]

only minimal technical support and therefore may not be able to mandate clients to upgrade to the more secure 802.1x-based authentication. In such circumstances, backward compatibility with captive portals provides an attractive trade-off between security and usability.

References

[1] Wi-Fi Alliance. `http://www.weca.net/`

[2] IEEE, "Wireless LAN medium access control (MAC) and physical layer (PHY) specifications," 1999. `http://standards.ieee.org/getieee802/download/802.11-1999.pdf`

[3] A. Stone, "For-Fee Hot Spots Strive to Make Wi-Fi Pay," *Pervasive Computing*, IEEE, pp. 3-7, July 2003.

[4] G. Appenzeller, M. Roussopoulos, and M. Baker, "User-Friendly Access Control for Public Network Ports," in *Proceedings of INFOCOM*, IEEE, pp. 699-707, March 1999.

[5] H. Xia and J. Brustoloni, "Detecting and Blocking Unauthorized Access in Wi-Fi Networks," in *Proceedings of IFIP Networking'2004 Conference*, Springer-Verlag, Lecture Notes in Computer Science, 3042:795-806, May 2004.

[6] IEEE, "Port-based network access control," 2001. `http://standards.ieee.org/getieee802/download/802.1X-2001.pdf`

[7] C. Rigney, S. Willens, A. Rubens, and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)." RFC 2865, June 2000.

[8] B. Aboba and D. Simon, "PPP EAP TLS Authentication Protocol." RFC 2716, Oct. 1999.

[9] A. Palekar, D. Simon, G. Zorn, , and S. Josefsson, "Protected EAP protocol (PEAP)." Internet Draft, Mar. 2003.

[10] G. Zorn, "Microsoft PPP CHAP Extensions, Version 2." RFC 2759, Jan. 2000.

[11] N. Borisov, I. Goldberg, and D. Wagner, "Intercepting mobile communications: The insecurity of 802.11," in *Proc. Seventh Annual International Conference on Mobile Computing and Networking (MOBICOM)*, pp. 180–188, ACM, 2001.

[12] W. Arbaugh, N. Shankar, and J. Wang, "Your 802.11 network has no clothes," in *Proc. First IEEE International Conference on Wireless LANs and Home Networks*, Dec. 2001.

[13] S. Fluhrer, I. Mantin, and A. Shamir, "Weaknesses in the key scheduling algorithm of RC4," in *Eighth Annual Workshop on Selected Areas in Cryptography*, Aug. 2001.

[14] A. Stubblefield, J. Ioannidis, and A. Rubin, "Using the Fluhrer, Mantin, and Shamir attack to break WEP," Tech. Rep. TD-4ZCPZZ, AT&T Labs, Aug. 2001.

[15] Airsnort. `http://airsnort.shmoo.com/`

[16] IEEE, "Specification for enhanced security," unapproved draft for 802.11i. `http://standards.ieee.org/getieee802/new.html`

[17] NIST, "Specification for the Advanced Encryption Standard (AES)," Federal Information Processing Standards Publication 197. `http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf`

[18] L. Blunk and J. Vollbrecht, "PPP Extensible Authentication Protocol (EAP)." RFC 2284, Mar. 1998.

[19] C. Rigney, W. Willats, and P. Calhoun, "RADIUS Extensions." RFC 2869, June 2000.

[20] T. Dierks and C. Allen, "The TLS Protocol Version 1.0)." RFC 2246, Jan. 1999.

[21] G. Pall and G. Zorn, "Microsoft Point-To-Point Encryption (MPPE) Protocol." RFC 3078, Mar. 2001.

[22] G. Zorn, "Deriving Keys for use with Microsoft Point-To-Point Encryption (MPPE)." RFC 3079, Mar. 2001.

[23] J. Malinen, "Host AP driver for Intersil Prism2/2.5/3." `http://hostap.epitest.fi`

[24] PayPal. `http://www.paypal.com/`