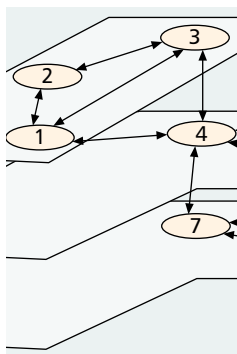


PROACTIVE KEY DISTRIBUTION USING NEIGHBOR GRAPHS

ARUNESH MISHRA, MIN HO SHIN, NICK L. PETRONI, JR., T. CHARLES CLANCY, AND
WILLIAM A. ARBAUGH, UNIVERSITY OF MARYLAND



Neighbor graphs can be utilized to obtain a 99 percent reduction in the authentication time of an IEEE 802.11 handoff (full EAP-TLS) by proactively distributing necessary key material one hop ahead of the mobile user.

ABSTRACT

User mobility in wireless data networks is increasing because of technological advances, and the desire for voice and multimedia applications. These applications, however, require that handoffs between base stations (or access points) be fast to maintain the quality of the connections. In this article we introduce a novel data structure, the Neighbor Graph, that dynamically captures the mobility topology of a wireless network. We show how neighbor graphs can be utilized to obtain a 99 percent reduction in the authentication time of an IEEE 802.11 handoff (full EAP-TLS) by proactively distributing necessary key material one hop ahead of the mobile user. We also present a reactive method for fast authentication that requires only firmware changes to access points and hence can easily be deployed on existing wireless networks.

INTRODUCTION

Wireless networks, specifically those based on the IEEE 802.11 standard (WiFi), are experiencing rapid growth due to their low cost and unregulated bandwidth. As a result of this tremendous growth, pockets of connectivity have been created similar to the first few years of cellular systems. The logical next step for WiFi-based networks is support for fast roaming within the same administrative domain and then eventually between different administrative domains. Finally, we expect that roaming between networks of differing physical layers (i.e., vertical handoffs) will occur once multimode — WiFi and Global System for Mobile Communications/code-division multiple access (GSM/CDMA) — handsets become more available, which in turn will change how WiFi networks are used.

Previous studies of wireless network mobility have shown that users tend to roam in what we call *discrete mobility* where the user utilizes the network while stationary (or connected to the same base station). The user ceases operation before moving and continues to use the network after moving to a new location. That is, the users do not usually move while using the network because the majority of current network applica-

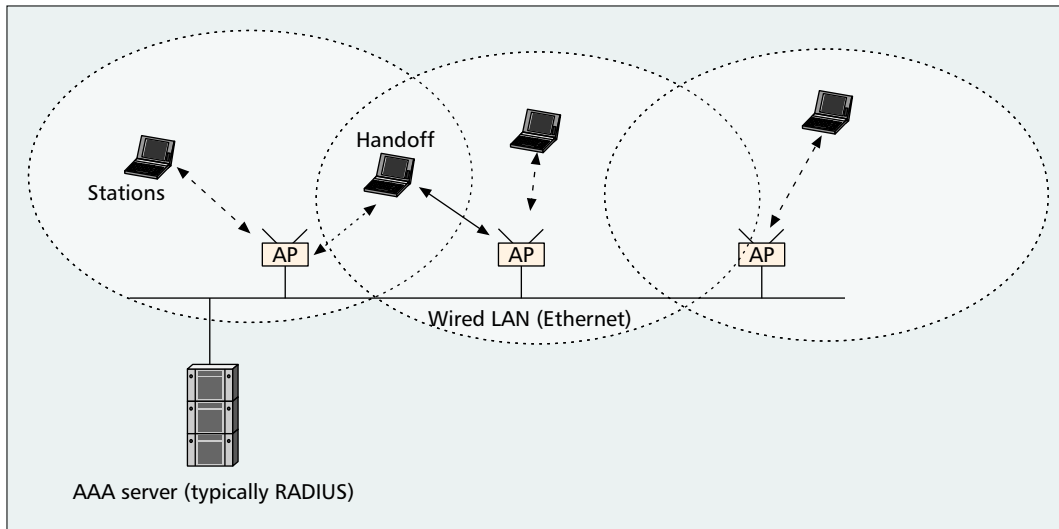
tions and equipment do not easily lend themselves to what we call *continuous mobility*, where the user moves while utilizing the network [1].

Voice-based applications are the predominant application in continuous mobility, as seen in the current cellular networks, and we expect voice and multimedia applications to serve as the catalyst for continuous mobility in WiFi networks much as they did for cellular networks once multimode handsets and end-user applications become more widely available.

Supporting voice and multimedia with continuous mobility, however, implies that the total latency (layers 2 and 3) of handoffs between base stations must be small. Specifically, the overall latency should not exceed 50 ms to prevent excessive jitter. Unfortunately, the vast majority of WiFi-based networks do not currently meet this goal with layer 2 latencies contributing approximately 90 percent of the overall latency, which exceeds 100 ms [2, 3].

In the context of wireless LANs, the term handoff refers to a station (any device with an 802.11 interface) moving its association and state information (key material) from one access point (AP) to another. Logically, a wireless handoff is composed of four phases: probe, decision, association, and authentication. In the probe phase, the mobile station seeks to identify a candidate set of next APs via active or passive means. Once the candidate set of next APs has been identified, the station selects the next AP and performs any needed housekeeping (flushing buffers, etc.) in the decision phase. Next, the mobile station begins the association phase with the selected AP. Finally, authentication or reauthentication is completed [2].

In this article we focus on improving the authentication delay incurred during a horizontal handoff *within the same administrative domain*. That is, we focus solely on reducing the authentication delay when a station moves from one AP to another — a layer 2 handoff — in order to enable applications that facilitate the concept of continuous mobility. The current draft for the IEEE 802.11 security architecture recommends that this authentication process be completed using Extensible Authentication Protocol-Transparent Layer Security (EAP-TLS),



■ **Figure 1.** Typical topology of a wireless LAN.

which has been included as the default authentication method in Windows XP. Unfortunately, a complete EAP-TLS handshake, including RADIUS messages, requires on the order of 1 s — a number far too large to support any form of streaming media. To answer this question, the IEEE included “Pre-authentication” in the draft, which permits a mobile station to “pre-authenticate” itself to the next AP (see the related work section for a more complete description). Unfortunately, pre-authentication has several shortcomings. First, a station can only pre-authenticate to another AP on the same LAN (i.e., the station cannot authenticate beyond the first access router as a single administrative domain might have multiple access routers). Second, a full EAP-TLS authentication to all potential next APs is not a scalable solution in terms of the number of stations and the APs as most networks use a centralized authentication server (RADIUS) that can quickly become a bottleneck. This obviously prevents WiFi networks from reaching much of the previously discussed vision.

To solve this problem, we designed, implemented, and tested a solution that only requires only small changes at the authentication, authorization, and accounting (AAA) server and the AP, and supports all the same security properties as EAP-TLS and pre-authentication but at significantly reduced latencies. Combining our key distribution method with a novel algorithm for dynamically identifying and maintaining the mobility topology of the network, *Neighbor Graphs*, results in an efficient key distribution method that amortizes the cost of the initial EAP-TLS authentication across all handoffs within the same administrative domain without loss of security. By using proactive key distribution, we reduced the latency of the authentication phase from an average of 1.1 s to an average of 25 ms. Moreover, our concept of proactive key distribution using neighbor graphs can easily be extended to other authentication mechanisms (PEAP, EAP-SIM, EAP-AKA, etc.) used by the AAA server.

The drawback of the above scheme is that it

requires changes (firmware updates) at the user, the AP, and the AAA server. In contrast to the proactive scheme, we later present a reactive scheme for fast authentication that can easily be deployed (without changes to existing APs) until proactive solutions can be integrated into commercial equipment.

The rest of this article is organized as follows. We provide a brief overview of the IEEE 802.11 security architecture (the IEEE 802.11i standard). We discuss our neighbor graph data structure, and how it can be constructed and maintained by the network autonomously. We then discuss the proactive key distribution method using the neighbor graphs as a vehicle. We present implementation results from an in-building wireless network testbed. We also present a novel reactive approach for fast authentication. A detailed description of related techniques to reduce authentication latency, and a discussion on proactive key distribution and context transfer is provided. Finally, we conclude and discuss future enhancements.

IEEE 802.11i AUTHENTICATION OVERVIEW

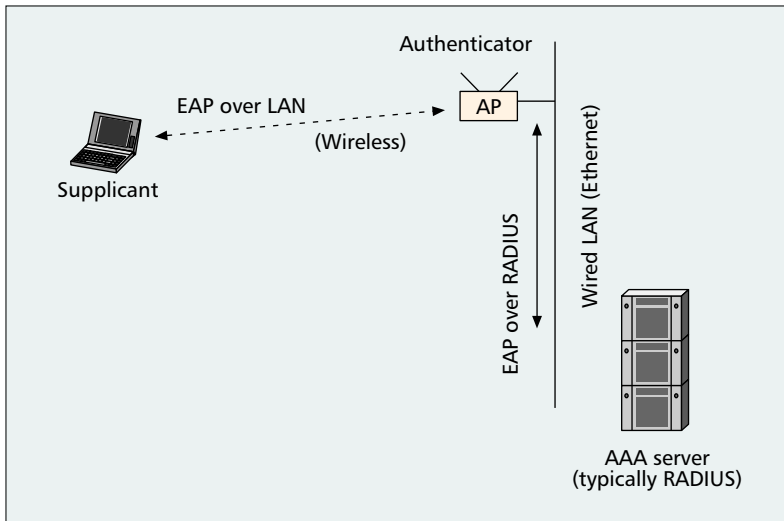
The authentication framework developed by the IEEE Task Group I — Security (TGi) is a complex combination of several different protocols. While a thorough understanding of each of these protocols is not required, basic knowledge of each will assist in understanding the problems we are addressing as well as our solution.

Figure 1 shows the architecture of a wireless LAN. The APs are typically connected together to a backend authentication server (AAA server such as RADIUS) over an Ethernet (or virtual LAN, VLAN). As in any architecture, the trust assumptions are key to the correct operation of the system. TGi makes the following trust assumptions:

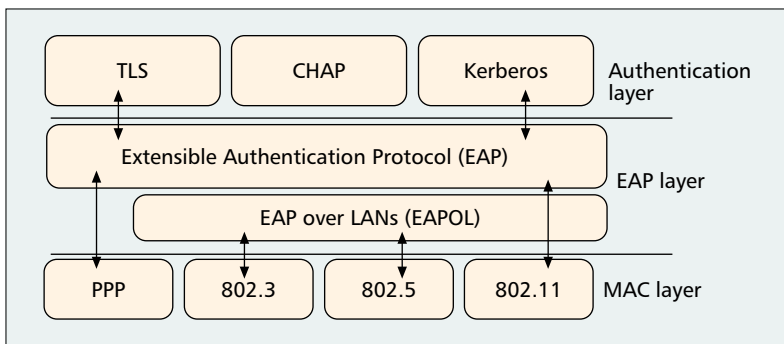
- The AAA server is trusted.
- The AP to which a mobile station is associated is trusted; nonassociated APs are not trusted.

These assumptions, which are different from

The IEEE 802.1X standard provides an architectural framework to facilitate network access control at the link layer for various link technologies. The standard abstracts the notion of three entities: the supplicant, the authenticator or network port, and the authentication server.



■ Figure 2. The entities in an IEEE 802.1X setup.



■ Figure 3. The EAP stack.

those in a cellular network, are due to the nature of 802.11 equipment. Access points are low-cost devices often placed in locations that lack proper physical security. Therefore, it is important to prevent the compromise of a single AP permitting a compromise of the entire network.

IEEE 802.1X

The IEEE 802.1X [4] standard provides an architectural framework to facilitate network access control at the link layer for various link technologies (IEEE 802.11, FDDI, token ring, IEEE 802.3 Ethernet, etc.). The standard abstracts the notion of three entities: the *supplicant*, the *authenticator* or *network port*, and the *authentication server*. Figure 2 shows the typical communication setup. A supplicant is an entity that desires to use a service (link layer connectivity) offered via the notion of a *port* on the *authenticator* (e.g., a switch or an AP). Thus, for a single network there will be many ports through which supplicants can authenticate themselves and obtain network access. An authenticator is in control of a set of ports, and a network might have multiple authenticators. As an example, an Ethernet switch can be an authenticator that controls network access on multiple physical Ethernet ports available on the device. In the IEEE 802.11 scenario, a port corresponds to an association between a supplicant and the authenticator (AP).

The supplicant authenticates via the authenticator to a central *authentication server* that directs the authenticator to provide access after successful authentication. Typically, the authentication server and authenticator communicate using the *Remote Authentication Dial-In User Service* (RADIUS) protocol. The RADIUS protocol contains mechanisms for per-packet authenticity and integrity verification between the AP and the RADIUS server, although these measures are not as strong as desired.

The authentication process between the authentication server and the supplicant (via the authenticator) is carried over EAP, which is described in the following section.

EXTENSIBLE AUTHENTICATION PROTOCOL

The IEEE 802.1X standard employs the *Extensible Authentication Protocol* to permit a variety of authentication mechanisms. Figure 3 shows the protocol layers for communication between the supplicant and the authenticator. EAP is built around the challenge-response communication paradigm. There are four types of messages: *EAP Request*, *EAP Response*, *EAP Success*, and *EAP Failure*. The EAP Request message is sent to the supplicant indicating a challenge, and the supplicant replies using the EAP Response message. After multiple exchanges of the Request/Response messages, the EAP Success/Failure message is used to notify the supplicant of the outcome. A multitude of authentication methods can be encapsulated in the EAP protocol, most notably EAP-TLS, EAP-MD5, EAP-AKA, and EAP-SIM. We discuss the TLS authentication method in further detail in the next section.

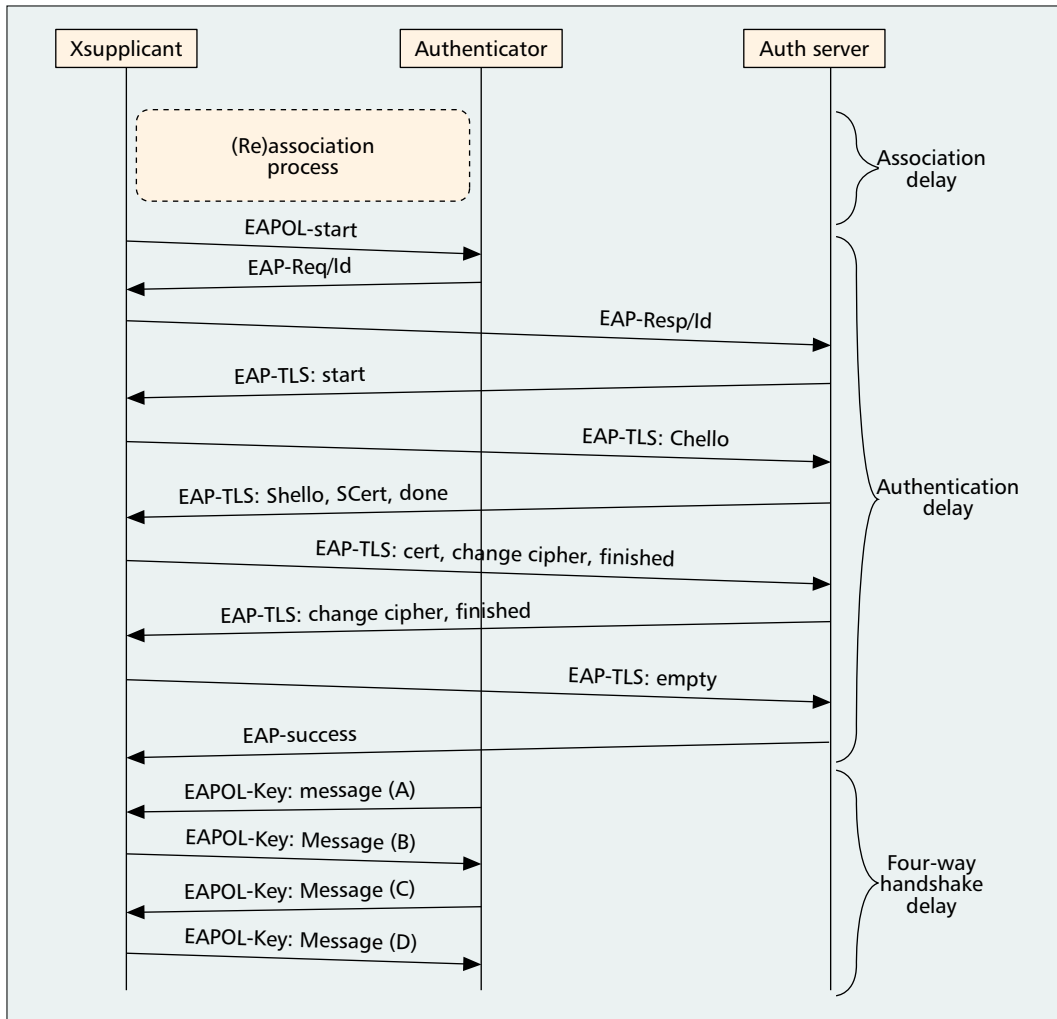
The EAP messages do not have an addressing mechanism and are encapsulated over *EAP Over LAN* (EAPOL) [4] protocol between the supplicant and the authenticator, and are carried as a RADIUS attribute between the authenticator and the authentication server. EAPOL also provides for a four-way handshake mechanism, discussed later.

TRANSPORT LAYER SECURITY

The Transport Layer Security protocol as described in RFC-2246 provides strong authentication and encryption at the transport level. It is divided into two protocols: the *handshake* protocol, which derives strong key material, and the *record* protocol, which performs the data transfer. The authentication part of TLS has been exported as an authentication mechanism over EAP in EAP-TLS RFC 2716. This is the most commonly used authentication mechanism over EAP within 802.11-based networks, and fits into the IEEE 802.1X model. Figure 4 shows the complete set of messages exchanged during a full EAP-TLS authentication.

In the application of TLS to IEEE 802.1X, the supplicant and authentication server have a certificate from a common trusted certificate authority (CA). The mutual authentication process based on these credentials achieves the following:

- Mutual authentication of the client and server
- A strong shared secret master key (MK)



■ **Figure 4.** The complete set of messages exchanged during the (re)association process. In particular, it shows the EAP-TLS authentication messages and the four-way handshake.

The Transport Layer Security protocol as described in RFC-2246 provides strong authentication and encryption at the transport level. It is divided into two protocols: the handshake protocol, which derives strong key material, and the record protocol, which performs the data transfer.

- An initialized set of pseudo-random functions (PRFs) that can be utilized to generate further key material

Let TLS-PRF denote the PRFs generated as a result of the authentication. The MK is used to derive a pairwise master key (PMK) by using Eq. 1.

$$PMK = TLS-PRF(MK, clientHello.random \mid serverHello.random) \quad (1)$$

The PMK is used along with certain cipher methods to derive four pairwise transient keys (PTKs) that are used for various purposes as shown in Fig. 5 [5]. The EAPOL-MIC and the EAPOL-Encr. keys are used to provide data origin authenticity and confidentiality for the four-way handshake discussed later. The other two keys are used for link layer encryption and authenticity depending on the cipher suite being employed.

FOUR-WAY HANDSHAKE

The IEEE 802.11 Task Group I defines an IEEE 802.1X protocol called a four-way handshake. This protocol is used to confirm the liveness of the AP and the station (STA), guarantees the freshness, and synchronizes the shared session

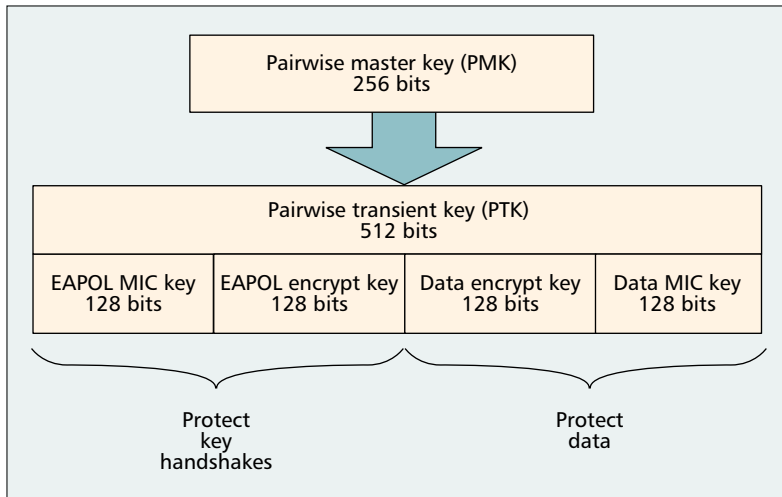
key and binds the PMK to the medium access control (MAC) address of the STA. The communication is carried using EAPOL key messages [6].

Message (A) Authenticator → Supplicant — This is the first EAPOL-Key message and is sent from the authenticator to the supplicant. It contains ANonce — a nonce value generated by the authenticator. Once the supplicant has received this message it can compute the four transient keys.

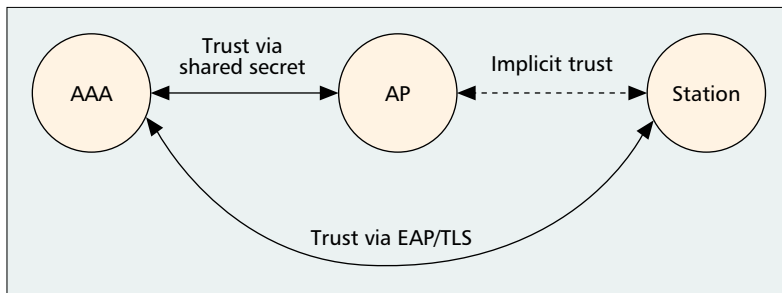
Message (B) Supplicant → Authenticator — This message contains SNonce — a supplicant generated nonce and an MIC over the message to protect its integrity. The authenticator uses SNonce to generate the transient keys, and verifies the MIC.

Message (C) Authenticator → Supplicant — This message includes the earlier ANonce and an MIC check that can be verified by the supplicant proving that the authenticator has a matching PMK.

Message (D) Supplicant → Authenticator — This message signifies the completion of the four-way handshake and signals the installation of the keys by both entities for data communication.



■ Figure 5. The key structure: PMK and the derived PTK.



■ Figure 6. The trust relations in TG.

The four-way handshake protocol is used during full authentication and reauthentication; hence, this cost (i.e., the overhead incurred) will be present in both situations. We also do not include the cost of the handshake in the timings of EAP-TLS. In this work we do not implement the handshake for the above reason; instead, we have implemented a simpler two-way handshake mechanism for demonstration purposes.

TG TRUST RELATIONSHIPS

One of the interesting and disappointing problems with TG's new security architecture is the trust relationships in an operational network. Many people believe that the AP is a trusted party, and this is not completely correct.

Figure 6 depicts the trust relationships within TG. The solid arrows represent an explicit mutual trust relationship, while the dotted line represents an implicit trust relationship that *must* be created in order to make security claims about the communications path. This trust relationship between the AP and the STA is transitive, and derived from the fact that the station trusts the AAA server and the AAA server trusts the AP. This, unfortunately, is not ideal since in many cases the trust relationship between the AAA server and the AP will not exist if shared keys are not used to protect the RADIUS traffic. However, the majority of the AP vendors in TG had a strong desire for an inexpensive AP that was more of a relay than a participant in communications.

PROPERTIES OF A SUCCESSFUL AUTHENTICATION

After the successful completion of the EAP-TLS authentication phase the following statements hold:

- The mobile station's identity has been proved.
- Based on the above identity, the mobile station's access to the network has been granted by the AAA server.
- The mobile station and AAA server share a strong master secret, the MK.
- The mobile station, AAA server, and associated AP all share a common secret, a PMK, derived from the MK.
- A session key, the PTK, is derived from the PMK using the four-way handshake and is only shared between the mobile station and the associated AP.

NEIGHBOR GRAPHS

In this section we describe the notion of the neighbor graph data structure and the abstractions these graphs provide. Neighbor graphs are used to determine the candidate set of APs with which a roaming STA could potentially reassociate. Usually this candidate set is a small fraction of the total number of APs forming the wireless network. Hence, schemes that proactively transfer STA context and key material to this candidate set of APs prior to reassociation become feasible.

DEFINITIONS

Reassociation relationship: Two APs, say, ap_i and ap_j , are said to satisfy a reassociation relationship if it is possible for a STA to perform an 802.11 reassociation through some path of motion between the physical locations of ap_i and ap_j .

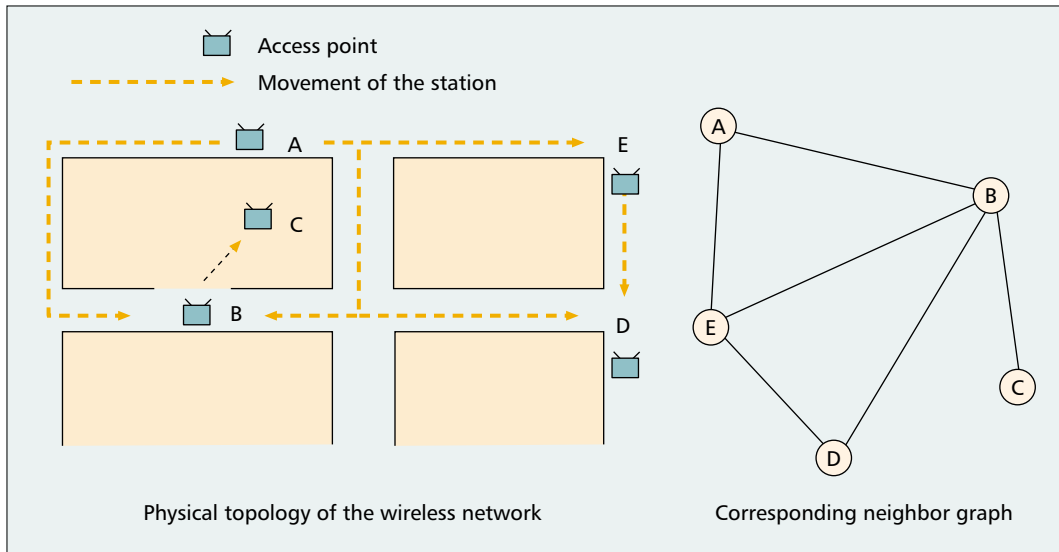
Consider the placement of APs in a simple in-building scenario as shown in Fig. 7. The dotted lines show a potential path of motion. APs A and E satisfy the reassociation relationship, because there exists a path of motion (as can be seen from the figure) by which an STA can reassociate between A and E.

The reassociation relationship depends on the placement of APs, signal strength, and other topological factors, and in many cases corresponds to the physical distance (vicinity) between the APs. The reassociation relationship between APs forms the basis for the construction of the neighbor graph data structure, as discussed below.

AP neighbor graph: Define an undirected graph $G = (V, E)$ where $V = \{ap_1, ap_2, \dots, ap_n\}$ is the set of all APs (constituting the wireless network under consideration), and there is an edge $e = (ap_i, ap_j)$ between ap_i and ap_j if they satisfy a reassociation relationship.

Association pattern: Define the association pattern $\Gamma(c)$ for client c as $\{(ap_1, t_1), (ap_2, t_2), \dots, (ap_n, t_n)\}$, where ap_i is the AP to which the STA reassociates (new AP) at time t_i , and $\{(ap_i, t_i), (ap_{i+1}, t_{i+1})\}$ is such that the handoff occurs from ap_i to ap_{i+1} at time t_{i+1} ; the STA maintains continuous logical network connectivity from time t_1 to t_n .

The neighbor graph and association pattern



■ **Figure 7.** An example placement of APs and the corresponding neighbor graph.

are related according to the following observation. We define the *locality of mobility* principle to state that for a client c with association pattern $\Gamma(c)$ as defined above, the neighbor graph $G = (V, E)$ captures the locality (of motion) in the association pattern; that is, for any two successive APs, say, ap_i and ap_{i+1} in $\Gamma(c)$, the edge $e = (ap_i, ap_{i+1}) \in E$. This concept of locality is the abstraction captured by the neighbor graph as a data structure.

IMPLEMENTATION ISSUES

The neighbor graph can be autonomously learned and maintained by a wireless network without the need for any manual configuration. Also, the data structure can be maintained either in a distributed fashion by the APs themselves [7] or in a centralized manner at the authentication server, as in this article. In this application of neighbor graphs for proactive key distribution, we construct and maintain the data structure at the authentication server (RADIUS).

Edge Creation — Edges can be created either on receipt of an 802.11 *reassociation request* frame by an AP or explicitly by APs themselves on reauthentication. The reassociation request frame contains the address of the old AP. Thus, the APs can maintain the edges in a distributed fashion, or explicitly inform the AAA server using a specific message. Also, if the APs implement the IEEE 802.1f Inter-AP Protocol, the receipt of a *Move-Notify* message (sent from an AP to the old AP during a reassociation) can induce an edge at the old AP.

Edge Deletion — Unused and stale edges (i.e., reassociation paths that rarely occur) can be deleted over time in a least recently used (LRU) fashion. This is necessary in order to delete incorrectly added edges. One situation where this could happen is a client that goes into power save mode, and potentially wakes up in a different location to reassociate to any arbitrary AP on the wireless network.

The Learning Curve — Each edge that is induced in the neighbor graph is learned at the cost of a slow authentication (full EAP-TLS). Thus, while the network learns the edges on the neighbor graph, there will be a penalty of one slow authentication per edge added, or $O(|E|)$ full EAP-TLS handoffs. This happens when a network is installed for first use, or there are topological changes (changes in AP positions that create/delete edges in the neighbor graph) in the network. Amortized over the total number of handoffs that can occur in the network, this penalty can be considered negligible.

Autonomous generation also eliminates the need for any survey or other manual construction methods. As a result, this also makes the data structure adaptive to changes in the reassociation relationship that might occur because of topology changes.

PROACTIVE KEY DISTRIBUTION

Proactive key distribution seeks to reduce the latency of the authentication phase by predistributing key material ahead of a mobile station. Our approach provides all of the properties of a full EAP-TLS authentication, but at significantly less cost in terms of latency and computational power of the mobile station.

PMK TREES

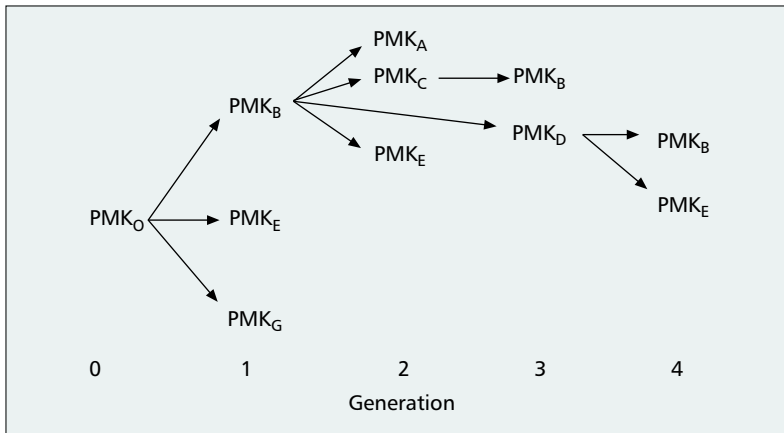
In the current, 802.11i framework the PMK is derived from the MK by Eq. 1. Predistributing this PMK, which is permitted in the current IEEE TGi draft as PMK caching, violates the TGi trust assumptions. Rather than predistribute this PMK, we change the derivation of the PMK to the recurrence shown in Eq. 2, where n represents the n th reassociation for $n \geq 0$.

$$PMK_0 = \text{TLS-PRF}(\text{MK}, \text{clientHello.random} \parallel \text{serverHello.random}) \quad (2)$$

$$PMK_n = \text{TLS-PRF}(\text{MK}, \text{PMK}_{n-1} \parallel \text{AP_MAC} \parallel \text{STA_MAC}).$$

The recurrence shown in the equation creates

Pro-active key distribution seeks to reduce the latency of the authentication phase by pre-distributing key material ahead of a mobile station. Our approach provides all of the same properties of a full EAP-TLS authentication, but at significantly less cost in terms of latency and computational power of the mobile station.



■ Figure 8. A PMK tree.

a PMK tree with reassociation pattern $\Gamma(STA)$, a path within the tree shown in Fig. 8.

In Fig. 8, the reassociation pattern is $\Gamma(STA) = A, B, C, D$.

PMK SYNCHRONIZATION

There are two conditions that can exist when a mobile station arrives at an AP with respect to predistribution of the correct PMK: either the AP and mobile station share the same PMK, or they do not. The handshake (two-way in our case and four-way in the case of TGi) determines which of these cases exist. This also ensures both *liveness* and *freshness* of the key.

PMK DISTRIBUTION

Once a mobile station completes an initial full EAP-TLS authentication, as denoted by the AAA server sending an ACCESS ACCEPT message to the AP, indicating successful completion of the authentication process as well as PMK_0 , the AAA server and mobile station share the MK, and the AAA server, the access point, and the mobile station all share PMK_0 . The AAA server now determines the neighbors of the associated AP and sends a NOTIFY-REQUEST that a specific mobile station may roam into the coverage area of each of the neighboring APs [8]. This message is advisory only, and an AP may or may not decide to request the security association or PMK from the AAA server at this time. If the AP does decide to request the PMK, the AP sends a NOTIFY-ACCEPT message. If not, the AP sends a NOTIFY-REJECT message to the AAA server. Upon receiving the NOTIFY-ACCEPT message, the AAA server responds with an ACCESS-ACCEPT message that contains the appropriate PMK as well as authorization for the mobile station to remain connected to the network. As a note, RADIUS messages have been designed around a challenge-response paradigm, and technically we violate it here by introducing the NOTIFY-REQUEST message. The Internet Engineering Task Force (IETF) is working on DIAMETER (an enhanced AAA server backward compatible to RADIUS) that includes the kinds of messages we discuss above, and our approach fits nicely into DIAMETER.

TWO-WAY HANDSHAKE

After the key distribution, the four-way handshake (discussed earlier) confirms the freshness of the keys being used by the AP and roaming STA. In our implementation we used a simpler two-way handshake (an EAPOL start message and an EAP Success message if the AP has the correct key) for purposes of demonstration. Since the four-way handshake is performed during both full authentication and fast re-authentication, it does not affect the key distribution scheme.

IMPLEMENTATION AND EXPERIMENTAL RESULTS

In this section, we present implementation results to demonstrate the performance of the proactive key distribution scheme. We have implemented the proactive key distribution method and standard full authentication over an in-building wireless testbed network comprising nine APs spread over three floors. Since the four-way handshake process appears in both schemes after the key has been delivered, we did not implement the four-way handshake and instead implemented a simple two-way handshake to verify the key freshness. We measured 90 full EAP-TLS authentication latencies with an average of approximately 1.1 s. Using the proactive key distribution scheme for fast re-authentication we obtained an average latency of 48 ms (a 99.6 percent reduction). *NOTE: We achieved an average of 20 ms in the laboratory, however. An investigation into the difference identified a hardware bug in the Power Over Ethernet circuitry of APs. We were only able to resolve the problem in one of the deployed APs prior to publication, and handoffs to this AP averaged 25 ms. Once the hardware in all of the testbed APs is updated, we fully expect an average latency of 25 ms for all of the APs.*

We also measured the overhead on the wired distribution system added by the two additional messages between the RADIUS server and the authenticator. With eight neighbors to distribute the key, the overhead was approximately 21 ms on average. Note that this overhead plays no role in the reauthentication latency, and just adds to the load on the RADIUS server and distribution system. We have included it for completeness.

THE IMPLEMENTATION

The wireless testbed network spans three floors (2nd, 3rd, and 4th) of a university building and consists of nine APs, as shown in Fig. 9. The AP is based on a NET4521 *Soekris* board, which has a 133 MHz AMD processor, 64 Mbytes SDRAM, two PC-Card/Cardbus slots for wireless adapters, and one *CompactFlash* socket. The board is powered using Power Over Ethernet through the Ethernet cable. A 200 mW Prism 2.5 based wireless card was used as the AP interface with a 1 ft yagi antenna. OpenBSD 3.3 with AP functionality was used as the operating system.

The supplicant and authenticator software was based on the *openIx* implementation built at

the University of Maryland, College Park (<http://www.open1x.org>). We also used the *Freeradius* software for the RADIUS server, modified to implement the key distribution scheme and maintain the neighbor graph data structure. The RADIUS server was installed on a backend machine (PIII 551.247 MHz, 128 Mbytes RAM). The *Xsupplicant* and *authenticator* software was modified to include the simple two-way handshake instead of the four-way handshake for purposes of demonstration.

EXPERIMENTAL RESULTS

The experimental setup consisted of a supplicant roaming in the wireless testbed. A laptop with PIII 1.8 GHz, 256 Mbytes RAM, and a Prism 2.5 based *DemarcTech* wireless card was used as the supplicant. Three experiments were done to measure three different latencies, as detailed below.

Measuring Full Authentication Latency: The supplicant was made to roam from one AP to another in the wireless network, and a full IEEE 802.1X EAP TLS authentication was performed at each reassociation. We measured 90 such authentications resulting in an average latency of 1.1 s.

Fast Re-Authentication: Fast reauthentication using proactive key distribution was enabled on the RADIUS and authenticators. The RADIUS server was initialized with the neighbor graph shown in Fig. 9. We used a static neighbor graph for ease of demonstration. The graph used in our experiments was constructed by human observation of the reassociation messages. Autonomous construction methods detailed earlier should be used in order to keep the neighbor graph fresh and dynamic; this has no effect on the performance of the key distribution scheme. Figure 10 shows the authentication latencies. The first authentication (which occurs at the start of a session) is a full one and hence incurs high latency (approximately 800 ms), while all subsequent 18 reauthentications reflect the latency of the two-way handshake.

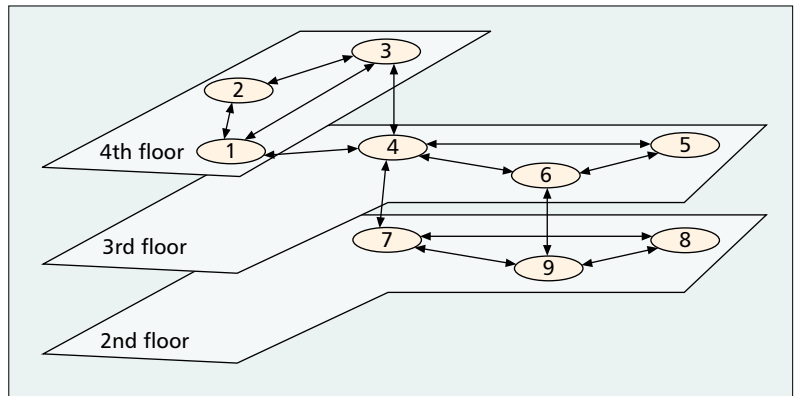
Overhead at the RADIUS Server: In this experiment we measured the additional overhead incurred by communication required for distributing the keys proactively using the Notify-Request, Notify-Accept, and Access-Accept messages. We measured 80 authentications and obtained an average overload of 21 ms. This overhead does not increase the handoff latency.

A REACTIVE METHOD FOR FAST SECURE REKEYING

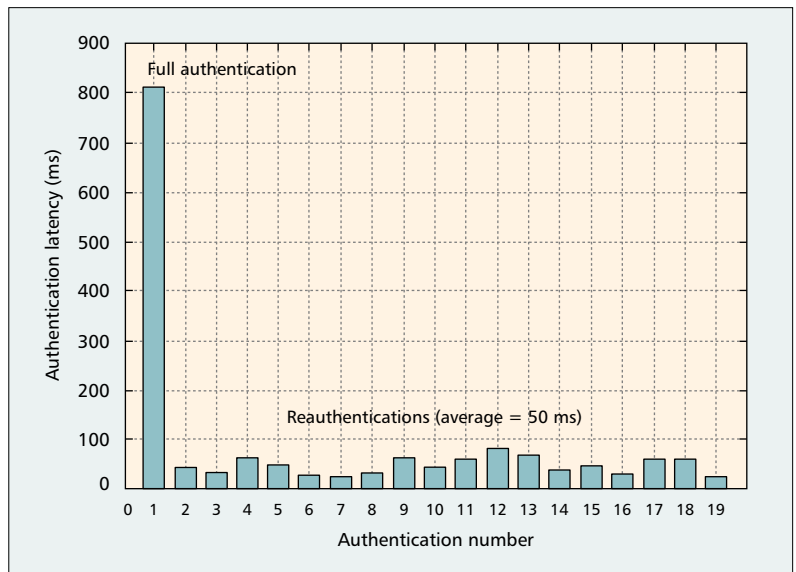
While the focus of this article has been on a novel approach to proactive keying, this section describes a reactive scheme that is both fast and secure. In addition to describing the protocol, results from laboratory tests conducted using the implementation are provided.

OVERVIEW

The protocol, which utilizes the EAP Identity message to notify the AAA server of a request for fast handoff, minimizes the number of messages between the AP and the AAA server, and



■ Figure 9. The topological placement of the APs in our wireless testbed and the resulting structure of the neighbor graph.

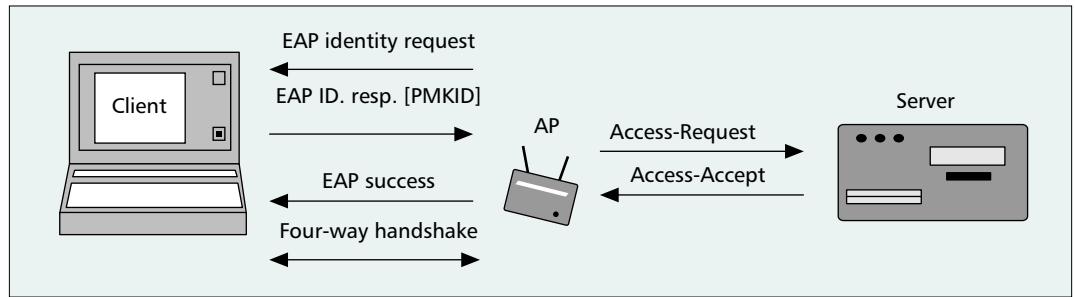


■ Figure 10. The authentication latencies as observed by the roaming supplicant in the wireless testbed, with proactive key distribution enabled. As can be seen, the first authentication reflects the full authentication latency and initiates the key distribution mechanism.

requires no changes to the existing EAP, EAPOL, or RADIUS protocols. In fact, to the casual observer the exchange looks exactly like a “canned success,” the term provided by IEEE 802.1X for granting network access immediately without performing authentication. However, while in the case of canned success all clients are provided access, the fast authentication scheme described here actually encapsulates its proof of identity inside the EAP Identity Response so that the AAA server can make an informed decision without using any EAP methods. The principle behind the scheme is simple: reduce the conversation to the minimal number of messages required by the existing infrastructure and provide the necessary protections to ensure that the result is no less secure than full authentication. The remainder of this section describes the details of our protocol, its advantages and disadvantages, and the results of our laboratory tests.

One of the fundamental design goals of our reactive approach was to avoid changes to the

Authentication servers are fewer in number than access points and typically have general purpose processors and operating systems. Client updates are simple as well, since the workload is divided among users who can trivially install software patches.



■ **Figure 11.** *The reactive fast handoff exchange.*

AP if at all possible. The reasons for this is simple. First, APs are likely the most difficult and costly equipment to replace and upgrade. While we believe the most efficient and effective approach to rekeying is a proactive scheme similar to that described earlier, the reality is that many organizations have invested large amounts of money in already deployed equipment. The ability to upgrade to a fast handoff scheme without buying new equipment or spending large amounts of time reflashing old equipment is a clear win. Second, the majority of 802.1X equipment on the market has been well tested for interoperability. Vendors would have far less overhead in rapidly deploying a fast handoff scheme if fewer protocol changes needed to be made. Unlike APs, supplicants and authentication servers are more easily upgraded. Authentication servers are fewer in number than APs and typically have general-purpose processors and operating systems. Client updates are simple as well, since the workload is divided among users who can trivially install software patches.

Based on the goals described above, we designed a reactive keying scheme that utilizes the EAP Identity Response to ask the AAA server for an expedited authentication response. The resulting protocol, detailed in Fig. 11, starts with the standard EAP Identity Request and Response sequence. As with the proactive scheme, the reactive scheme only works after a normal full authentication has begun the user's session. This full authentication uses TLS to produce a master secret (or uses another method to produce a shared MK) that is then used to bootstrap a set of keys for communication between the client and the AP. Once this MK and a corresponding PMK are in place, future PMKs can be calculated by both the client and authentication server using Eq. 2. In this scheme, a client using a valid PMK can request a fast handoff by including proof of knowledge of the PMK in the Identity Response, along with the user's identity. The authentication server then performs a trivial check for whether or not the Identity Response contains such proof, checks to make sure the proof is valid, and if so immediately responds with an Access Accept containing an EAP Success packet and the new PMK (encrypted for the AP). For this protocol we use the PMKID, which is defined by TGi and shown in Eq. 3, as proof of knowledge of the PMK. Note that once this Identity Response is sent, it can trivially be replayed by an observer. For this reason, the old PMK must immediately be invalidated once the

new PMK is sent from the authentication server to the new AP. Because the new PMK is based on the MK and the old PMK, only the authentication server and client can have knowledge of the new value before it is passed to the AP. Therefore, even the replay described above does not result in a violation of any of our security properties.

PMKID

$$= \text{HMAC_SHA1-128}(\text{PMK}, \text{"PMK Name"} || \text{AP-MAC-Address} || \text{Client-MAC-Address}). \quad (3)$$

IMPLEMENTATION AND EXPERIMENTAL RESULTS

As stated above, in order to verify the viability of the fast reactive method described here, we implemented the protocol and tested it using commercially available APs and freely available software. For the client, we modified *Xsupplicant* to calculate the current PMKID and include it in future Identity Responses when a fast handoff is desired. The only other change required was an implementation of Eq. 2 to calculate the new PMK. Similarly for the authentication server, we modified *FreeRadius* to cache the MK, PMK, and PMKID for each session, search for the valid PMKID when receiving a properly formed fast handoff Identity Response, and generate the new PMK and pass it to the AP in an Access Accept. We tested our implementation using two extremely common APs available on the market today. Because these APs were not WPA enabled but did support 802.1X, a four-way handshake could not be achieved. However, our tests included dynamic rekeying using the new PMK and standard 802.1X EAPOL Key messages.

The results produced by our experiments, which took place in a laboratory without interference or heavy load on any of the three machines, were extremely promising. As shown in Table 1, the time from receipt of an Identity Request until receipt of the second EAPOL Key packet was never greater than 35 ms for our reactive fast handoff scheme, while a full authentication never took less than 85 ms. Furthermore, the average times for our scheme using both APs was approximately 30 ms. These latencies are different from those obtained in the previous section primarily because the earlier experiments were carried out in a testbed network, while the experiments in this section were carried out on laboratory equipment, and the hardware equipment differed in speed and processing capability. Based on these results, we believe our fast reactive scheme is a feasible

	AP1	AP2
Full auth minimum	0.085	0.100
Full auth maximum	0.100	0.150
Full auth average	0.093	0.112
Fast auth minimum	0.025	0.025
Fast auth maximum	0.035	0.031
Fast auth average	0.030	0.030
Average fast/full	32.76%	26.28%

■ **Table 1.** Fast reactive approach — experimental results (in seconds).

solution that can be deployed immediately until better proactive schemes similar to those described in this article can be built into new equipment.

RELATED WORK

There has been prior work on reducing authentication latency by doing a predictive pre-authentication to a set of APs. Pack [9, 10] proposes a fast handoff scheme using a *predictive* authentication method based on IEEE 802.1X. In their scheme, pre-authentication is performed to the k most likely next APs. The k stations are selected using a weighted matrix representing the likelihood (based on the analysis of past network behavior) that a station, associated with AP_i , will move to AP_j . The mobile station may select only the most likely next APs to pre-authenticate, or select all of the potential next APs [9, 10]. Pack uses the notion of a frequent handoff region (FHR) to represent the adjacent APs, which is obtained by examining the weighted matrix. The weights within the matrix are based on an $O(n^2)$ analysis of RADIUS log information using the inverse of the ratio of the number of handoffs from AP_i to AP_j to the time spent by the mobile station at AP_i prior to the handoff. In [9] pre-authentication means the following. When a station authenticates to AP_i , an authentication server (AAA server) sends security information not only to AP_i but also to other APs in FHR. As a consequence, the next handoff to one AP in FHR does not require any message exchanges between the AP and the AAA server, because the AP already has the security information.

There are several issues with pre-authentication. First, pre-authentication cannot occur beyond the first access router due to the fact that EAPOL packets are used to carry authentication information. This severely limits the ability to pre-authenticate to single LANs only, and prohibits WAN and internetwork roaming. Second, the cost of a full reassociation is prohibitive for a capable device such as the laptop used in our experiments; imagine the times for a small handset using a low-powered processor. In addition, the authentication process must be accomplished to each potential neighbor. Thus, the cost is several seconds rather than milliseconds. During the authentication time, the mobile sta-

tion is on a different channel and unable to process traffic from or to the currently associated AP. Finally, unless there is a significant overlap in coverage, pre-authentication will just not work due to the length of times cited earlier.

The construction of an FHR matrix requires $O(n^2)$ computation and space, where n is the number of APs in the network, and must be created at the authentication server. Furthermore, the FHR notion does not quickly adapt to changes in the network topology. This is in contrast to our neighbor graphs, which require $O(\text{degree}(ap))$ computation and storage space per AP, and quickly adapt to changes in the network topology. Additionally, neighbor graphs can be utilized in either a distributed fashion at each AP or client, or a centralized fashion at the authentication server.

The previous work on context transfer mechanisms has primarily been reactive in nature. Reference [3] uses reactive context relocation at the IP layer, and [11] has general-purpose transfer mechanisms without detailing transfer triggers. The IP layer context transfer mechanisms focus solely on the transfer of context from access router to access router, and Koodli [3] mentions APs briefly, indicating that access routers and APs can be collocated. The context transfer mechanisms are designed solely for access routers and are reactive rather than proactive as in neighbor graphs [3]. The SEAMOBY context transfer protocol provides a generic framework for either reactive or proactive context transfers [11]. The framework, however, does not define methods for implementing either reactive or proactive context transfers. As a result, our approach can easily be integrated into the SEAMOBY protocol, providing a proactive key distribution mechanism.

There has been prior work on topology learning algorithms. In the 1980s to overcome the geographic limitations of a LAN, they were connected using *bridges*. In this approach, a bridge connecting two or more links listens promiscuously to all packets and forwards them to a link on which the destination station is known to reside. A bridge also dynamically learns the locations of stations so that it can forward traffic to the correct link. In [12] Perlman proposed a *self-configuring* and *distributed* algorithm to allow bridges to learn the loop-free subset of the topology that connects all LANs, by communicating with other bridges. This subset is required to be loop-free (i.e., a spanning tree) to avoid unnecessary congestion caused by infinitely circulating packets. This Spanning Tree Algorithm/Protocol [13] is self-configuring because the only a priori information necessary in a bridge is its own unique ID (MAC address). The algorithm requires a very small bounded amount of memory per bridge, and a bounded amount of communications bandwidth for each LAN. Furthermore, there are no modifications to stations, and the algorithm interoperates with older bridges. Neighbor graphs are also self-configuring and operate in the same manner: examining network traffic, specifically layer 2 management frames or AAA messages, to create the wireless network topology dynamically. The two algorithms and their purposes are different, however.

Based on the results presented here, we believe our fast reactive scheme is a feasible solution that can be deployed immediately until better proactive schemes similar to those described in this article can be built into new equipment.

We are working on other interesting applications using the neighbor graph data structure. Neighbor graphs can also be used to eliminate the expensive scanning operation for faster MAC layer handoffs by making an intelligent guess about the list of APs on a particular channel.

CONCLUSIONS

Wireless networking has changed considerably over the last decade, and the next decade will likely see the ubiquity of wireless network service. Accomplishing this goal will require the interworking of different administrative domains and different physical layers. If WiFi networks are to be participants in this vision, the current handoff latencies must be reduced significantly.

In this article we consider the problem of horizontal handoffs in wireless LANs within the same administrative domain. We present a novel data structure, neighbor graphs, that dynamically capture the mobility topology of a wireless network. We discuss a proactive key distribution method using neighbor graphs, which distributes key material one hop ahead of a mobile user. While the proactive method is a good long-term solution, we also present a fast reactive method that works with off-the-shelf and currently deployed 802.1X compatible APs. We demonstrate that both approaches provide the same level of security as a full EAP-TLS authentication, but at significantly lower latency (99 percent reduction).

We are working on other interesting applications using the neighbor graph data structure. Neighbor graphs can also be used to eliminate the expensive scanning operation for faster MAC layer handoffs by making an intelligent guess about the list of APs on a particular channel. Neighbor graphs could also potentially lead to a scalable method of organizing and managing a large-scale cooperative wireless network that interconnects APs from different network domains and with different characteristics.

ACKNOWLEDGMENT

The authors would like to thank Pasi Eronen of the Nokia Research Center for initial contributions and feedback on our fast reactive protocol. The authors wish to acknowledge efforts from Mike van Opstal of our laboratory for helping with the experiments on our testbed, and finally Kyunghun Jang and Insun Lee of Samsung Electronics for their contributions and feedback. This work was funded in part by a grant from Samsung Electronics, 0307158417, and by U.S. National Institute of Standards and Technology Critical Infrastructure Grant 60NANB1D0113.

REFERENCES

- [1] M. Balazinska and P. Castro, "Characterizing Mobility and Network Usage in a Corporate Wireless Local-Area Network," *Int'l. Conf. Mobile Systems, Apps., and Services*, May 2003.
- [2] A. Mishra, M. Shin, and W. Arbaugh, "An Empirical Analysis of the IEEE 802.11 Mac Layer Handoff Process," *ACM SIGCOMM Comp. Commun. Rev.*, vol. 33, Apr. 2003.
- [3] R. Koodli and C. Perkins, "Fast Handover and Context Relocation in Mobile Networks," *ACM SIGCOMM Comp. Commun. Rev.*, vol. 31, Oct. 2001.
- [4] IEEE Std. P802.1X, "Standards for Local and Metropolitan Area Networks: Standard for Port Based Network Access Control," Oct. 2001.

- [5] J. Edney and W. A. Arbaugh, *Real 802.11 Security*, Addison Wesley, 2003.
- [6] IEEE Std. 802.11i, "Draft Amendment to Standard for Telecommunications and Information Exchange between Systems-lan/man Specific Requirements, Part 11: Wireless Medium Access Control and Physical Layer (phy) Specifications: Medium Access Control (MAC) Security Enhancements," May 2003.
- [7] A. Mishra, M. Shin, and W. Arbaugh, "Context Caching Using Neighbor Graphs for Fast Handoffs in a Wireless Network," to appear, *Proc. IEEE INFOCOM 2004*.
- [8] W. A. Arbaugh and B. Aboba, "Experimental Handoff Extension to RADIUS," Internet draft, May 2003.
- [9] S. Pack and Y. Choi, "Fast Inter-AP Handoff Using Predictive-Authentication Scheme in a Public Wireless LAN," *IEEE Networks*, Aug. 2002.
- [10] S. Pack and Y. Choi, "Pre-Authenticated Fast Handoff in a Public Wireless LAN based on IEEE 802.1x Model," *IFIP TC6 Pers. Wireless Commun.*, Oct. 2002.
- [11] M. Nakhjiri, C. Perkins, and R. Koodli, "Context Transfer Protocol," Internet Draft: draft-ietf-seamoby-ctp-01.txt, Mar. 2003.
- [12] R. Perlman, "An Algorithm for Distributed Computation of a Spanning Tree in an Extended LAN," 1985, pp. 44-53.
- [13] R. Perlman, *Interconnections, 2nd Edition: Bridges, Routers, Switches and Internetworking Protocols*, Pearson Education, Sept. 1999.

BIOGRAPHIES

ARUNESH MISHRA (arunesh@cs.umd.edu) is a fourth-year graduate student in the Department of Computer Science at the University of Maryland, College Park. His research areas include wireless networks and systems security. He received a B.Tech. in computer science from the Indian Institute of Technology, Guwahati, and an M.S. in computer science from the University of Maryland, College Park.

MIN HO SHIN (mhshin@cs.umd.edu) received his B.S. degree in computer science and statistics from Seoul National University in 1998. He also received his M.S. degree in computer science from the University of Maryland in 2003. Currently he is a graduate research assistant with Maryland Information System Security Laboratory (MISSL) and a Ph.D. student at the University of Maryland, College Park. His current research interests include wireless network security and 3G/WLAN integration security.

NICK L. PETRONI, JR. [M] (npetroni@cs.umd.edu) is a third-year graduate student in the Department of Computer Science at the University of Maryland, College Park. His research interests include information security, trusted computing, and wireless networks. He received a B.S. in computer science from the University of Notre Dame and an M.S. in computer science from the University of Maryland, College Park. He is a member of the ACM.

T. CHARLES CLANCY (clancy@cs.umd.edu) is a Ph.D. candidate in computer science at the University of Maryland. He earned his M.S. in electrical engineering at the University of Illinois in Urbana-Champaign, and his B.S. in computer engineering at the Rose-Hulman Institute of Technology. His research interests include network protocols and cryptography.

WILLIAM A. ARBAUGH (waa@cs.umd.edu) is an assistant professor in the Department of Computer Science at the University of Maryland, College Park. His research interests include information systems security and privacy with a focus on wireless networking, embedded systems, and configuration management. He received a B.S. from the United States Military Academy at West Point, an M.S. in computer science from Columbia University, New York, and a Ph.D. in computer science from the University of Pennsylvania, Philadelphia. He is on the editorial boards of *IEEE Computer* and *IEEE Security and Privacy*.