Secure In-Band Wireless Pairing

Shyamnath Gollakota, Nabeel Ahmed, Nickolai Zeldovich, and Dina Katabi
Massachusetts Institute of Technology

ABSTRACT

This paper presents the first wireless pairing protocol that works in-band, with no pre-shared keys, and protects against MITM attacks. The main innovation is a new key exchange message constructed in a manner that ensures an adversary can neither hide the fact that a message was transmitted, nor alter its payload without being detected. Thus, any attempt by an adversary to interfere with the key exchange translates into the pairing devices detecting either invalid pairing messages or an unacceptable increase in the number of such messages. We analytically prove that our design is secure against MITM attacks, and show that our protocol is practical by implementing a prototype using off-the-shelf 802.11 cards. An evaluation of our protocol on two busy wireless networks (MIT's campus network and a reproduction of the SIGCOMM 2010 network using traces) shows that it can effectively implement key exchange in a real-world environment.

1 INTRODUCTION

Recent trends in the security of home WiFi networks are driven by two phenomena: ordinary users often struggle with the security setup of their home networks [14], and, as a result, some of them end up skipping security activation [19, 26]. Simultaneously, there is a proliferation of WiFi gadgets and sensors that do not support an interface for entering a key. These include WiFi sound systems, medical sensors, USB keys, light and temperature sensors, motion detectors and surveillance sensors, home appliances, and game consoles. Even new models of these devices are unlikely to support a keypad because of limitations on their form factor, style, cost, or functionality. Responding to these two requirements—easing security setup for home users, and securing devices that do not have an interface for entering a key—the WiFi Alliance has introduced the Push Button Configuration (PBC) mechanism [26]. To establish a secure connection between two WiFi devices, the user pushes a button on each device, and the devices broadcast their Diffie-Hellman public keys [7], which they then use to protect all future communication. PBC is a *mandatory* part of the new WiFi Protected Setup certification program [27]. It is already adopted by the major WiFi manufacturers (e.g., Cisco, NetGear, HP, Microsoft, Sony) and implemented in about 2,000 new products from 117 different companies [25].

Unfortunately, the PBC approach taken by the WiFi Alliance does not fully address WiFi security. Diffie-Hellman's key-exchange protocol [7] protects against only passive adversaries that snoop on the wireless medium to obtain key exchange messages. Since the key exchange messages are not authenticated in any way, the protocol is vulnerable to an active man-in-the-middle (MITM) attack. That is, an adversary can impersonate each device to the other, convincing both devices to establish a secure connection via the adversary. With WiFi increasingly used in medical sensors that transmit a patient's vital signals [11] and surveillance sensors that protect one's home [16, 21], there is a concern that, being vulnerable to MITM attacks, PBC may give users a false sense of security [15, 26].

One may wonder why the WiFi Alliance did not adopt a user-friendly solution that also protects against MITM attacks. We believe the reason is that existing user-friendly solutions to MITM attacks require devices to support an out-of-band communication channel [6, 10, 17, 18, 20, 22]. For example, devices can exchange keys over a visual channel between an LCD and a camera [18], an audio channel [10], an infrared channel [2], a dedicated wireless channel allocated exclusively for key exchange [6], etc. Given the cost, size, and capability constraints imposed on many WiFi products, it is difficult for the industry to adopt a solution that requires an out-of-band communication channel.

This paper presents *tamper-evident pairing* (TEP), a novel protocol that provides simple, secure WiFi pairing and protects against MITM attacks without an out-of-band channel. TEP can also be incorporated into PBC devices and existing WiFi chipsets without hardware changes.

TEP's main challenge in avoiding MITM attacks comes from operating on a shared wireless network, where an adversary can mask an attack behind cross traffic, making it difficult to distinguish an adversary's actions from legitimate traffic patterns. To understand this, consider a key exchange between Alice and Bob, where Bob sends his Diffie-Hellman public key to Alice. Lucifer, the adversary, could tamper with this key exchange as follows:

- *Collision:* Lucifer can jam Bob's message, causing a collision, which would not look out-of-the-ordinary on a busy wireless network. The collision prevents Alice from decoding Bob's message. Lucifer can now send his own message to Alice, in lieu of Bob's message, perhaps with the help of a directional antenna so that Bob does not notice the attack.

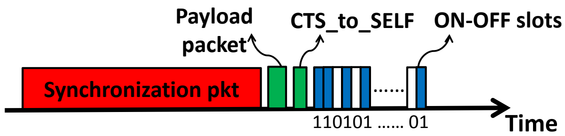

Figure 1: **The format of a tamper-evident announcement (TEA).**

- *Capture effect:* Lucifer can transmit simultaneously with Bob, but at a significantly higher power, to produce a capture effect at Alice [24]. In this case, Alice will decode Lucifer's message, in which he impersonates Bob, despite Bob's concurrent transmission. Bob will not know about Lucifer's transmission.
- *Timing control:* Lucifer can try to impersonate Alice by continuously occupying the wireless medium after Bob sends out his key, so that Lucifer can send out a message pretending to be Alice, but Alice does not get a chance to send her legitimate key.

To address these attacks in TEP, we introduce a *tamper-evident announcement* (TEA) primitive. The key characteristics of a TEA message is that an attacker can neither hide a TEA transmission from other nodes within radio range, nor can it modify the content of the TEA without being detected. Thus, a TEA provides stronger guarantees than payload integrity because it also protects the fact that a message was transmitted in the first place.

Fig. 1 shows the structure of a TEA. First, to ensure that Lucifer cannot mask Bob's TEA message by introducing a collision, the TEA starts with an exceptionally long packet. Since standard WiFi collisions are significantly shorter, Alice needs to detect only exceptionally long collisions (i.e., exceptionally long bursts of energy) as potential attacks on the key exchange process.

Second, to ensure that Lucifer cannot alter the payload of Bob's TEA by transmitting his own message at a high power to create a capture effect, we force any TEA message to include silence periods. As shown in Fig. 1, the payload of the TEA message is followed by a sequence of short equal-size packets, called *slots*, where the transmission of a packet is interpreted as a "1" bit, and an idle medium is interpreted as a "0" bit. The bit sequence produced by the slots must match a hash of the TEA payload. If Lucifer overwrites Bob's message with his own, he must transmit slots corresponding to a hash of his message, including staying silent during any zero hash bits. However, since the hash of Lucifer's message differs from that of Bob's message, Bob's message will show up on the medium during Lucifer's "0" slots. Alice will detect a mismatch between the slots and the message hash and reject Lucifer's message.

Third, to ensure that legitimate nodes do not mess up the timing of Alice and Bob's key exchange, the TEA message includes a CTS-to-SELF, as shown in Fig. 1. CTS-to-SELF is an 802.11 message that requires honest nodes to refrain from transmitting for a time period specified in the packet. TEP leverages this message for two goals. First, it uses it to reserve the medium for the duration of the TEA slots to ensure that legacy 802.11 nodes, unaware of the structure of a TEA message, do not sense the medium as idle and transmit during a TEA's silent slots. Second, TEP also uses CTS-to-SELF to reserve the medium for a short period after the TEA slots, to enable Alice to send her key to Bob within the interval allowed by PBC. Once Alice starts her transmission, the medium will be occupied, and honest 802.11 nodes will abstain from transmitting concurrently. If Lucifer transmits during the reserved time frame, Alice will still transmit her TEA message, and cause a collision, and hence an invalid TEA message that Bob can detect.

We build on TEA to develop the TEP pairing protocol. TEP exploits the fact that any attempts to alter or hide a TEA can be detected. Thus, given a pairing window, any attempt by an adversary to interfere with the pairing exchange translates into either an increase in the number of TEA messages or some invalid TEA messages. This allows the pairing devices to detect the attack and indicate to the user that pairing has failed and that she should retry. The cost of such a mechanism is that the user has to wait for a pre-determined duration of the pairing window. In §5.4, we describe how one may eliminate this wait by having a user push the button on a device a second time.

This paper formalizes the above ideas to address possible interactions between the pairing devices, adversaries, and other users of the medium, and formally proves that the resulting protocol is secure against MITM attacks. Further, we build a prototype of TEP as an extension to the Ath5k driver [1], and evaluate it using off-the-shelf 802.11 Atheros chipsets. Our findings are as follows:

- TEP can be accurately realized using existing OS and 802.11 hardware. Specifically, our prototype sender can schedule silent and occupied slots at a resolution of 40μs, and its 95th percentile scheduling error is as low as 1.65μs. Our prototype receiver can sense the medium's occupancy over periods as small as 20μs and can distinguish occupied slots ("1" bits) from silent slots ("0" bits) with a zero error rate.
- Results from running the protocol on our campus network and applying the traces from the network during the SIGCOMM 2010 conference, show that TEP never confuses honest 802.11 traffic for an attack. Furthermore, though our implementation is for 802.11, it can coexist with nearby Bluetooth devices which do not respect TEP silent slots. In this case, TEP can still perform a key exchange using 1.4 attempts, on average.

Contributions: This paper presents, to our knowledge, the first wireless pairing protocol that defeats MITM attacks without any key distribution or out-of-band channels.

It does so by introducing TEA, a new key exchange message constructed in a manner that ensures an adversary can neither hide the fact that a message was transmitted, nor alter its payload without being detected. Our protocol is prototyped using off-the-shelf 802.11 devices and evaluated in production WiFi networks.

2 Related Work

There has been a lot of interest in user-friendly secure wireless pairing, which has led to a number of innovative solutions [2, 6, 10, 17, 18, 20, 22]. TEP builds on this foundational work. However, TEP is the first to provide a secure pairing scheme that defeats MITM attacks without out-of-band channels, or key distribution or verification.

Closest to TEP is the work on integrity codes [5], which protects the integrity of a message's payload by inserting a particular pattern of ON-OFF slots. Integrity codes, however, assume a dedicated out-of-band wireless channel. In contrast, on shared channels, honest nodes may disturb the ON-OFF pattern by acquiring the medium during the OFF slots. Further, the attacker can hide the fact that a message was transmitted altogether, by using collisions or a capture effect. We build on integrity codes, but introduce TEA, a new communication primitive that not only protects payload integrity but also ensures that an attacker cannot hide that a message was transmitted. We further construct TEP by integrating TEA with the 802.11 standard, the PBC protocol, and the existing OS network stack. Finally, we implement TEP on off-the-shelf WiFi devices and evaluate it in operational networks.

TEP is also related to work on secure pairing, which traditionally required the user to either enter passwords or PINs [3, 4, 12], or distribute public keys (e.g., STS [8], Radius in 802.11i [13], or any other public key infrastructure). These solutions are appropriate for enterprise networks and for a certain class of home users who are comfortable with security setup. However, the need to ease security setup for non-technical home users has motivated multiple researchers to propose alternative solutions for secure pairing. Most previous solutions use a trusted out-of-band communication channel for key exchange. The simplest channel is a physical wired connection between the two devices. Other variants of out-of-band channels include the use of a display and a camera [18], an audio-based channel [10], an infra-red channel [2], a tactile channel [22], or an accelerometer-based channel [17]. While these proposals protect against MITM attacks, many devices cannot incorporate such channels due to size, power, or cost limitations. In contrast, TEP eases the security setup for home users and defeats MITM attacks, without any out-of-band channel.

Finally, multiple user studies [14, 19, 26] have emphasized the difficulty in pairing devices for ordinary users. Our work is motivated by these studies. TEP requires the user to just push a button on each device—exactly as in PBC—and does not require any additional user involvement in key generation or verification.

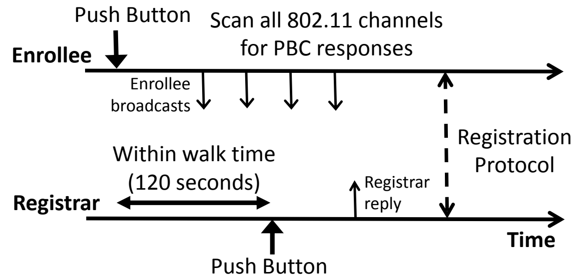

Figure 2: **A timeline depicting the operation of Push Button Configuration (PBC) between an enrollee and a registrar.**

3 PBC and 802.11 Background

3.1 Push Button Configuration

The WiFi-Alliance introduced the Push Button Configuration (PBC) mechanism to ease the security setup process for ordinary users, and to deal with devices that do not have an interface to enter passwords or PINs. In this section, we provide an overview of how PBC works.

Consider a home user who wants to associate an *enrollee* (PBC's term for the new device, e.g., a gaming console) with a *registrar* (PBC's term for, effectively, the access point). The user first pushes a button on the enrollee and then, within 120 seconds (called the walk time), pushes the button on the registrar. Once the buttons are pushed on the two devices, the devices perform a Diffie-Hellman key exchange to establish a secret key.

As shown in Fig. 2, once the button is pushed on the enrollee, it periodically sends probes [26] requesting replies from registrars whose PBC button has been pressed. Once the enrollee receives a reply, it makes a note of the reply and continues to scan all the 802.11 channels for additional replies. If the enrollee receives replies from more than one registrar, across all 802.11 channels, it raises a session overlap error, indicating that the user should try again later. On the other hand, if it receives a reply from only one registrar, it proceeds with the registration protocol, using the Diffie-Hellman key from that one reply.

A registrar, for its part, stays on its dedicated channel, and replies to probe requests only if the user has pushed its PBC button. Once the button is pushed, the registrar replies to PBC requests from potential enrollees. To detect conflicts, the registrar checks for requests in the last 120 seconds. If there are requests from more than one enrollee, the registrar signals a session overlap error and refuses to perform the PBC registration protocol, requiring the user to retry. If there was only one enrollee request, the registrar proceeds with the registration protocol using the Diffie-Hellman public key from that one request.

While PBC's use of Diffie-Hellman protects the devices from eavesdropper attacks, an active adversary can hide or change any of the messages, by resorting to collisions, capture effect attacks, or hogging the medium and delaying these messages. This allows an adversary to gain access to the user's registrar (e.g., their home network), the enrollee device, or to intercept and alter any future messages between the enrollee and registrar. Defending against such adversaries requires a system that is robust to MITM attacks, which is the main contribution of TEP.

3.2 802.11

Since our protocol involves low-level details of the 802.11 standard, we summarize the relevant aspects of 802.11 in this section. 802.11 requires nodes to sense the wireless medium for energy, and transmit only in its absence. 802.11 nodes can transmit using a range of bit rates, with the minimum bit rate of 1 Mbps. Coupling this with the fact that the maximum packet size used by higher layers is typically 1500 bytes, an honest node can occupy the channel for a maximum of 12 ms. 802.11 requires back-to-back packets to be separated by an interval called the DCF Inter-Frame Spacing (DIFS), whose value can be $34\mu s$, $50\mu s$, or $28\mu s$, depending on whether the network uses 802.11a, b, or g. 802.11 acknowledgment packets, however, can be transmitted after a shorter duration of $10\mu s$, called the Short Inter-Frame Spacing (SIFS).

4 Security Model

TEP addresses the problem of authenticating key exchange messages between two wireless devices, in the presence of an active adversary that may try to mount a man-in-the-middle attack.

4.1 Threat Model

The adversary can eavesdrop on all the signals on the channel, including all prior communications. The adversary can also be active and transmit with an arbitrary power, at any time, thereby corrupting or overpowering other concurrent transmissions. The adversary may know the TEP protocol, the precise times when devices transmit their announcements, and their exact locations. In addition, the adversary can know the exact channel between the pairing devices, and the channel from the pairing devices to the adversary. The adversary can also be anywhere in the network and is free to move. Multiple adversaries may exist in the network and can collude with each other.

The adversary can have access to state-of-the-art RF technologies: he can have a multi-antenna system, he may be able to simultaneously receive and transmit signals, and he can use directional antennas to ensure that only one of the pairing devices can hear its transmissions.

The adversary, however, does *not* have physical control over the pairing devices or their surroundings. Specifically, the adversary cannot place either of the two devices in a Faraday cage to shield all signals. We also assume that the adversary cannot break traditional cryptographic constructs, such as collision-resistant hash functions.

Term	Definition
Tamper-evident announcement	A wireless message whose presence and the integrity of its payload are guaranteed to be detected by every receiver within radio range (Figure 1).
Synchronization packet	An exceptionally long packet whose presence indicates a TEA. To detect a synchronization packet, it is sufficient to detect that the medium is continuously occupied for the duration of the synchronization packet, which is 19 ms.
Payload packet	The part of a TEA containing the data payload (e.g., a device public key).
ON-OFF slot	The interval used to convey one bit from sender to receiver. The slot time is 40μs. The bits in the slots are balanced, as described in §5.1.2.
Occupied/ON slot	A slot during which the medium is busy with a transmission.
Silent/OFF slot	A slot during which the medium is idle.
Sensing window	The interval over which the receiver collects aggregate information for whether the medium is occupied or silent.
Fractional occupancy	The fraction of time the medium was busy during a sensing window.

Table 1: Terminology used to describe TEP.

Finally, we assume that the PBC buttons operate according to the PBC standard [26] and that the user performs the PBC pairing as prescribed in the standard, i.e., the user puts the two devices in range then pushes the buttons on the two devices within 120 seconds of each other.

4.2 Security Guarantees

Under the assumptions outlined above, TEA guarantees that an adversary cannot tamper with the payload of a TEA message, or mask the fact that a TEA message was transmitted. Building on the TEA mechanism, TEP guarantees that in the absence of an active adversary, two pairing devices can establish secure pairing. In the presence of an adversary who is actively mounting MITM attacks (or in the presence of more than two devices attempting to pair at the same time), TEP ensures that the pairing devices will signal an error and never be tricked into pairing with the adversary (or, more generally, with the wrong device). In other words, TEP provides the PBC security guarantees augmented with protection against MITM attacks.

5 TEP Design

TEP's design is based upon the TEA mechanism, a unidirectional announcement protocol that guarantees that adversaries cannot tamper with or mask TEA messages without detection. TEP uses TEA to exchange public keys between the PBC enrollee and registrar in a way that resists MITM attacks. At a high level, when an enrollee enters PBC mode, it sends out a TEA message containing its public key. When a registrar in PBC mode receives this message (or suspects that an adversary may have tried to tamper with or mask such a message), it responds with its own public key. Both the enrollee and the registrar collect all TEA messages received during PBC's walk time period. If, during that time, each received exactly one unique public key (and no tampered messages), they can conclude that this public key came from the other party, and can use it for pairing. Otherwise, PBC reports a session overlap error (e.g., because multiple enrollees or registrars were pairing at the same time, or because an adversary interfered), and asks the user to retry.

The rest of this section describes our protocol in more detail, starting with the TEA mechanism, using terminology defined in Table 1.

5.1 Tamper-Evident Announcement (TEA)

The goal of TEA is to guarantee that if an attacker tampers with the payload of a TEA message, or tries to mask the fact that a message was transmitted at all, a TEA receiver within communication range will detect such tampering. In other words, TEA receivers will *always* detect when a TEA message was, or may have been, transmitted.

To provide this guarantee, TEA messages have a specialized structure, as shown in Figure 1. First, there is a synchronization packet, which protects the TEA's transmission from being masked, by unambiguously indicating to a TEA receiver that a TEA message follows. The synchronization packet contains random data, to ensure that an adversary cannot cancel out its energy.[1]

Second, the TEA message contains the announcement payload. The payload is always of fixed length, to ensure that an adversary cannot truncate or extend the payload in flight, but otherwise has no restrictions on its content or encoding. In our pairing protocol, the payload of a TEA message contains the sender's Diffie-Hellman public key, along with other registration information.

Third, the TEA message contains ON-OFF slots, which guarantee that any tampering with a TEA payload is detectable. Similar to the synchronization packet, the content of the ON slots is randomized. The first two slots, as shown in Fig. 3, encode the direction flag, which defines whether this TEA message was sent by an enrollee (called a TEA request, flag value "10") or by a registrar (called a TEA reply, flag value "01"). The remaining slots contain a cryptographic hash of the payload. While it is possible to also encode the payload using slots, it would be inefficient for long payloads, and unnecessary, since protecting a cryptographic hash suffices. To detect tampering, TEA encodes all slots in a way that guarantees that exactly half of the slots are silent, as we describe in §5.1.2.

[1] In practice, it is very hard to cancel a signal in flight but in theory an attacker that knows the transmitted signal and the channels to the receiver can construct a signal that cancels out the original signal at the receiver. Making the data random eliminates this option.

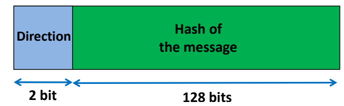

Figure 3: Data encoded in the ON-OFF slots. The first two bits specify the direction of the message, and the rest of the bits contain a cryptographic hash of the payload.

5.1.1 Detecting tampering

To determine if an adversary may have tampered with a TEA message, a TEA receiver performs several checks. First, the receiver continuously monitors the medium for possible synchronization packets. If it detects any burst of energy at least as long as the synchronization packet, it interprets it as the start of a TEA announcement. The receiver conservatively assumes that any such period of energy is a TEA message, and signals a missed message if it is unable to decode and verify the subsequent payload. To minimize false positives, we choose a synchronization packet that is longer than any regular contiguous WiFi transmission. An adversary cannot cancel out a legitimate synchronization packet because the adversary cannot eliminate the power on the channel. In fact, since the payload of the synchronization packet is random, the adversary cannot cancel the power from the packet even if he knows the exact channel between Alice and Bob, and is fully synchronized with the transmitter. Thus, an adversary cannot tamper with the presence of a TEA message by masking it out.

Second, once a TEA receiver detects the start of a TEA announcement, it attempts to decode the payload packet and the hash bits in the ON-OFF slots. If the receiver cannot decode the payload (i.e., the packet checksum fails), it indicates tampering. If the payload is decoded, the receiver verifies that the hash bits match the hash of the payload– i.e., it verifies that hashing the payload produces the same bits in the ON-OFF slots and that the number of ON slots is equal to that of OFF slots. If the receiver cannot verify the hash bits, it conservatively assumes that an adversary is tampering with the transmission. Once tampering is detected, the receiver signals a session overlap error (as in PBC), requiring the user to retry later.

5.1.2 Balancing the ON-OFF Slots

An adversary can transform an OFF slot to an ON slot (by transmitting in it) but cannot transform an ON slot to an OFF slot. Hence, to ensure that the adversary cannot tamper with even a single OFF slot without being detected, we make the number of the OFF slots in a TEA message equal to that of the ON slots, i.e., we balance the slots. The number of slots is fixed by the TEP protocol, thus avoiding truncation or extension attacks. Since the direction flag is already encoded in two balanced bits, we now focus on balancing the rest of the slots.

Our balancing algorithm takes the hash bits of the TEA payload and produces a balanced bit sequence to be sent in the ON-OFF slots. One inefficient but simple transformation is to use Manchester encoding of the hash bits to produce a balanced output bit sequence with twice as many output bits. TEA, however, introduces an efficient encoding that takes an even number, N, of input bits and produces $M = N + 2\lceil logN \rceil$ output bits which have an equal number of zeros and ones. The details of our efficient encoding algorithm are presented in Appendix A.

5.1.3 Interoperating with 802.11

To interoperate with other 802.11 devices that may not be TEA-aware, the ON-OFF slots are preceded by a CTS-to-SELF packet, which reserves the medium for the TEA message. This serves two purposes. First, since the sender does not transmit during the OFF slots, another 802.11 node could sense the wireless medium to be idle for more than a DIFS period, and start transmitting its own packet during that OFF slot. The 802.11 standard requires 802.11 nodes that hear a CTS-to-SELF on the channel to abstain from transmitting for the period mentioned in that packet, which will ensure that no legitimate transmission overlaps with the slots. Second, in case of a TEA message from an enrollee to a potential registrar, the CTS-to-SELF packet reserves the medium so that the registrar can immediately reply with its own TEA message. This prevents legitimates nodes from hogging the medium and delaying the registrar's response. However, reserving the channel for the entire length of a TEA message is inefficient, if no registrar is present. To avoid under-utilization of the wireless medium, the enrollee's CTS-to-SELF only reserves the channel for a DIFS period past its slot transmissions. If a PBC-activated registrar is present, it *must* start transmitting its response message within the DIFS period. On the other hand, if there is no registrar, other legitimate devices will resume transmissions promptly.

To maximize the probability that all devices can decode the CTS-to-SELF, it is transmitted at the most robust bit rate of 1 Mbps. Current 802.11 implementations obey a CTS-to-SELF that reserves the channel up to 32 ms. Our TEA message requires 144 slots,[2] and the slot duration is 40 μs (§6). This translates to about 5.8 ms, which is less than the 32 ms allowed by the CTS-to-SELF.

Finally, as shown in Figure 1, there is a gap between the synchronization and payload packets. If this gap is large, other 802.11 nodes would sense an idle wireless medium, and start transmitting, thus appearing to tamper with the TEA. To avoid this, we exploit the fact that 802.11 nodes are only allowed to transmit if they find the medium continuously idle for a DIFS. Thus, a TEA sender sends the payload packet immediately after the synchronization packet with a gap of a Short Interframe Space (SIFS), which is much less than DIFS.

5.1.4 API Summary

The interface provided by TEA is as follows. For the sender side, there is a single blocking function,

- void TEA_SEND (bool *dir*, str *msg*, time *t*),

which sends an announcement containing payload *msg*. The *dir* flag specifies the direction of the message, that is, whether it is a request message (from the enrollee) or a reply message (from the registrar). Time *t* specifies the deadline by which the message must start transmission. The TEA sender tries to respect carrier-sense in the medium access control (MAC) protocol, and waits until the medium is idle before transmitting its message. However, if the message cannot be transmitted by time *t* (e.g., because an adversary is hogging the medium), the sender overrides the MAC's carrier-sense, and transmits the announcement anyway, so that recipients will detect tampering. Note that the CTS-to-SELF requires honest nodes to release the medium for the registrar to transmit its own TEA reply.

For the receiver side, TEA provides two functions,

- handle TEA_RECV_START (bool *dir*), and
- msg_list TEA_RECV_GET (handle *h*).

The first function, TEA_RECV_START, starts listening on the wireless medium for TEA messages that are either requests (from an enrollee) or replies (from a registrar), based on the *dir* flag. The second function, TEA_RECV_GET, is used to retrieve the set of messages accumulated by the receiver since TEA_RECV_START or TEA_RECV_GET was last invoked. If TEA_RECV_GET could not decode a possible TEA message (or verify that it was not tampered with), it returns a special value RETRY, which causes the caller (i.e., TEP) to re-run its protocol. As an optimization, if *all* of the TEA messages that TEA_RECV_GET was unable to decode were overlapping with the receiver's own transmissions (i.e., a concurrent TEA_SEND), TEA_RECV_GET returns a special value OVERLAP instead of RETRY. We describe in §6.4 how a node detects TEA messages that overlap with its own transmissions, and in Appendix B how we use the overlap information to optimize wireless medium utilization.

5.2 Securing PBC using TEA

Using the TEA mechanism, we will now describe how TEP—a modified version of the PBC protocol—avoids man-in-the-middle attacks.

[2] Two of the slots are for the direction bit, and the remaining 142 are for the bit-balanced hash bits. More specifically, the bit balancing algorithm, in §5.1.2, takes N input bits and outputs $N + 2\lceil logN \rceil$ bits. Since the hash is a 128 bit function, the bit balancing algorithm produces 142 bit balanced hash bits.

Once the button is pressed on the enrollee, the enrollee repeatedly scans the 802.11 channels in a round robin manner, as in the current PBC protocol. On each channel, the enrollee transmits a TEA request, i.e., a TEA message with the direction flag set to "10". The TEA request contains the enrollee's public key (and any PBC information included in an enrollee's probe). If an adversary continuously occupies the medium for *tx_tmo* (e.g., 1 second), the enrollee overrides carrier-sense and transmits its message anyway. The enrollee then waits for a TEA response from a registrar, which is required to immediately respond. The enrollee records the responses, if any, and after a specified period on each channel it moves to the next 802.11 channel and repeats the process. The enrollee continues to cycle through all 802.11 channels for PBC's walk time period. The enrollee's logic corresponds to the following pseudo-code to build up r, the set of registrar responses:

```
r ← ∅
for 120 sec + #channels × (tx_tmo + 2 × tea_duration)
    do                        ▷ walk time + max enrollee scan period
    switch to next 802.11 channel
    h ← TEA_RECV_START (reply)
    TEA_SEND (request, enroll_info, now + tx_tmo)
    SLEEP (tea_duration)
    r ← r ∪ TEA_RECV_GET (h)
end for
```

A registrar follows a similar protocol. Once the PBC button is pressed, the registrar starts listening for possible TEA requests on its 802.11 channel. Every time a TEA message is received, the registrar records the message payload, and immediately sends its own TEA message in response, containing the registrar's public key. It is safe to reply immediately because the sender's TEA message ended with a CTS-to-SELF, which reserved the medium for the registrar's reply. The registrar's pseudo-code to build up e, the set of enrollee messages, is as follows:

```
e ← ∅
h ← TEA_RECV_START (request)
for 120 sec + #channels × (tx_tmo + 2 × tea_duration)
    do                        ▷ walk time + max enrollee scan period
    m ← TEA_RECV_GET (h)
    if m ≠ ∅ then             ▷ enrollee, RETRY, or OVERLAP
        e ← e ∪ m
        TEA_SEND (reply, registrar_info, now)
                                   ▷ send reply immediately
    end if
end for
```

After the PBC's walk time expires, both the enrollee and the registrar check the list of received messages. Successful pairing requires that both the enrollee and the registrar receive exactly one unique public key via TEA messages, and that no messages were tampered with (i.e., TEA_RECV_GET never returned RETRY or OVERLAP). If exactly one public key was received, it must have been the public key of the other party, and TEP can safely proceed with pairing. If more than one public key was received, or RETRY or OVERLAP was returned, then a session overlap error is raised, indicating that more than one pair of devices may be attempting to pair, or that an adversary is mounting an attack. In this situation, the user must retry pairing.

5.2.1 Reducing Medium Occupancy

The protocol described above is correct and secure (as we will prove in §7.1). However, it can be inefficient if somehow multiple registrars transmit overlapping replies at almost the same time. Each of them will then assume it may have missed a request from some enrollee (since it sensed a concurrent TEA message), and each will re-send its reply. This cycle may continue for the walk time of 120 seconds, unnecessarily occupying the wireless medium. In Appendix B, we describe an optimization that avoids this situation and we prove that the optimized protocol maintains the same security guarantees.

5.3 Example scenarios

Figure 4 shows how TEP works in five potential scenarios. In scenario (a), there is no attacker. In this case, the enrollee sends a request to which the registrar replies immediately. The two devices can thus proceed to complete pairing after 120 seconds. In scenario (b), the enrollee transmits its request, but the attacker immediately jams it so that the registrar can not decode the enrollee's request. However, the registrar detects a long burst of energy, which the registrar interprets as a TEA announcement, causing it to reply to the enrollee.

In scenario (c), the enrollee sends the request; the attacker then captures the medium at the same time as the registrar, and transmits a reply, at a high power, impersonating the registrar. Because of capture effect, the enrollee decodes the message payload from the attacker. But since the registrar and the attacker transmit the hash function of different messages in the ON-OFF slots, the enrollee notes that the slots do not have equal number of zeros and ones and hence detects tampering with the announcement.

In scenario (d), the adversary sends a request message in an attempt to gain access to the registrar; as stipulated by TEP, the registrar replies to this request. However, since the registrar waits for 120 seconds before completing the pairing, it also hears the request from the enrollee. Since the registrar receives requests from two devices, it raises a session overlap error.

Finally, in scenario (e), the adversary sends a TEA request, receives the registrar's reply, and then continuously jams the enrollee using a directional antenna. By using a directional antenna, the adversary ensures that the registrar does not detect the jamming signal and hence does not interpret it as an invalid TEA. The enrollee carrier-

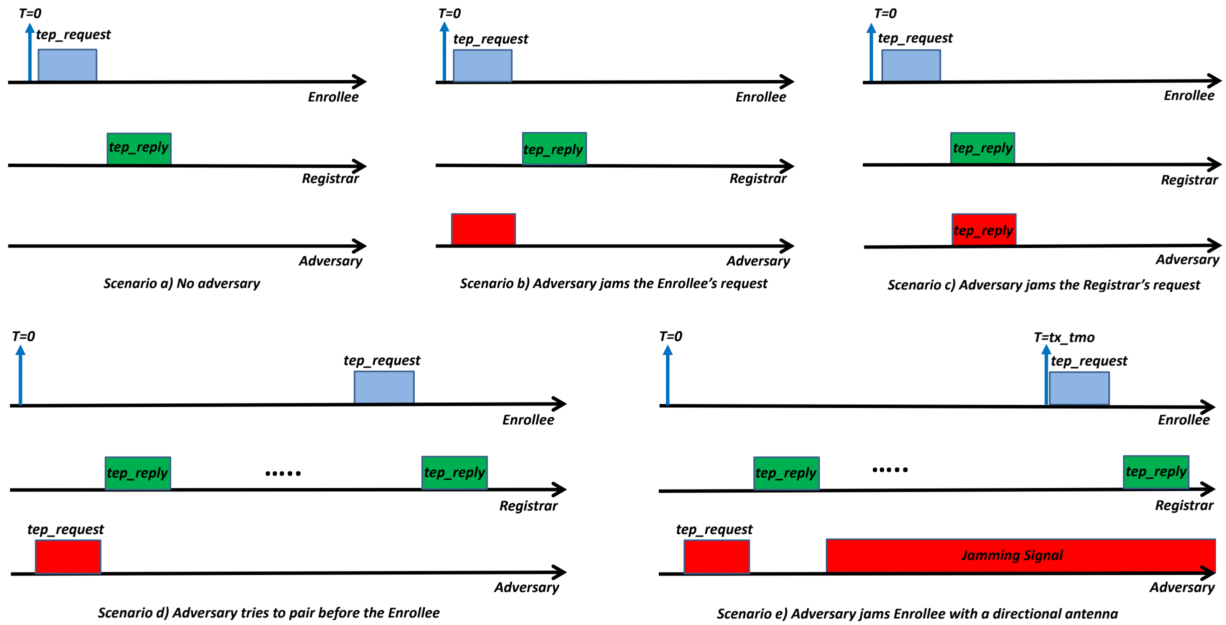

Figure 4: Timelines of five example runs of the TEP protocol.

senses, detects that the medium is occupied, and does not transmit until it times out after *tx_tmo* seconds, at which point it ignores carrier sense and transmits its TEA request. The registrar listens to this request message and detects the presence of the enrollee. Since the registrar receives requests from two devices, it raises a session overlap error.

5.4 Making Pairing Faster

The extension of PBC to use TEA, described above, requires the enrollee and registrar to wait for 120 seconds before completing the association process. If the enrollee does not wait for a full 120 seconds, and simply picks the first responding registrar, it may pick an adversary's registrar—a legitimate registrar only replies when its PBC button has been pushed, and the user might push the registrar's PBC button slightly later than the enrollee's. Because the enrollee does not know if the user has already pushed the registrar's button, it has to wait for 120 seconds to be sure that the user has pushed the button. In this section, we describe how one can eliminate this delay.

First, if the user always pushes the enrollee's button before the registrar's button, then the registrar does not need to wait for 120 seconds; the registrar needs to wait for just the time it takes an enrollee to cycle through all of 802.11's channels (which is less than 12s). Second, we can also eliminate the enrollee's wait time. Specifically, if the user explicitly tells the enrollee that the registrar's button was pushed, the enrollee can complete the association process after one cycle through the 802.11 channels, eliminating the additional wait time.

For example, one approach would be to have the user first press the button on the enrollee, then press the button on the registrar, and then again push the button on the enrollee. Note that, in this approach, the registrar does not have to wait for 120 seconds: because the registrar's button is always pushed after the enrollee, the registrar knows that the enrollee is active, and is guaranteed to see the enrollee's TEA message within the time required for the enrollee to cycle through all 802.11 channels. (Of course, if the 120 second period expires on the enrollee without any additional button pushes, the enrollee can proceed to completion as before, with 2 total button pushes from the user.)

6 TEA ON OFF-THE-SHELF HARDWARE

We implement TEA on Atheros AR5001X+ chipsets by modifying the ath5k driver, and running TEA's timing-sensitive code in a kernel driver.

6.1 Scheduling Slot Transmission

To reduce the air time of a TEA, we must minimize the size of a single slot packet in the ON-OFF slots. Since the slot packet's payload need not be decoded (just the presence or absence of a slot packet conveys a 1 or 0 bit), we transmit slot packets at the highest bitrate, 54 Mbps, for a total of 40 μs.

In addition to reducing the size of a slot packet, TEA must transmit slot packets at precise slot boundaries. Queueing in the kernel and carrier-sense in the card make precise transmission timing challenging. We avoid kernel queueing by implementing TEA in a kernel driver and using high-resolution timers. We avoid delays in the wireless card itself through several changes to the card firmware and driver, as follows. For the duration of the slots, we disable binary exponential backoff (802.11 BEB) by setting CW_{MIN} and CW_{MAX} to 1. To prevent carrier-sense backoff, we disable automatic noise calibration by setting the noise floor register to "high". We place slot packets in the high-priority queue. Finally, we disable the transmitter's own beacons by disabling the beacon queue. In aggregate, these changes allow us to make slot packets as short as 40 μs and maintain accurate slot timing.

6.2 Energy Detection at the Receiver

A TEA receiver detects a synchronization packet and distinguishes ON from OFF slots by checking the energy level on the medium. Hence, the receiver needs to distinguish the noise level, which is around -90dB, from an actual transmission. To do this, we set the noise floor to -90dB and deactivate auto-calibration while running TEP.[3]

While an ideal receiver would detect energy at the finest resolution (i.e., every signal sample), existing wireless chipsets do not give access to these samples. Instead, we exploit two registers provided by the ath5k firmware: *AR5K_PROFCNT_CYCLE* and *AR5K_PROFCNT_RXCLR*. The first register is incremented every clock cycle based on the clock on the wireless hardware. The second register on the other hand is increment only if the hardware finds high energy during that clock cycle.

Using these registers, we define a *sensing window (SW)* as the interval over which the receiver collects aggregate information for whether the medium is occupied or silent, as defined in Table 1. At the beginning of a SW, a TEA receiver resets both registers to 0, and reads them at the end of the SW. The ratio of these two registers at the end of the SW, $\frac{AR5K\_PROFCNT\_RXCLR}{AR5K\_PROFCNT\_CYCLE}$, is defined as the *fractional occupancy*. By putting a threshold on the fractional occupancy, a TEA receiver can detect whether the medium is occupied in a particular SW, and hence can detect energy bursts and measure their durations in units of the sensing window. Similar to the sender, a TEA receiver runs in the kernel to precisely schedule sensing windows.

Our implementation dynamically adjusts the length of the sensing window to minimize system overhead. The TEA receiver uses a long sensing window of 2 ms, until it detects a burst of energy longer than 17 ms. This indicates a synchronization packet, at which point the receiver switches to a 20 μs sensing window to accurately measure energy during slots, providing on average two sensing window measurements for every slot.

[3] There is a tradeoff between the noise floor and the permissible distance between the pairing devices. In particular, pairing devices separated by large distances have a weak signal and hence, to ensure detection, the noise floor should be set to a low value. On the other hand, pairing devices that are closer have a stronger signal, and hence the noise floor can be set to a higher value. We pick -90dB because it is the default noise floor value in typical WiFi implementations. Manufacturers, however, can pick a higher default value, as long as the pairing devices are placed closer to each other.

The receiver must be careful to ensure that a 20 μs sensing window allows accurate detection of slot occupancy. But, because the sender and receiver are not synchronized, sensing windows may not be aligned with slots, and in the worst case, will be off by half a sensing window, i.e., 10 μs. However, having a sensing window that is half the length of a slot ensures that at least one of every two sensing windows is completely within a slot (i.e., does not cross a slot boundary). Thus, to measure slot occupancy, the receiver compares the variance of odd-numbered sensing window measurements and even-numbered sensing window measurements, and uses the one with the highest variance. Because the slots are bit-balanced, the correct sequence will have an equal number of ones and zeros, having the higher variance.

This technique for measuring slot occupancy is secure in the presence of an adversary. As we will prove in Proposition 7.1, an adversary can introduce energy, but cannot cancel energy in an occupied slot. Thus, the adversary can only increase – but cannot reduce– the computed occupancy ratios in either the odd or the even windows. As a result, the adversary cannot create a different bit sequence in either the odd or even windows which still has an equal number of ones and zeros. Thus, sampling at twice the slot rate maintains TEA's security guarantees.

6.3 Sending A Synchronization Packet

To transmit a long synchronization packet, TEA transmits the maximum-sized packet allowed by our hardware (2400 bytes) at the lowest bit rate (1 Mbps), resulting in a 19 ms synchronization packet. While many receivers drop such long packets (the maximum packet size permissible by the higher layers is 1500 bytes), this does not affect a TEA receiver, since it does not need to decode the packet; it only needs to detect a long burst of energy.

6.4 Checking for TEA While Transmitting

While executing the TEP protocol (which lasts for 120 seconds), a node must detect TEA messages transmitted by other nodes even if they overlap with its own transmissions. We distinguish two cases: First, when the node transmits a standard 802.11 packet, it conservatively assumes that the channel has been occupied by part of a synchronization packet for the duration of its transmission. The node samples the medium before and after its transmission, checking for continuous occupancy by a synchronization packet. As our evaluation shows (§7.3), the longest packets in operational WiFi networks are about 4 ms (a collision of two packets sent at the lowest 802.11g rate of 6 Mb/s), making synchronization packet false positives unlikely even with the conservative assumption that the entire 4 ms transmission overlapped with part of a synchronization packet (19 ms).[4]

Second, a node that is transmitting a TEA request must not miss a concurrently transmitted TEA reply, and similarly a node that is transmitting a reply must not miss a concurrent request. To detect partially-overlapping TEA messages, a node samples the medium before and after every synchronization packet, and after the slots of every TEA message, and if it detects energy, it assumes that it may have missed an overlapping TEA message (and thus, TEA_RECV_GET will return OVERLAP, unless it observes other possibly-missed messages, in which case it will return RETRY.) Since the total length of the ON-OFF slots is shorter than the length of the synchronization packet, sampling the medium after the end of a synchronization packet (i.e., before the start of the payload and slots) and after the end of the slots suffices to detect an overlapping synchronization packet. Finally, in the case when two TEA messages are perfectly synchronized, the node uses the direction bits to detect a collision. Since the direction flag for a request is "10" and a reply "01", the node checks for this scenario by checking the energy level during the OFF slot in the direction field in its own transmission. If the OFF slot shows a high energy level, TEA_RECV_GET will return OVERLAP (or RETRY, if there are other missed messages).

7 EVALUATION

We evaluate TEP along three axes: security, accuracy, and performance. Our findings are as follows:

- TEP is provably secure to MITM attacks.
- TEP can be accurately realized using existing OS and 802.11 hardware. Specifically, our prototype sender can schedule ON-OFF slots at a resolution of 40μs, and its 95^{th} percentile scheduling error is as low as 1.65μs. Our prototype receiver can sense the medium's occupancy over periods as small as 20μs and can distinguish ON slots from OFF slots with a zero error rate.
- Results from two operational networks—our campus network and SIGCOMM 2010—show that TEP never confuses cross traffic for an attack. Further, even in the presence of Bluetooth devices which do not obey CTS-to-SELF and may transmit during TEP's OFF slots, TEP can perform key exchange in 1.4 attempts, on average.

7.1 Evaluating TEP's Security

We analyze TEP's security using the threat model in §4.1. To do so, we formally state our definitions, then prove that a TEA is tamper resistant and that wireless pairing using TEP is secure to MITM attacks.

[4] Note that even if some networks have normal packets that are much larger than 4 ms, this may create false positives but does not affect the security of the protocol.

Definition *Tamper evident:* A message is said to be tamper evident if an adversary can neither change the message's content without being detected nor hide the fact that the message has been transmitted.

Before we proceed to prove that a TEA is tamper evident we first prove the following proposition about the capability of an adversary.

Proposition 7.1 *Let $s(t)$ be the transmitted signal, and $h(t)$ be the channel impulse function. Assuming the transmitted signal is unpredictable, and the receiver is within radio range of the sender, an adversary cannot cancel the signal energy at the receiver even if he knows the channel function between the sender and receiver, $h(t)$.*

Proof The received signal is a convolution of the transmitted signal and the channel impulse function, plus the adversary's signal $a(t)$, plus white Gaussian noise $n(t)$, i.e., $r(t) = h(t) * s(t) + a(t) + n(t)$. To cancel the received energy, the adversary needs to produce a signal $a(t)$ so that $r(t) \approx n(t)$, or equivalently, $h(t) * s(t) + a(t) \ll n(t)$. Since the receiver is within radio range of the sender, we know $h(t) * s(t) \gg n(t)$, and, since $n(t)$ is physically unpredictable, that $a(t) \approx -h(t) * s(t)$. But an adversary that can compute such an $a(t)$ directly contradicts our assumption that $s(t)$ is unpredictable, and thus an adversary cannot compute such an $a(t)$. $\square$

Since the synchronization packet and ON slots have random contents, Prop. 7.1 implies that an adversary cannot hide the channel energy during the transmission of the synchronization packet or the ON slots from a receiver. Based on this result we proceed to prove the following:

Proposition 7.2 *Given the transmitter and receiver are within range, and the receiver is sensing the medium, a TEA, described in 5.1, is tamper evident.*

Proof We prove Prop. 7.2 by contradiction. Assume that one party, Alice, sends a TEA to a second party, Bob. Suppose that Alice's TEA to Bob fails to be tamper-evident. This can happen because the adversary succeeds either in hiding from Bob that Alice sent a TEA, or in changing the TEA content without being detected by Bob. To hide Alice's TEA, the adversary must convince Bob that no synchronization packet was transmitted. This requires the adversary to cancel the energy of the synchronization packet at Bob, which contradicts Prop. 7.1. Thus, the adversary must have changed the announcement content.

Suppose the adversary changed the data encoded in the slots. Prop. 7.1 says that the adversary cannot cancel the energy in an ON slot, and hence cannot change an ON slot to an OFF slot. Since the number of ON and OFF slots is balanced, the adversary cannot change the slots without increasing the number of ON slots, and thus being detected. Thus, the only alternative is that the adversary must have changed the message packet. Since the ON-OFF slots include a cryptographic hash of the message, this means that the adversary constructed a different message packet with the same hash as the original message packet. This contradicts our assumption that the hash is collision-resistant. Thus, the adversary cannot alter the announcement content, and TEA is tamper-evident. □

Although Prop. 7.2 guarantees that a TEA message is tamper-evident if the receiver is sensing the medium, the receiver may be transmitting its own message at the same time. We now prove that a TEA is tamper-evident even if the receiver transmits its own messages.

Proposition 7.3 *Given a receiver (Bob) that can send its own messages, a TEA sent by a transmitter (Alice) in range of the receiver is tamper-evident, if the receiver follows the concurrent-transmission protocol of* §*6.4, and the receiver and transmitter send TEA messages with different directions (request or reply).*

Proof If Bob detects the synchronization packet (SP) of Alice's TEA, the TEA is tamper-evident: either Bob will refrain from sending during that TEA, in which case Prop. 7.2 applies, or Bob will transmit concurrently, and TEA_RECV_GET will return RETRY or OVERLAP .

If Bob fails to detect Alice's SP, it must have happened while Bob was sending his own message (otherwise, Prop. 7.2 applies). Since regular 802.11 packets are shorter than a SP, and §6.4 conservatively assumes the medium was occupied for the entire duration of the transmitted packet, Bob could not have missed a SP while sending a regular packet. Thus, the only remaining option is that Alice's SP overlapped with a TEA sent by Bob.

Consider four cases for when Alice's SP was sent in relation to the SP of Bob's TEA. First, if Alice's SP started before Bob's SP, Bob would detect energy before starting to transmit his SP and return OVERLAP or RETRY (§6.4), making the TEA tamper-evident. Second, if Alice's SP started exactly at the same time as Bob's SP, Bob would detect energy during the direction bits and return OVERLAP or RETRY (§6.4), making the TEA tamper-evident. Third, if Alice's SP started during Bob's SP, Bob would detect energy after his SP and return OVERLAP or RETRY (§6.4), making the TEA tamper-evident. Fourth, if Alice's SP started after Bob's SP ended, Bob would detect energy from Alice's SP after the end of his TEA slots and return OVERLAP or RETRY (§6.4), making the TEA tamper-evident. Thus, in all cases, the TEA is tamper-evident. □

We now prove TEP is secure against a MITM attack.

Proposition 7.4 *Suppose an enrollee and a registrar are within range, both are following the TEP protocol as described in* §*5.2 and the user does the stipulated actions required by PBC. Under the threat model defined in* §*4.1, an adversary cannot convince either the enrollee or the registrar to accept any public key that is not the legitimate public key of the other device.*

Proof We prove Prop. 7.4 by contradiction, considering first the registrar, and then the enrollee.

First, suppose an adversary convinces the registrar to accept a public key other than that of the enrollee. By §5.2, this means the registrar received exactly one public key (and, thus, did not receive the enrollee's key), and TEA_RECV_GET never returned OVERLAP or RETRY. By assumption, the enrollee and registrar entered PBC mode within 120 seconds of each other, which means they were concurrently running their respective pseudo-code for at least $\#channels \times (tx\_tmo + 2 \times tea\_duration)$ seconds, and therefore the enrollee must have transmitted at least one TEA message on the registrar's channel while the registrar was listening. Prop. 7.3 guarantees that the registrar must have either received that one message, or detected tampering (and returned OVERLAP or RETRY), which contradicts our assumption that the registrar never received the enrollee's message and never returned OVERLAP or RETRY. Thus, an adversary cannot convince the registrar to accept a public key other than that of the enrollee.

Second, suppose an adversary convinces the enrollee to accept a public key other than that of the registrar. By §5.2, this means that the enrollee received exactly one public key response to its requests (and, thus, did not receive the registrar's key), and TEA_RECV_GET never returned OVERLAP or RETRY. As above, there must have been a time when the registrar was listening, and the enrollee transmitted its request message on the registrar's channel. Prop. 7.3 guarantees that the registrar must have either received the enrollee's message, or detected tampering (and returned OVERLAP or RETRY). In both of those cases, §5.2 requires the registrar to send a reply. Prop 7.3 similarly guarantees that the enrollee must have either received the registrar's reply, or detected tampering (and returned OVERLAP or RETRY), which directly contradicts our supposition. Thus, an adversary cannot convince the enrollee to accept a public key other than the registrar's, and TEP is secure. □

7.2 Evaluating TEP's Accuracy

We check whether TEP can be accurately realized using existing operating systems and off-the-shelf 802.11 hardware. Our experiments use our Ath5K prototype described in §6 and run over our campus network. Figure 5 shows the locations of the TEP nodes, which span an area

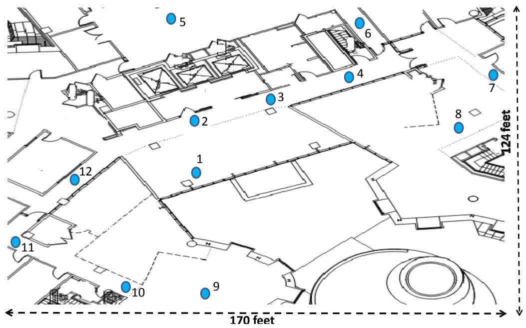

Figure 5: **Locations of nodes (indicated by blue circles) in our experimental testbed, which operates as part of our campus network.**

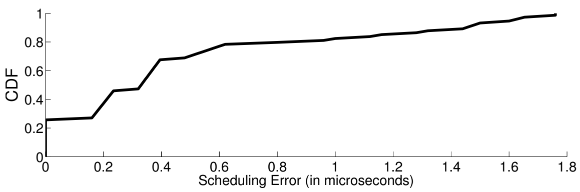

Figure 6: **CDF of TEP slot scheduling errors. The figure shows that the maximum scheduling error is 1.8 μs which is significantly lower than the slot duration of 40μs.**

of 21,080 square feet (1,958 m^2) with both line-of-sight and non-line-of-sight links.

7.2.1 Transmitter

The performance of TEP hinges on the transmitter accurately scheduling the transmission of the ON-OFF slots. The difficulty in accurate scheduling arises from the fact that we want to implement the protocol in software using standard 802.11 chipsets. Hence, we are limited by the operating system and the hardware interface. For example, if the kernel or the hardware introduces extra delays between the slot packets, it will alter the bit sequence conveyed to the receiver, and will cause failures. Given that our slot is 40μs, we need an accuracy that is on the order of few microseconds. Can we achieve such an accuracy with existing kernels and chipsets?

Experiment. We focus on the most challenging ON-OFF slot sequence from a scheduling perspective: alternating zeros and ones which requires the maximum scheduling precision. We set the slot time to 40μs, by sending a packet at the highest bitrate of 54 Mbps. To measure the produced slots accurately, we capture the signal transmitted by our 802.11 sender using a USRP2 software radio board [9]. Our USRP2 board can measure signal samples at a resolution of 0.16 μs, allowing us to accurately compute the duration of the produced slots. We run the experiment 1000 times for each sender in our testbed and measure the exact duration of every slot. We then compute the scheduling error as the difference between the measured slot duration and the intended 40 μs.

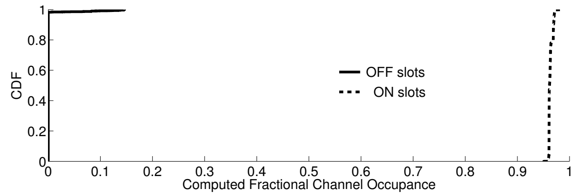

Figure 7: **CDFs of the fractional occupancy during ON slots and OFF slots. The figure shows that the two distributions have no overlap and hence the receiver cannot confuse ON and OFF slots.**

Results. Fig. 6 shows the CDF of slot scheduling errors. The figure shows that the median scheduling error is less than 0.4 μs and the maximum error is 1.8 μs. Thus, despite operating in software and with existing chipsets, a TEP sender can accurately schedule the ON-OFF slots at microsecond granularity.

7.2.2 Receiver

TEP's security depends on the receiver's ability to distinguish ON slots from OFF slots. In this section, we check that given that the receiver is within the sender's radio range (i.e., can sense the sender's signal), it can clearly distinguish ON slots from OFF slots.

Experiment. In each run, the sender sends a sequence of alternating ON-OFF slots, using a slot duration of 40 μs. The receiver uses a sensing window of 20μs to measure fractional occupancy. This means the receiver has twice as many measurements of fractional occupancy as there are slots. As explained in §6.2, the receiver keeps either the odd or even measurements depending on which sequence has higher variance. Hence, for each slot, the receiver has exactly one fractional occupancy measurement. We then compare the measured fractional occupancy for known ON slots vs. known OFF slots to determine if the receiver can reliably distinguish between them based on measured fractional occupancy. We randomly pick two nodes in the testbed to be sender and receiver, and repeat the experiment for various node pairs in the testbed.

Results. Fig. 7 plots the CDFs of fractional occupancy for ON slots and OFF slots. The figure shows that the two CDFs are completely separate; that is, there is no overlap in the values of fractional occupancy that correspond to OFF slots and those that correspond to ON slots. Hence, by looking at the fractional occupancy the receiver can perfectly distinguish the ON slots from OFF slots. This result shows that a TEP receiver based on current OSes and 802.11 hardware can accurately decode the ON-OFF slots necessary for the TEP protocol.

7.3 Evaluating TEP's Performance

We are interested in how TEP interacts with cross traffic in an operational network. Cross traffic does not hamper TEP's security (the proofs in §7.1 apply in the presence of cross traffic). However, cross traffic may cause *false*

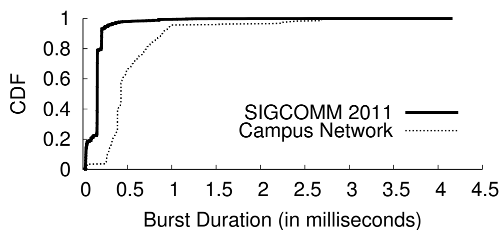

Figure 8: CDF of the duration of energy bursts in the SIGCOMM 2010 network and our campus network. The figure shows that energy bursts caused by normal traffic are much shorter than a TEP synchronization packet (19 ms). Thus, it is unlikely that TEP will confuse normal traffic as a synchronization packet.

positives, where a node incorrectly declares that a TEP message has been tampered with by an adversary. Such events can unnecessarily delay secure pairing.

We investigate TEP's interaction with cross traffic using results from two operational networks: the SIGCOMM 2010 network, which is a heavily congested network, and our campus network, which is a moderately congested network. As in §7.2, our experiments use our modified Ath5k driver on AR5001X+ Atheros chipsets. In addition to cross-traffic on the TEP channel, both networks carried traffic on adjacent 802.11 channels.

7.3.1 Impact of Cross Traffic on a Sync Packet

In TEP, a receiver detects a TEA if the medium is continuously occupied for a period longer than the duration of a synchronization packet (19 ms). We would like to check that a receiver is unlikely to encounter false positives while detecting synchronization packets. False positives could occur in two scenarios: either (1) legitimate traffic includes such continuous long bursts of energy, or (2) a TEP receiver is incapable of detecting the short DIFS intervals that separate legitimate packets, and mistakes a sequence of back-to-back WiFi packets as a continuous burst of energy.[5] We empirically study each case below.

Experiment 1. We first check whether legitimate traffic can cause the medium to be continuously occupied for a duration of 19 ms. We use two production networks: our campus network and the SIGCOMM 2010 network. Since we would like to capture all kinds of energy bursts, including collisions, we sense the medium using USRP2 radios. USRP2s allow us to directly look at the signal samples and hence are much more sensitive than 802.11 cards. We used a USRP2 board to eavesdrop on the channel on which these networks operate and log the raw signal samples. In order to compute the length of bursts on the channel, we need to be able to identify the beginning of a burst and its end in an automated way. To do so, we use the double sliding window packet detection algorithm[6] typically used in hardware to detect packet arrivals [23]. We collected over a million packets on the SIGCOMM network and about the same number on our campus network. We processed each trace to extract the energy bursts and their durations (as explained above) and plot the CDF of energy burst durations in Fig. 8.

Result 1. The results in Fig. 8 show that all energy bursts in both networks lasted for less than 4.3 ms, which is much shorter than a TEP synchronization packet. In particular, the majority of energy bursts last between 0.25 ms and 2 ms. This corresponds to a packet size of 1500 bytes transmitted at a bit rate between 6 Mb/s and 48 Mb/s, which spans the range of 802.11g bit rates. A few bursts lasted for less time which are likely to be short ACK packets. Also a few bursts have lasted longer than 2 ms. Such longer bursts are typically due to collisions. Fig. 9 illustrates this case, where the second packet starts just before the first packet ends, causing a spike in the energy level on the channel. Soon after, the first packet ends, causing the energy to drop again, but the two transmissions have already collided.[7] Interestingly, the bit rates used in our campus network are lower than those used at SIGCOMM. This is likely because at SIGCOMM, the access point was in the conference room and in line-of-sight of senders and receivers, while in our campus, an access point serves multiple offices that span a significant area and are rarely in line-of-sight of the access point.

Overall, the results in Fig. 8 indicate that bursts of energy in today's production networks have significantly shorter durations than TEP's synchronization packet, and hence are unlikely to cause false positives.

Experiment 2. The second scenario in which a node may incorrectly detect a synchronization packet occurs when the node confuses a sequence of back-to-back packets separated by DIFS as a single continuous energy burst. Thus, we evaluate our prototype's ability to distinguish a synchronization packet from a stream of back-to-back 802.11 packets. To do so, we randomly pick two random nodes in our testbed in Fig. 5, and make one node transmit a stream of back-to-back 1500-byte packets at the lowest rate of 1 Mbps, while the other node senses the

[5] A data packet and its ACK are separated by a SIFS, which is smaller than a DIFS, but ACKs are short packets and the next data packet is separated by a DIFS. Hence the maximum packing occurs with back-to-back data packets without ACKs.

[6] The double sliding window algorithm compares the energy in two consecutive sliding windows. If there is no packet, i.e., the two windows are both capturing noise, the ratio of their energy is around one. Similarly, if both windows are already in the middle of a packet, their relative energy is one. In contrast, when one window is partially sliding into a packet while the other is still capturing noise, the ratio between their energy starts increasing. The ratio spikes, when one window is fully into a packet while the other is still fully in the noise, which indicates that the beginning of the packet is at the boundary between the two windows. Analogously, a steep dip in energy corresponds to the end of a packet [23].

[7] Collisions of two 1500-byte packets transmitted at 6 Mb/s may be slightly longer than 4 ms because of the additional symbols corresponding to link layer header and trailer, and the PHY layer preamble.

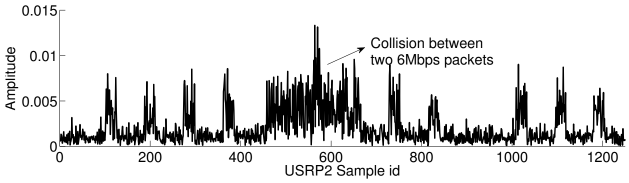

Figure 9: **The energy pattern of the maximum energy burst in the SIGCOMM trace. The figure indicates that such relatively long bursts are due to collisions at the lowest bit rate of 6 Mb/s. The other spikes correspond to packets sent at higher bit rates.**

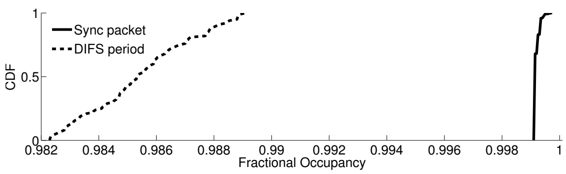

Figure 10: **CDF of fractional occupancy measured by a receiver for transmissions of either a synchronization packet or a sequence of back-to-back 1500-byte packets separated by DIFS. The figure shows a full separation between the two CDFs, indicating that a TEP receiver does not confuse back-to-back packets as a synchronization packet.**

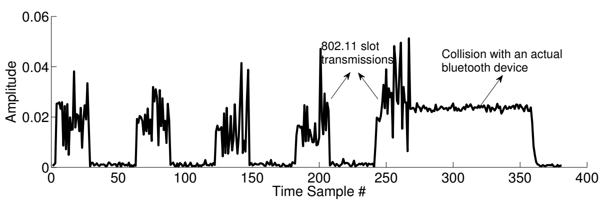

Figure 11: **Energy pattern for TEA slots in the presence of a Bluetooth device causing interference.**

medium using the default sensing window of 2 ms. We then make the same sender transmit a stream of synchronization packets while the receiver senses these packets using a 2 ms window. For both cases, we compute the fractional occupancy in each sensing window. We repeat the experiment with multiple node pairs and compare the fractional occupancy during back-to-back packets and synchronization packets.

Result 2. Fig. 10 compares the CDF of the fractional occupancy during a synchronization packet and the CDF of the fractional occupancy when the sensing window includes back-to-back packets separated by a DIFS,[8] taken over 100K synchronization packets and 100K DIFS occurrences. The figure shows that the two CDFs are sufficiently separate making it unlikely that TEP confuses back-to-back packets as a synchronization packet.

[8] Sometimes the DIFS may be split between two consecutive sensing windows, in this case we include in the CDF whichever of these two window has the lower fractional energy. This is because it is sufficient that one sensing window shows a relatively low fractional occupancy to declare the end of energy burst.

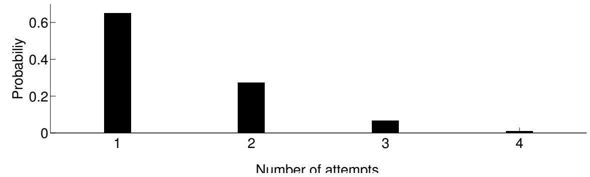

Figure 12: **Number of attempts required for TEP to successfully pair in the presence of an interfering Bluetooth device.**

7.4 Performance with Non-802.11 Traffic

Finally, while 802.11 nodes comply with the rules of CTS-to-SELF, and abstain from transmitting during TEA's ON-OFF slots, other devices may continue to transmit, causing TEA nodes to detect tampering. Fig. 11 shows a collision between a TEA and a Bluetooth transmission from an Android phone as captured by a USRP2. Bluetooth devices do not typically decode 802.11 CTS-to-SELF packets, and hence, as shown in the figure, end up transmitting during the ON-OFF slots. In this section we examine the impact of a nearby Bluetooth device on TEA.

Experiment. We place a TEA sender in location 1 (Fig. 5) and make other nodes act as TEA receivers. We co-locate a Bluetooth device next to the TEA sender. The sender periodically sends an announcement. The receivers first detect the synchronization packets, decode the CTS-to-SELF, and then try to verify the slots. If the receiver can successfully verify, it declares success. Otherwise, it attempts to verify the slots in the next time period.

Results. Fig. 12 shows the CDF of the number of required attempts before a TEA receiver succeeds in receiving a correct TEA. Bluetooth transceivers operate on 79 bands in 2402-2480 MHz and frequently jump across these bands. Thus, the probability that they interfere with TEA in successive runs of the protocol is relatively low. The figure shows that, even in the presence of Bluetooth devices which cannot decode a CTS-to-SELF, a TEA receiver requires 1.4 attempts on average, and 4 attempts maximum, before it receives the announcement.

8 CONCLUSION

This paper presented Tamper-Evident Pairing (TEP), the first wireless pairing protocol that works in-band, with no pre-shared keys, and protects against MITM attacks. TEP relies on a Tamper-Evident Announcement (TEA) mechanism, which guarantees that an adversary cannot tamper with either the payload in a transmitted message, or with the fact that the message was sent. We formally proved that the design protects from MITM attacks. Further, we implemented a prototype of TEA and TEP for the 802.11 wireless protocol using off-the-shelf WiFi devices, and showed that TEP is practical on real-world 802.11 networks and devices.

ACKNOWLEDGMENTS

We thank Ramesh Chandra, James Cowling, Haitham Hassaneih, Nate Kushman, Jad Naous, Benjamin Ransford, and our shepherd Diana Smetters for their insightful comments. We also thank Jukka Suomela and Piotr Indyk for help with the efficient bit-balancing algorithm in the Appendix. This work is funded by NSF and SMART-FM.

REFERENCES

[1] Atheros linux wireless driver. http://wireless.kernel.org/en/users/Drivers/ath5k.

[2] D. Balfanz, G. Durfee, D.K.Smetters, and R. Grinter. In search of usable security – five lessons from the field. *IEEE Journal on Security and Privacy*, 2(5):19–24, September–October 2004.

[3] S. M. Bellovin and M. Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *Proceedings of the 13th IEEE Symposium on Security and Privacy*, Oakland, CA, May 1992.

[4] V. Boyko, P. MacKenzie, and S. Patel. Provably secure password-authenticated key exchange using diffie-hellman. In B. Preneel, editor, *Advances in Cryptology—Eurocrypt 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 156–171. Springer-Verlag, 2000.

[5] M. Čagalj, J.-P. Hubaux, S. Čapkun, R. Rangaswamy, I. Tsigkogiannia, and M. Srivastava. Integrity codes: Message integrity protection and authentication over insecure channels. In *Proceedings of the 27th IEEE Symposium on Security and Privacy*, pages 280–294, Oakland, CA, May 2006.

[6] S. Čapkun, M. Čagalj, R. Rengaswamy, I. Tsigkogiannis, J.-P. Hubaux, and M. Srivastava. Integrity codes: Message integrity protection and authentication over insecure channels. *IEEE Transactions on Dependable and Secure Computing*, 5(4):208–223, October–December 2008.

[7] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, November 1976.

[8] W. Diffie, P. C. van Oorschot, and M. J. Wiener. Authentication and authenticated key exchanges. *Designs, Codes, and Cryptography*, 2(2):107–125, 1992.

[9] Ettus Inc. Universal software radio peripheral. http://ettus.com.

[10] M. T. Goodrich, M. Sirivianos, J. Solis, G. Tsudik, and E. Uzun. Loud and clear: human-verifiable authentication based on audio. In *Proceedings of the 26th International Conference on Distributed Computing Systems*, Lisboa, Portugal, July 2006.

[11] J. D. Halamka. Telemonitoring for the home. http://geekdoctor.blogspot.com/2010/04/telemonitoring-for-home.html, April 2010.

[12] IEEE. 802.15.1 specification: Personal area networks, 2002.

[13] IEEE. 802.11i specification: Amendent 6: MAC security enhancements, 2004.

[14] Kelton Research. Survey: Protecting wireless network an essential element of home security. http://www.wi-fi.org/news_articles.php?f=media_news&news_id=1, November 2006.

[15] C. Kuo, J. Walker, and A. Perrig. Low-cost manufacturing, usability and security: An analysis of bluetooth simple pairing and wi-fi protected setup. In *Proceedings of the Usable Security Workshop*, Lowlands, Scarborough, Trinidad/Tobago, February 2007.

[16] R. Li. WiFi hitting the security camera scene. *eZine Articles*, March 2010. http://ezinearticles.com/?id=3963601.

[17] R. Mayrhofer and H. Gellersen. Shake well before use: Authentication based on accelerometer data. In *Proceedings of the 5th International Conference on Pervasive Computing*, Toronto, Canada, May 2007.

[18] J. M. McCune, A. Perrig, and M. K. Reiter. Seeing-is-believing: using camera phones for human-verifiable authentication. In *Proceedings of the 26th IEEE Symposium on Security and Privacy*, Oakland, CA, May 2005.

[19] D. A. Norman. The way I see it: When security gets in the way. *Interactions*, 16(6), November–December 2009.

[20] V. Roth, W. Polak, E. Rieffel, and T. Turner. Simple and effective defense againgst evil twin access points. In *Proceedings of the 1st ACM Conference on Wireless Network Security*, Alexandria, VA, March–April 2008.

[21] SensorMetrics, Inc. Intellisense WiFi products: Temperature sensors, motion sensors, power sensors. http://www.sensormetrics.com/wifi.html.

[22] F. Stajano and R. Anderson. The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks. In *Proceedings of the 7th International Workshop on Security Protocols*, 1999.

[23] J. K. Tan. An Adaptive Orthogonal Frequency Division Multiplexing Baseband Modem for Wideband Wireless Channels. Master's thesis, MIT, 2006.

[24] C. Ware, J. Judge, J. Chicharo, and E. Dutkiewicz. Unfairness and capture behavior in 802.11 adhoc networks. In *Proceedings of the IEEE International Conference on Communications*, 2000.

[25] WiFi Alliance. WPS Certified Products. http://www.wi-fi.org/search_products.php.

[26] WiFi Alliance. WiFi protected setup specification, version 1.0h, 2006.

[27] WiFi Alliance. WiFi Alliance to ease setup of home WiFi networks with new industry wide program. http://www.wi-fi.org/news_articles.php?f=media_news&news_id=263, January 2007.

A BIT-BALANCING ALGORITHM

TEA's bit-balancing algorithm takes an even number, N, of input bits and produces $M = N + 2\lceil logN \rceil$ output bits which have an equal number of zeros and ones. If the input sequence has an odd number of bits, we pad a 1 bit to it to make it an even length sequence.

Let the input bit sequence of our algorithm be denoted by IN, and the output bit-balanced sequence be denoted by OUT. We define D_0 to be the difference between the number of ones and zeros in the input IN. Also D_i is defined as the difference between the number of ones and zeros after flipping the first i

	Input Sequence:	1 0 0 0,		$D_0 = -2$
$i = 1:$	1 0 0 0	$\rightarrow$	0 0 0 0,	$D_1 = -4$
$i = 2:$	0 0 0 0	$\rightarrow$	0 1 0 0,	$D_2 = -2$
$i = 3:$	0 1 0 0	$\rightarrow$	0 1 1 0,	$D_3 = 0$
	Output Sequence:	0 1 1 0 1 0 0 1		

Table 2: Example run of our 0-1 balanced function

bits in the input *IN*. Our algorithm works as follows.

- **Step 1:** Compute the difference D_0 between the number of ones and number of zeros in *IN*. Set i to 1 and S_0 to *IN*.
- **Step 2:** Flip the i^{th} bit in S_{i-1} to get S_i. Then compute the new difference, D_i as $D_i = D_{i-1} \pm 2$ depending on whether the i^{th} bit is one or zero.
- **Step 3:** If $D_i = 0$, then set *INDEX* to i and OUT_{temp} to S_i and go to Step 4. Otherwise increment i and go to Step 2.
- **Step 4:** Set the output *OUT* to be the concatenation of OUT_{temp} and the Manchester encoding of the bit representation of $INDEX - 1$. Since S_{INDEX} is N bits long and the Manchester encoding of $INDEX - 1$ is $2\lceil log(N) \rceil$ bits long, the output *OUT* is $N + 2\lceil log(N) \rceil$ bits long.

To see how the above algorithm works, let us take the 4 bit input sequence, 1000, shown in Table 2. The difference D_0 for this sequence is -2. In the first iteration, we flip the first bit to get the bit sequence 0000 which has a difference $D_1 = -4$. In the second iteration, we flip the second bit to get 0100 which has a difference $D_2 = -2$. Finally, in the third iteration, we flip the third bit to get 0110 which has a difference $D_3 = 0$. Thus, we output this sequence concatenated with the Manchester encoding of $3 - 1$, which is 1001. Thus, the bit balanced output sequence is 01101001.

The above algorithm relies on the fact that there exists an *INDEX* bit position for which $D_{INDEX} = 0$. Such an *INDEX* always exists for the following reason. First, because the sequence S_0 has an even number of bits, D_0 is even. Further, for every bit flipped, D_i differs from D_{i-1} by exactly ± 2. Finally, since S_N is the bitwise opposite of S_0 and thus $D_N = -D_0$, there must exist an *INDEX* for which $D_{INDEX} = 0$.

Note that this is a one-to-one mapping and the decoding can be done in linear time. Specifically the decoder takes the last $2\lceil log(N) \rceil$ bits and constructs *INDEX* from its Manchester encoding. Then it takes the first N bits and flips the first *INDEX* bits in the first N bits to get the original bit sequence.

B Reducing Medium Occupancy

TEP's specifications in §5.2 ensure that if there is any possibility that a registrar missed a TEA request (i.e., if TEA_RECV_GET returned RETRY or OVERLAP), that registrar will immediately transmit a TEA reply, without regard for carrier-sense. Thus, if, by some chance, multiple registrars transmit overlapping replies at almost the same time, each of them will then assume it may have missed a request from some enrollee (since it sensed a concurrent TEA message), and each will re-send its reply. This cycle of replies may continue until each registrar's PBC walk time (120 sec) expires. This section shows how to modify the basic TEP protocol to avoid occupying the medium for 120 seconds in this situation.

To address this issue, we make two changes to the TEP protocol from §5.2. First, the registrar does not re-transmit replies if all of the possibly-missed TEA requests overlapped with its previous transmission. In other words, the registrar performs TEA_SEND only if $m \neq$ OVERLAP. Not re-sending the reply is safe only if enrollees whose requests may not get a reply also learn of the TEA overlap (and thus return a session overlap error). To guarantee this, we make a second change, to the enrollee, so that it listens for *tea_duration* both before and after transmitting its request. This ensures that an enrollee hears any TEA replies (from registrars) that overlap its own TEA request. (As before, if an enrollee detects a TEA message it cannot decode, it triggers a session overlap error.) Thus, the enrollee pseudo-code is augmented as follows (changing the loop duration and introducing an additional SLEEP before sending):

```
r ← ∅
for 120 sec + #channels × (tx_tmo + 3 × tea_duration) do
                          ▷ walk time + max enrollee scan period
    switch to next 802.11 channel
    h ← TEA_RECV_START (reply)
    SLEEP (tea_duration)
    TEA_SEND (request, enroll_info, now + tx_tmo)
    SLEEP (tea_duration)
    r ← r ∪ TEA_RECV_GET (h)
end for
```

The registrar must also wait for the same increased loop time to accommodate the modified enrollee. With these changes, TEP safely avoids occupying the medium for the whole walk time in cases when multiple registrars hear each other's replies.

B.1 Extending the Security Proof

Next, we prove that the above optimization is secure.

Proposition B.1 *An enrollee and registrar following the optimized TEP protocol (from Appendix B) cannot be tricked into accepting an incorrect public key, as in Prop. 7.4.*

Proof The only change in the optimized protocol that affects the proof for Prop. 7.4 is that the registrar does not resend its reply when $m =$ OVERLAP. The registrar still computes the same set e of enrollee messages as in Prop. 7.4, and therefore cannot be tricked into accepting an incorrect public key.

We prove that the enrollee also cannot be tricked, by contradiction. Suppose that the enrollee is tricked into accepting the wrong key. From the proof of Prop. 7.4, this must be because the registrar did not respond to the enrollee's request. This must be because the registrar's TEA_RECV_GET returned OVERLAP, i.e., the registrar missed zero or more requests, all of which overlapped the registrar's TEA_SEND. Thus, the enrollee must have transmitted its request within *tea_duration* of the registrar transmitting a reply.

By the enrollee pseudo-code in Appendix B, the enrollee was listening for TEA messages for *tea_duration* before and after sending its request. If the enrollee's TEA_RECV_GET returned the registrar's reply, the enrollee could not have accepted a different key (by Prop. 7.4). Thus, the enrollee's TEA_RECV_GET must have returned RETRY or OVERLAP . But in both of these cases, the enrollee would not have accepted any key. Thus, by contradiction, the enrollee cannot be tricked, and the optimized TEP protocol is secure. □