Janne Lindqvist, Tuomas Aura, George Danezis, Teemu Koponen, Annu Myllyniemi, Jussi Mäki, and Michael Roe. 2009. Privacy-preserving 802.11 access-point discovery. Cambridge, United Kingdom. Microsoft Research Technical Report, MSR-TR-2009-7. An abridged version of this article is available in: David Basin, Srdjan Capkun, and Wenke Lee (editors). Proceedings of the Second ACM Conference on Wireless Network Security (WiSec 2009). Zürich, Switzerland. 16-18 March 2009, pages 123-130.

Privacy-Preserving 802.11 Access-Point Discovery (full version)

Microsoft Research Technical Report **MSR-TR-2009-7**
January 2009

Janne Lindqvist
Helsinki University of Technology (TKK), Finland
email: janne.lindqvist@iki.fi

Tuomas Aura
Microsoft Research, Cambridge, UK

George Danezis
Microsoft Research, Cambridge, UK

Teemu Koponen
Helsinki Institute for Information Technology (HIIT), Finland

Annu Myllyniemi
Helsinki University of Technology (TKK), Finland

Jussi Mäki
Helsinki University of Technology (TKK), Finland

Michael Roe
Microsoft Research, Cambridge, UK

Privacy-Preserving 802.11 Access-Point Discovery

(full version)[§]

Janne Lindqvist[*] Tuomas Aura[‡*] George Danezis[‡]
Teemu Koponen[†] Annu Myllyniemi[*] Jussi Mäki[*] Michael Roe[‡]

ABSTRACT

It is usual for 802.11 WLAN clients to probe actively for access points in order to hasten AP discovery and to find "hidden" APs. These probes reveal the client's list of preferred networks, thus, present a privacy risk: an eavesdropper can infer attributes of the client based on its associations with networks. We propose an access-point discovery protocol that supports fast discovery and hidden networks while also preserving privacy. Our solution is incrementally deployable, efficient, requires only small modifications to current client and AP implementations, interoperates with current networks, and does not change the user experience. We prove the security and privacy properties of our protocol, and provide performance measurements based on a prototype implementation.

1 Introduction

WLAN access-point (AP) discovery based on the IEEE 802.11 [19] standard suffers from a well known privacy problem [15, 32, 33]: WLAN clients probe actively for their preferred networks. These *directed active probes* reveal the client's list of preferred network identifiers, SSIDs, to anyone listening. The continuous probing is necessary for fast handoffs when the signal from a previous AP fades. It is also needed to implement the *hidden network* feature popular with many network administrators, in which the AP does not advertise its SSID and waits for probes with the right identifier.

Leaking the list of the client's preferred networks can be a serious privacy problem. The SSIDs are human-readable and often contain names of organizations, companies or government departments, which may leak the client's affiliation with them. Some WLAN clients probe automatically for all previously visited networks, which can act as a map of the user's movements. We illustrate the importance of the problem through three fictional yet plausible examples:

- John works at a major consultancy company, and often visits client sites as part of his work. An eavesdropper observers the probes from John's laptop at a local cafe. He learns the client sites that John has visited and may infer information about their commercial relationships.
- Jenny works at a local hospital. An attacker seeks unauthorized access to patient records. He eavesdrops a coffee-shop network and identifies Jenny's laptop as having been connected to the hospital WLAN. He can then target her for social engineering or steal her laptop in order to extract her credentials for the hospital network.
- Jack works for the government and participates in a conference abroad. A local extremist group detects his association with a foreign government network and targets him for abuse.

In these examples, the network names are leaked by a link-layer network discovery protocol. Thus, higher-layer security or privacy mechanisms, such as encryption or the use of anonymous communications, cannot prevent the information leakage. Strengthening the privacy of access point discovery is therefore a key enabler for higher level privacy protocols.

In the current 802.11 access point discovery schemes, there is a tradeoff between performance and privacy. Directed active probing is a relatively effective way to maintain an up-to-date list of available APs. The client sends a probe message on each radio channel and gets a response only from APs that serve the requested SSID. An alternative to probing is passive scanning, in which the client waits for beacons on each radio channel. This is much slower than probing and may result in lower quality of service. Another possibility is to send undirected active probes that do not specify the SSID and to which all APs respond. This has the problem that it uses more bandwidth and could be slower if there are multiple responses, although not as slow as passive scanning.

It might be possible to develop heuristics that reduce the number of unsuccessful probes compared to current implementations and, thus, also reduce information leakage. A privacy solution that depends on heuristics is, however, fragile and may fail when the operating environment of the protocol changes.

Another tradeoff is between privacy for the client and for the network. Hidden networks avoid their SSID from being shown on the client user interface, which

[*]Helsinki University of Technology (TKK)
[†]Helsinki Institute for Information Technology (HIIT)
[‡]Microsoft Research
[§]An abridged version of this paper appears in Proceedings of ACM WiSec'09 [25]. This version also supersedes previous technical report published in Helsinki University of Technology Department of Computer Science and Engineering series B 3/08.

gives them some privacy, but this comes at the cost of the clients having to send a directed probe to every AP they encounter. There are arguments for why disabling the SSID advertisement is an unreasonable practice and should be banned. (The benefit for the AP is limited because the SSID can be sniffed when as a client probes for it and then associates.) Yet this is unlikely to happen in practice because the feature is widely deployed and hiding the network is commonly recommended as a best practice in wireless-network security [14, 34, 37, 39]. Moreover, many access point owners are focused on the security of their infrastructure rather than that of the clients. They may see hiding the network as defense in depth when used in combination with other mechanisms such as link-layer encryption and MAC-address filtering, even if it adds only little security.

In this paper, we present a practical privacy-preserving access-point discovery protocol that addresses the problems outlined above. We aim to allow active probing and hidden networks, yet protect the privacy of clients. Our protocol does not allow the typical adversary to find out which networks a client is configured to connect to or which networks it has visited previously. For access points, the protocol strengthens network hiding because the SSID cannot be captured even from client probes.

We assume a location-constrained adversary that is able to roam between access points, to record and replay messages sent between honest clients and access points, and to mount man-in-the-middle attacks at a single access point at a time. We do not expect the adversary to be present at multiple networks at the same time or to relay protocol messages between two access-point locations in real time. Thus, we exclude wormhole attacks where the attacker relays the protocol exchanges between a client and remote access points.

To achieve high efficiency the proposed protocol uses only symmetric primitives, such as message authentication codes based on cryptographic hash functions (e.g. HMAC [24]) and symmetric encryption. It also fits into the current 802.11 2-round active discovery protocol, requiring only minimal changes to clients and access points as well as no changes to the user experience for configuring pre-shared keys. In summary, the contributions of this paper are the following:

1. Analysis of the privacy implications of disabling or enabling the SSID broadcast in wireless LANs. Previous work has pointed out the problems. We look for a tradeoff and approach the question with a view towards developing a low-cost solution.

2. Protocol design syntactically based on the ISO/IEC standard 9798-4 entity authentication but with different security requirements, including new privacy requirements.

3. Formal model and verification of the privacy properties of the authentication protocol.

4. Detailed definition of the protocol as a small extension of the discovery protocol of the 802.11 MAC layer. The implementation is compatible with other extensions of 802.11 and other privacy mechanisms for 802.11, such as MAC address randomization (which should be used together with our protocol). The same clients can access both hidden and non-hidden networks.

5. Implementation of the protocol on commercial 802.11 hardware with no significant performance loss.

6. We have taken the system viewpoint, considering the whole system and designing the protocol to work well in that context, instead of just developing a new stand-alone protocol.

Finally, our proposal sets a new base level of privacy protection for future wireless networking standards and privacy extensions to 802.11.

The rest of the paper is organized as follows. In the next section, we give some background on current WLAN access point discovery and privacy. Section 3 describes the design requirements and assumptions. Then, we explain our solution in Section 4 and evaluate its security and performance in Section 5. We discuss further system aspects in Section 6. Related work is surveyed in Section 7. Finally, Section 8 concludes the paper.

2 IEEE 802.11 and Privacy

An infrastructure-mode 802.11 wireless network consists of one or more access points (AP) and some client stations (client STA), which connect to a wired network and the Internet via an AP. Each client station is identified by a hardware MAC address and each AP by a basic service set identifier (BSSID), which typically is equal to the hardware MAC address of the AP's wireless interface. The network, also called an extended service set (ESS), as a whole has a a human-readable name called a service-set identifier (SSID). An AP may sometimes belong to multiple extended service sets and, thus, have more than one SSID. This is typically the case when the wired network is divided logically into several virtual local area networks (VLAN). A standard client may be associated with at most one AP and SSID at a time. (Experimental client implementations have supported multiple simultaneous attachments to different APs [9].) A typical client is configured with a list of SSIDs of the networks it will try to connect when it detects one in the vicinity.

Most clients use the globally unique identifier of their network card as their MAC address. This identifier

identifies the card manufacturer, not the mobile computer or it user, but because the MAC address appears in every frame sent to the network, it can be used to correlate appearances of a personal wireless device and, thus, trace the user. This tracing requires multiple observations at different locations or at different times. As a defense against such tracing, strategies have been devised to periodically randomize the client MAC addresses [17,21], although no such strategy has been yet widely deployed.

The SSID, on the other hand, is selected by the network operator. Typical SSID values are derived from the names of businesses, university departments, coffee shops, commercial wireless operators, and fictional names chosen by home users. An AP broadcasts regularly (e.g. at 100 ms intervals) a beacon in which it advertises its SSIDs. The SSID is not globally unique and it only gives partial information about the identity of the network. But since SSIDs are human readable, this information is often immediately meaningful to a human observer without the need for extensive data collection or correlating observations from different times and locations. Hence, AP operators may feel that the SSID is sensitive information and that broadcasting it makes them vulnerable to unwanted attention.

For this reason, the AP operator can configure the AP in such a way that it does not broadcast its SSID in the beacon frames. Instead, clients have to probe the AP to find out whether it belongs to a specific network. In practice, a client keeps a list of known networks and when it arrives in the radio range of an AP, it sends a *Probe Request* message for every SSID in the list. The *hidden SSID* gives the AP a degree of privacy compared to the usual *public SSID*: the SSID will not be visible in the user interface of wireless clients that come into the range of the AP. The privacy protection is, however, very weak. A hacker can sniff the plaintext Probe Requests and detect which SSIDs in them lead to an association. Nevertheless, disabling the SSID is a widely recommended practice [14, 34, 37, 39] and many AP administrators heed the advice. This indicates that there probably is a legitimate need for some wireless networks to appear nameless to outsiders.

Unfortunately, disabling the SSID broadcast has the unintended consequence that the client stations go around broadcasting Probe Requests containing the SSID [15,32]. Depending on the implementation, the client will either probe for all known SSIDs (e.g., Windows XP pre-SP2) or only ones for which the probing has been manually enabled (e.g., Windows Vista). In any case, the privacy problem has not been solved but only shifted from the AP to the clients. Clients can be profiled and even identified based on the set of SSIDs which they probe [32].

There is another situation in which clients send Probe Requests without first hearing a beacon with the same

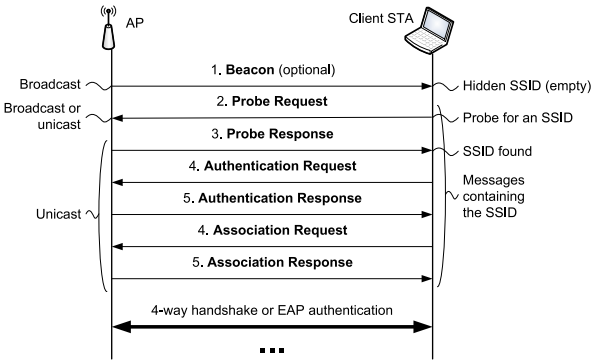

Figure 1: 802.11 network discovery and association with a hidden SSID or active probing (in active probing, the beacon is not sent)

SSID value. A client requiring low-latency connectivity, such as a WLAN phone during a VoIP call, may probe access points either proactively or when it observes a sudden drop in the signal quality from its current AP. It does this to speed up reassociation because scanning for the periodic beacons may cause 100 ms or more delay. This procedure is unlikely to cause major privacy issues if used only on the rare occasions when it is needed for application-layer quality of service (QoS). In practice, though, most 802.11 client implementations are not aware of the application-layer connection state and perform this *directed active probing* even when it is not strictly necessary. It would also be difficult to define heuristics for active probing that would accurately take into account both privacy constraints and the QoS needs of various applications. Some existing clients probe regularly for all known networks and, thus, leak their entire list of SSIDs to the network.

Figure 1 shows in detail the 802.11 network attachment procedure for an AP that has a hidden SSID. The AP sends a beacon with an empty SSID field. After observing the beacon, the client usually sends several Probe Request frames with different SSID values. If the AP recognizes the SSID, it returns the Probe Response frame. After that, the client can continue with the usual open authentication and association exchanges. The frames in these exchanges contain the plaintext SSID to identify the ESS to which the client wants to attach.

Some networks are *open* in the sense that the client can proceed with IP-layer communication immediately after this unauthenticated discovery and attachment procedure. In *closed* networks, the AP continues by initiating an authentication protocol based on either a pre-shared key (PSK) or the extensible authentication protocol (EAP), which involves an authentication server.

The goal of our work is to give both the APs and the client stations a level of privacy protection against

physical fingerprint of the radio transmitter
logical MAC-layer fingerprint (capabilities and parameters)
client MAC address, access point BSSID
SSID(s) in Beacon and Probe Response
willingness to associate with an SSID
SSID in authentication and association exchanges
TLS certificates in EAP-TLS
physical location of the clients and AP
association between clients and APs (implicitly associates APs with each other)

Table 1: Information leaks in 802.11

other nodes in the same physical location. We securely hide the SSID from passive and active attackers that are located on the access link. Our protocol gives equal protection both when clients connect to a network which has a hidden SSID and when the client performs directed active probing. The protection is particularly necessary when the client or AP uses randomized MAC addresses because the information leaked by SSIDs could defeat the purpose of the address randomization. We believe, however, that the greatest benefit will be to normal users with globally unique MAC addresses. Their anonymity will be protected against casual observers, such as other customers in a coffee shop, who do not perform global traffic analysis but may be interested in meaningful plaintext identifiers that are available on the spot.

Finally, if a client chooses to associate with a network, it is trivial for an adversary to infer that there is some kind of affinity between the two. Access points can be identified by their BSSIDs. There is not much reason for randomizing the BSSIDs because APs are stationary and could be identified based on the geographic location anyway. If clients use randomized MAC addresses and leak no other identifying information, the associations between the clients and APs give relatively little information to the adversary. On the other hand, if the client MAC addresses are permanent or if the adversary can otherwise recognize the individual clients, it can over time construct a bipartite graph of the associations between the clients and APs, from which it may be possible to infer further information on organization structures. Collecting and analyzing such information is extremely tedious, however, compared to the ease of sniffing plaintext SSIDs in Probe Requests. We summarize the information leaks in 802.11 in Table 1.

3 Design Requirements and Assumptions

The design is constrained by several functional and security requirements that we examine in detail in this section.

System requirements. First of all, to maximize the possibility that a system will be taken into use, it should be incrementally deployable. A networked system needs to interoperate with current networks. In particular in our case, access point operators need to be able to advertise an old network with unmodified beacons and at the same time send new beacons for the modified network. Similarly, the clients in transition phase can send old Probe Requests for networks that do not support the new features and can also associate with available public networks. All this, and the privacy-preserving properties, should be enabled with minimal changes to the current protocols.

Mobile and wireless networking imposes constraints on the efficient use of resources. Access points and clients are usually CPU-constrained and clients are also energy-constrained. Therefore, we require sparing use of symmetric key primitives, and no additional messages transmitted by the client as part of discovery. Thus, in addition to requiring minimal changes to deployed WLAN software, the protocol needs to be lightweight, that is, only require few rounds, for the discovery to be fast.

User experience. The user interface for configuring wireless networks depends on the particular device or operating system. For some legacy operating systems, the device manufacturers provide their own configuration tools. The Universal Access Method (UAM), which is used by service providers to authenticate the user with a Web browser, varies from provider to provider. Nevertheless, many user interfaces have similar features. Typically, the user can select the name of the network to connect to from a list. The WLAN client can also be configured to automatically reconnect to a known network when the wireless interface is activated. Hidden networks require the user to initially type the name of the network to be able to attach to it.

We favored not changing the user experience at all: users can still choose the networks to connect to and configure; they see the name of the access point being used; the client device can automatically connect to networks using the new or old protocol. To access hidden networks, the client has to know the SSID and the key of the hidden network. The key and SSID can be bootstrapped using methods that are currently used, for example, a password, WiFi Protected Setup [38], or Network-in-a-Box [4].

Threat model and privacy assumptions. The privacy properties we need to provide are two-fold: First an adversary cannot infer the network name of the access point more easily than with today's "hidden networks"; second the adversary cannot learn or infer which networks are known to a client through the network discovery

protocol alone. We examine each of those in detail.

Current "hidden networks" guarantee that a rogue client, that does not know the name of the network, cannot learn it by interacting with the access point alone. On the other hand, an eavesdropper can observe an honest client connecting to the network and learn its SSID. The proposed discovery protocol needs to be robust against such an attack: an adversary observing the discovery protocol cannot learn the name of the network, but only a randomly generated temporary identifier. This identifier can be changed as frequently as the network operators wish, subject to some efficiency trade-offs. Thus, we aim to provide slightly stronger properties than current "hidden networks". This also ensures that two access points belonging to the same network, and having the name SSID, cannot be distinguished from two unrelated access points — which is useful for hiding whole networks, and their size.

The second goal of the attacker is to identify the networks to which a client is willing to connect. We must ensure that an adversary merely observing the discovery protocol cannot infer which access points or SSIDs the client recognizes. Naturally, we cannot reasonably prevent the attacker from observing the communication between a client and the access point to which it is finally attached. Yet, we should make it difficult for the adversary to profile and identify clients based on the networks they know; passively identify networks previously visited; or actively probe the client to learn the networks known.

We assume that the attacker may have previously been at the physical location of any access network, or may have access to it later, and tries to correlate any information it can obtain from the network with any data that it can obtain from the client station. Despite allowing the adversary to gather global information before and after the protocol execution, we exclude adversaries that are able to perform live relay attacks from the clients to remote honest networks. To some extent this limits the protection offered by our system against a global active adversary, but *only* in case the client decides to associate with the remote network. In such cases the association is anyhow leaked by the fact that there is further communication between the client and the access point, and no discovery protocol alone can protect against such inferences.

We are equally concerned about active and passive attacks. If the access point sends its SSID or another permanent identifier in a beacon message or the client probes for various networks by broadcasting their identifiers, these messages can be observed by a passive eavesdropper. It is also very easy to set up a dummy access point or to act as a client to real access points. Indeed, an active attacker has the advantage that it can initiate communication at a time of its own choosing rather than wait for communication between an honest client and access point to take place.

PSK	shared key between all APs and client stations in the ESS
Ka	authentication key derived from PSK, used to compute the response from the challenge
Ke	encryption key derived from PSK, used to encrypt R-SSID
N_{client}	the client nonce, acts as a challenge
N_{AP}	the AP nonce, used to make response values unpredictable
$PRF_K(\dots)$	a keyed pseudorandom number
R-SSID	random value used in the following messages instead of the real SSID

Table 2: Symbols used in the protocol description

4 Privacy-Preserving AP Discovery Protocol

In this section, we describe an access-point discovery protocol that does not reveal the SSID to outsiders. The protocol has been tightly integrated with the standard 802.11 network discovery protocol.

4.1 Shared-Key Group Identification

The protocol is a lightweight nonce-based shared-key group identification with two messages, which are piggybacked on the Probe Request and Response frames of the standard 802.11 network association protocol. The basic assumption of the protocol is that the APs and all client stations of a single wireless network (ESS) share a single pre-shared secret key PSK (or an SSID and a password for deriving one). Neither the AP nor the client broadcasts the SSID. Instead, the client uses a challenge-response protocol to recognize APs that know the pre-shared key and, thus, belong to the ESS. We only need two messages because the identification is unidirectional: the client authenticates the AP. If the two nodes do not share a key, the protocol fails and neither learns each other's SSIDs or any similar linkable identifier.

Figure 2 shows the challenge and response messages in the context of the 802.11 network attachment procedure. Table 2 summarizes the symbols used in the protocol.

Just as in standard 802.11, the protocol may start with the client listening to beacons, or by active probing, directly from the probe message. When the client receives a Probe Response from one or more APs, it compares them with its list of known wireless networks. In standard 802.11, it does this by comparing the SSID strings in the Probe Response and in the known list. In our protocol, the client computes a pseudorandom function (PRF) value and compares it with the PRF value in the response.

The PRF values will only match if the client and access point are configured with the same secret key PSK. If the client finds a match, it may continue with the open authentication and association exchanges. Alternatively, it may cache the information that a network is available and take no further action. If there is no match, the protocol stops there. This behavior is similar to the standard SSID probing.

In the following, we describe the contents of each message and then explain the cryptographic details:

1. Beacon. The AP advertises its presence by sending periodic beacons. The SSID field in the beacon frame is empty. This frame is identical to the standard protocol with a hidden SSID. The AP should also advertise the fact that it supports the new protocol.

2. Probe Request. The client uses the Probe Request to send a challenge to one or more access points. It does this either after receiving a beacon or, in active probing, on its own initiative. The client includes its nonce N_{client} in the request. The SSID field of the Probe Request is empty.

3. Probe Response. The AP copies the client nonce N_{client} to the response and includes its own nonce N_{AP}. It selects a random identifier R-SSID and encrypts it with the shared key. Finally, it signs the message by computing a keyed pseudorandom function of the contents and the shared key. The full message is the following:

$$N_{client},\ N_{AP},\ E_{Ke}(\text{R-SSID}),$$
$$PRF_{Ka}(N_{client},\ N_{AP},\ E_{Ke}(\text{R-SSID})).$$

4–5. Open Authentication Request and Response. A pair of dummy messages required by the 802.11 standard for backward compatibility. The SSID field in these messages contains the R-SSID.

6–7. Association Request and Response. A pair of messages that moves the AP and client station to the associated state. The SSID field in these messages contains the R-SSID.

8. 4-way handshake. The standard 4-way handshake based on the pre-shared key defined in WPA2 and 802.11-2007 for strong mutual authentication and session-key generation.

The AP and clients are usually configured with an SSID and a password. From these, they compute a pre-shared key (PSK) in the way recommended by the 802.11 standard:

$$\text{PSK} = \text{PBKDF2(Password, SSID, SSID length, 4096, 256)}$$

Including the SSID in the computation helps to avoid collisions between keys when two networks accidentally choose the same password. The SSID also acts as salt,

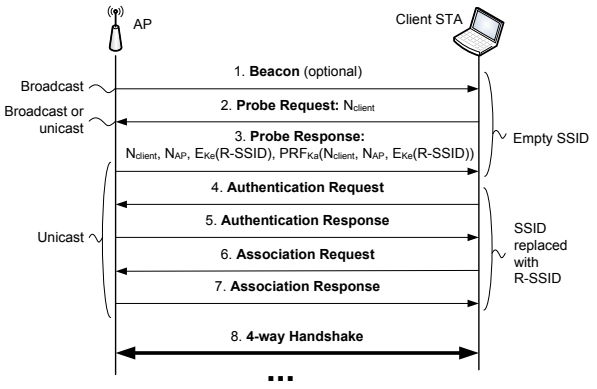

Figure 2: Privacy-preserving access-point discovery protocol

making password guessing mode difficult. The PBKDF2 function is defined in the PKCS standards [23] and it involves 4096 iterations of the SHA-1 hash function to make brute-force attacks that much slower. From the PSK, we compute two keys, one for authentication and one for encryption:

$$Ka = PRF_{PSK}(\text{"privacy key 1"} \mid N_{client} \mid N_{AP})$$
$$Ke = PRF_{PSK}(\text{"privacy key 2"} \mid N_{client} \mid N_{AP})$$

The keyed pseudorandom function $PRF_{PSK}(...)$ for the key derivation can be implemented as a standard one-way hash function, such as SHA-1, on the key and the input.

The nonces are 128-bit random or unpredictable pseudorandom values. The client nonce N_{client} is used for freshness of the network identification. The client should replace the nonce after receiving correct response, although it can accept multiple responses to the same probe as long as they have different N_{AP} values. Clients that randomize their MAC addresses should also generate a new nonce when the address changes. This is necessary for unlinkability. In practice, it is probably easiest to use a new nonce for every Probe Request. The purpose of the AP nonce N_{AP}, on the other hand, is not freshness of authentication but to make the responses unlinkable. Without the server nonce, an active attacker could replay challenges and see if it gets the same response from different APs. For this reason, the server should generate a new nonce for each response.

The PRF for computing the Probe Response can be the same function as that used for key derivation. We might expect standards bodies to prefer a construction specifically designed for authentication, such as HMAC [24].

When the client receives the Probe Response, it will not only verify the response with one key but with all Ka values it knows. Successful verification of the PRF value means that the AP supports the given SSID; verification failure means it does not. The amount of computation remains reasonable because a typical client is configured

to connect to at most tens of networks and only some of them have a hidden SSID. An access point that uses our protocol for multiple SSIDs needs to also compute and send multiple Probe Responses, one for each SSID. The client only needs to verify one set of responses from each AP. After that, the client can cache the information about which SSIDs are linked to which BSSID, and which APs serve no recognized networks. A periodic cache flush is needed to prevent the use of the cache behavior to trace the client, and a manual recovery mechanism may be needed to flush the cache immediately in case the AP's set of SSIDs changes.

The R-SSID is a random value that replaces the SSID in all management frames after the Probe Request and Response. Its role is to identify the ESS between the client and the access point. The AP generates a new random R-SSID for each Probe Response and caches the mapping between it and the confidential SSID for 60 seconds. (Cryptographic mechanisms can be applied to implement this in a stateless way but that is not essential.For example, the AP could calculate

$$\text{R-SSID} = \text{PRF}_{\text{Kap}}(\text{"R-SSID generation"} \mid \text{time} \mid \text{SSID} \mid \text{client MAC address})$$

where Kap is a key known only by the AP and time changes every 60 seconds.) If the client continues to associate to the AP using the R-SSID, the value will be stored as long as the client is in the authenticated or associated state. The main reason for the random R-SSID is to prevent observers from linking APs of the same ESS to each other. A secondary reason is to prevent the observers from counting the number of different ESSs served by the same AP, and from knowing which clients of the same AP belong to the same ESSs. The encryption of the R-SSID in the Probe Response provides rudimentary access control: only authorized clients are able to decrypt the fresh R-SSID. However, this is not a strong access-control mechanism, because the attacker could sniff an R-SSID from the Authentication Request of an honest client. Strong access control is provided by the following EAP authentication or 4-way handshake. An access point that belongs only to a single ESS and will execute the 4-way handshake could, optionally, use a fixed R-SSID value and send it in the Probe Response without encryption.

5 Evaluation

We evaluated the proposed protocol in several ways. It was implemented on existing 802.11 MadWifi drivers to verify that the implementation cost is as low as expected. The performance impact of the protocol was also measured in the ORBIT [30] wireless testbed developed and operated by the WINLAB, Rutgers University. We also ran the access point implementation on a small indoor access point, the Meraki Mini [27]. Moreover, we provide informal and formal security analysis of the protocol.

5.1 Comparison of the communications cost

We compare our protocol against both undirected probing of non-hidden networks and against directed probing of legacy hidden networks. In undirected active probing, the client sends one Probe Request on each radio channel and listens for responses. Every AP in the area will respond to the probing, but the number of APs in the same location is usually limited. In directed active probing, the client has to probe for each known SSID on each radio channel. We were surprised to find out that many implementations scan for them one SSID at a time. The implementations send a directed probe for an SSID on one radio channel, listen for responses, and move to the next channel. The time taken by this kind of scanning will increase linearly with the number of SSIDs that the client is looking for. In comparison, our protocol discovers hidden networks with performance equal to undirected active probing. We only send one Probe Request per radio channel and get a response from each AP in the area. Therefore, our protocol improves both the security and, assuming the client is looking for multiple SSIDs, performance of hidden-network discovery. The communications cost of our protocol is equal to that of standard undirected probing. The above comparison, of course, does not take into account the cryptographic overhead, which will be discussed next and shown to be negligible.

5.2 Implementation and performance measurements

The protocol implementation uses an Information Element (IE) field, a standard element defined in the 802.11 specifications that can be used for vendor-specific information, to carry our discovery protocol: the nonce is added to the Probe Request as an IE, while the AP uses an IE for conveying the nonces, encrypted R-SSID and the HMAC of the nonces and the encrypted R-SSID.

ORBIT measurements. The radio nodes in the ORBIT testbed are PCs with a 1 GHz VIA C3 processor, 512 MB RAM and wireless cards using Atheros AR5212 chipset.

We ran the tests in the grid using a single radio node as an AP, while 100 clients probed the AP. The clients continuously sent Probe Requests with broadcast SSID every 125 ms, that is, eight requests per second for a 10 minutes period. Figure 3 shows the probe processing times for the reference unmodified MadWifi implementation (solid line) and for our implementation (dotted line).

We also used mpstat to measure the CPU load for the AP. The CPU user and system load for both cases were under 1 %, and most of the time the tool showed 0 % usage.

Meraki Mini microbenchmarks The Meraki Mini access point uses an Atheros AR2315 System-on-a-Chip design clocked at 180 MHz. We chose OpenWrt [29] as

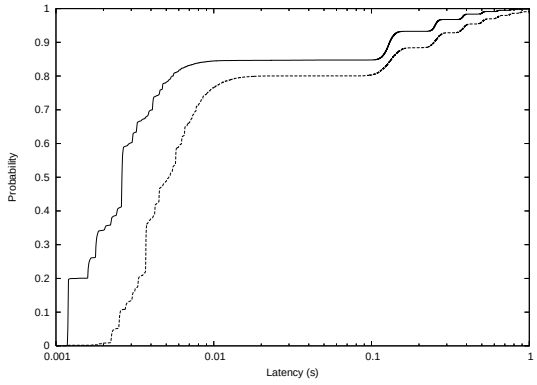

Figure 3: Probe processing times for standard protocol (broadcast SSID) (solid line) and for our protocol (dotted line)

our Linux distribution for the access point.

We probed the access point with a single client 1000 times with request rate interval of 200 ms. We obtained the following results for request-response pairs:

- legacy WiFi: average 1.8 ms latency and median 1.5 ms
- our protocol: average 3.2 ms latency and median 3.1 ms

To see how much of the reduction in performance was caused by the increase in packet size, we measured the effect of just padding the payload with 'A' characters to the size needed for our protocol. This gave a 2.8 ms latency with 2.1 ms median.

Additionally, we measured the raw processing times for creating and verifying the packets used in the protocol. We implemented a command line tool that was used to build the packets. 10 000 Probe Response packets were created in 53 167 ms and the *validation* of 100 000 Probe Responses took 340 301 ms. Thus, on average

- Probe Response is created in 0.53 ms
- Probe Response are verified in 0.34 ms

on a low-end hardware such as Meraki Mini.

Summary of results The main differences between the standard protocol and our privacy-preserving protocol are the additional computation related to forming and verifying the discovery packets, and the increased payload.

The ORBIT testbed measurements results show that even with large-scale enterprise deployment with high client usage, our protocol would not introduce negative impact on the overall user experience. We note that the load used in the measurements for the single AP was more than a single AP needs to handle even in an enterprise setting.

The microbenchmarks show that the protocol is deployable to a very low-cost commercial off-the-shelf access point. The results also show that the cost of the cryptographic operations is almost negligible compared to increasing the packet length of the standard messages. In this case also, our protocol would not introduce negative impact on the overall user experience.

Finally, we note that modern access points, as well as low-end WLAN clients, increasingly often have cryptographic hash and symmetric encryption functions implemented on hardware for WPA or WPA2, and therefore, the cost of above operations can be further reduced.

5.3 Security and privacy properties of the protocol

The privacy properties of the protocol are as follows. If a passive or an active attacker sees two access points which the client tries to contact, it cannot tell if their PSKs are the same or different. Likewise, if the attacker sees two clients trying to discover an access point, it cannot tell if their PSK is the same or different. Thus, the encryption function does not reveal the identity of the parties. Also, the protocol establishes the goal that in the discovery, no SSIDs or other fixed identifiers are given to an unauthorized station.

With the introduction of our protocol, e.g., WPA or WPA2 provides the same security as before, we only improve privacy of the access point discovery, even though we share the PSK. The protocol is vulnerable to offline dictionary attacks if the shared secret is derived from a weak password. This is equivalent to WPA-PSK and most other password-based authentication protocols.

The protocol can be used to launch denial of service attacks against the clients and the access points. However, since the protocol is very lightweight, the attacker does not gain much more than with current WLAN protocols. Against access points, the attacker can send arbitrary Probe Requests which create unnecessary Probe Responses. Against clients, the attacker could introduce unnecessary verifications of the keyed pseudorandom function sent in the Probe Responses.

5.4 Formal Verification

We used the `proverif` tool [6] to formally model and verify the security properties of the protocol. The `proverif` input is shown in Appendix A. Two models are used, one for authentication and confidentiality, the other for privacy. In both models, there are an unlimited number of concurrent clients and access points. We prove the following properties:

Confidentiality. (`query attacker:RSSID`)
The attacker is not able to derive R-SSID, i.e. it remains

secret after the Probe Response. Of course, the R-SSID will be revealed if the client decides to use it and decides to associate with the AP.

Authentication. (`query ev:rx(m) => ev:tx(m)`) An `rx(SSID)` event will not occur at the client unless there was a corresponding `tx(SSID)` event at the access point. That is, the client will only store an R-SSID if the access point (and not the attacker) sent it.

Privacy. An access point using key `K1` is observationally equivalent to one using key `K2`. This is modeled using the `choice` keyword. The idea is that the attacker is unable to distinguish an access point for network 1 (using shared key `K1`) from an access point for network 2 (using shared key `K2`). Clients using different keys are also observationally equivalent, but we don't formally verify this property because it is trivially true: the message sent by the client does not involve the key.

The privacy properties fail to hold if the client acts on the result of the authentication in some externally visible way, for example by sending the Open Authentication Request to an authenticated access point. This is because it enables a "wormhole" attack: the attacker forwards messages between the client location and the access point, and looks to see if the protocol completes. An AP from a known network can be used as an oracle to identify clients that wish to connect to that network, and vice-versa. For this reason, we leave out the `rx(ssid)` event from the privacy model. If it is left in, `proverif` will find an attack.

6 System Aspects

This section discusses design considerations beyond the cryptographic discovery protocol presented in Section 4.

Key provisioning The protocol requires a shared key between the access points and clients that belong to the same ESS. Our goal was not to change the existing conventions for key provisioning. Exactly as in the standard WPA-PSK security protocol, the stations need to be configured with the SSID and password. These can, for example, be written on a whiteboard, or one can use WiFi Protected Setup [38], Network-in-a-Box [4], or more advanced credential provisioning mechanisms already deployed in managed enterprise networks. Even though the access point in our protocol does not explicitly send the SSID to the client, the client knows it and can display it as a network name in its user interface. Thus, the WAP-PSK user experience is preserved both during the first connection to a new network and on later reconnections.

The access point can also support different keying schemes depending on its processing power and privacy needs. The basic setting would be to have a single key for all users. Alternatively, each user could have his own key. The choice of the key distribution defines the anonymity set for the client. When more users share the same secret, and if the attacker compromises the key, from a single message it knows only that the user belongs to a group of users. However, if the keys are pairwise with client and the access point, compromising the key compromises the user completely, and the access point discovery messages could be then easily used to track the single user. The shared key has other ramifications, too. For example, is it reasonable to assume that the users of a conference belong to the same secret society, that is, should the conference visitors to be able to track each other?

Interoperability with public and legacy networks The same client can be configured to connect to three types of networks: *public*, legacy *hidden* and *securely-hidden* networks. The first two are the existing networks with broadcast and non-broadcast SSIDs, and the last type uses the protocol presented in this paper. An AP may also have multiple SSIDs that belong to any of these categories. The important rule is that a client that knows an SSID to belong to a securely-hidden network must ignore beacons that advertise the SSID. Otherwise, the attackers could send such beacons to detect clients that are willing to connect to them. Similarly, an AP that is configured to be a part of a securely-hidden network must not respond to legacy active probes or association attempts that contain the plaintext SSID.

For performance and reliability reasons, it is not a good idea to use secure probes for networks that do not require them. The only reason to enable secure and insecure probing for the same SSID would be transition: it allows gradual reconfiguration of clients and access points over a period of time, until every node operates in the more secure mode, after which the less secure mode should be disabled. This kind of gradual upgrading path is critical for the new protocol to be ever deployed and we have been careful not to prevent it in the protocol design.

One caveat of the approach is also that, if only some users start to use the approach, their presence is easier to spot than that of legacy WLAN users. However, this would be true for all non-steganographic privacy mechanisms. On the other hand, with our approach, the probes do not leak user-readable names and the affiliations.

Steps after Network Discovery Having carefully redesigned the network discovery to hide all identifiers, we have to worry about the same information leaking in the later stages of network attachment. Typically, the client and the AP will run an authenticated key-exchange protocol, such as EAP and the 802.11 4-way handshake, to establish a secure session. After that, any identifiers in higher-level protocols will be encrypted. The key exchange itself, however, may reveal the identity of the client or the server. The WPA-PSK authentication, which

consists of a 4-way handshake based on the shared key PSK, is safe in this respect.

Some EAP authentication methods, on the other hand, send the client and authentication-server credentials as plaintext. In EAP-TLS, the server certificate can be sniffed by adversaries on the local network, and the client certificate is protected in only the latest EAP-TLS specification [36]. PEAP [31] has the same problem, only the client is protected. Naturally, it rarely makes sense to use an SSID-hiding network-discovery protocol in connection with an EAP method that reveals the network identity. The existence of protocol does, however, encourage future designers of EAP authentication methods to consider identity protection as a design requirement. Also, we note that the clients would still benefit from our protocol in this case, too, since the attacker can see the client certificates only if it is present during the association to the network, that is, the certificates are not sent if the correct network is not found.

Randomized MAC addresses. Randomized MAC addresses have been proposed before [17, 21] to prevent location tracking possibilities of WLAN clients. Our protocol can be used together with such privacy-enhancing techniques. Indeed, it makes sense to combine the two mechanisms. In that case, the client should update the nonce in our protocol, which otherwise could be reused for a short period, whenever it changes it MAC address.

User interface design alternatives. We have presented a design that does not change the user experience of WLAN access. However, there are some interesting design alternatives that were considered in the process. The protocol could be modified in such a way that the user only configures the password and the SSID is discovered automatically. In this version, the AP sends the encrypted SSID in the Probe Response (in addition to the R-SSID). The SSID can then be displayed to the user by the client user interface. This would allow also updating of the SSID, which is not possible in current networks. Such design choices could, however, prove problematic. If two network administrators accidentally choose the same passwords, that could lead to unintentional discovery of the wrong network. Also, it is convenient for the network to have a name so that users can talk about it even before connecting to it.

7 Related Work

WLAN privacy has been considered in many works before: location privacy risks of using fixed MAC addresses [17, 18], preventing location tracking by modifying the power of transmissions [21], how users can be tracked with different implicit identifiers [15, 32] and how the client can be identified by the intervals between Probe Requests [11, 13]. Using pseudorandom and changing identifiers have been proposed for MAC addresses [17, 18, 21] or for the whole stack [2, 26].

The ISO/IEC standard 9798-4 [20] on two pass authentication using a cryptographic check function can be considered as the abstract base for the discovery protocol we present in this paper. The standard protocol does not consider privacy properties and (naturally) how to integrate it to a privacy-preserving system that we describe. Similar approaches have been, however, proposed for RFID (WSRE) [22] and Bluetooth [40] and the upcoming Bluetooth update Wibree [12]. The approaches share in common that the client needs to do a key search for every received hash, and additionally the Bluetooth [40] proposal requires remembering previous identifiers. In contrast, with the privacy improvement for the Nike+iPod sport kit [35] the client needs to do a check for every received message. However, even though these protocols resemble our simple discovery protocol, none of them are directly applicable to the presented problem in the 802.11 access point discovery.

There are similar generic approaches to the problem that we have tackled in this paper, such as secret sets [28], private authentication [1], SmokeScreen [10], secret handshakes [3, 5], short group signatures [7] or broadcast encryption with short ciphertexts [8]. However, since these approaches involve either public key cryptography [1, 7, 8, 28], exponentiation [3, 5] require a predefined set [28], rely on timestamps and synchronization [10], they are not directly applicable to the problem we have presented. Further, these mechanisms are unnecessarily complex to the presented problem, since the approaches cover e.g. more complex requirements for secret handshakes, and could not be tightly integrated to the standard 802.11 access point discovery protocol.

Finally, independent of our work, Tryst [16, 33] proposes to solve the confidential service discovery problem. However, the authors of Tryst rely on explicit pairing for bootstrapping the discovery, in order to record the time when the keys were shared. Accordingly, their approach requires considering clock skew. In contrast, our approach does not require changes to the user experience, because even in the bootstrapping of the protocol our solution is completely agnostic for sharing the secret, that is, the SSID and the network password can be written on a whiteboard or passed on a piece of paper, and we do not need to rely on the accuracy of hardware clocks at all. In conclusion, Tryst is a clean-slate replacement for 802.11 access-point discovery, whereas we investigated how privacy-preserving access-point discovery can be introduced to 802.11 with minimal changes to existing software.

8 Conclusions

We propose a privacy-preserving alternative to the standard WLAN access-point discovery to replace the current directed active probing mechanism. The proposed approach is easy to deploy because it requires only minor changes to existing standard protocols, and to client and access-point software. The same clients and access-points can support simultaneously the new and old protocol modes for different SSIDs. Also, the protocol design preserves the current user experience. In essence, we demonstrate that, by deploying a simple but carefully designed extension to the 802.11 MAC protocol, we can resolve problems of hidden networks, gain a considerable increase in privacy, and even hasten the hidden access-point discovery when probing for multiple networks.

Acknowledgments

The work was initiated from discussions with N. Asokan and Jan-Erik Ekberg about Wibree address privacy. We thank Richard Black, Ranveer Chandra, Jukka Manner, Antti Oulasvirta, Ken Rimey for their insightful comments on early presentations of this work, and Juha-Matti Tapio helping out with cross-compilation issues on Meraki/OpenWRT and measurement scripts. Finally, we are grateful to Marco Gruteser and Ivan Seskar for the possibility to use the ORBIT testbed.

9 References

[1] M. Abadi and C. Fournet. Private authentication. *Theor. Comput. Sci.*, 322(3):427–476, Sept. 2004.

[2] J. Arkko, P. Nikander, and M. Näslund. Enhancing Privacy with Shared Pseudo Random Sequences. In *Proc. of Security Protocols*, Cambridge, UK, Apr. 2005.

[3] G. Ateniese, M. Blanton, and J. Kirsch. Secret handshakes with dynamic and fuzzy matching. In *Proc. of NDSS '07*, Feb. 2007.

[4] D. Balfanz, G. Durfee, R. E. Grinter, D. Smetter, and P. Stewart. Network-in-a-Box: How to Set Up a Secure Wireless Network in Under a Minute. In *Proc. of USENIX Security*, May 2004.

[5] D. Balfanz, G. Durfee, N. Shankar, D. Smetters, J. Staddon, and H.-C. Wong. Secret handshakes from pairing-based key agreements. In *Proc. of IEEE Security and Privacy*, May 2003.

[6] B. Blanchet. Proverif: Cryptographic protocol verifier in the formal model. Available at: `http://www.proverif.ens.fr/`.

[7] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *Proc. of Crypto '04*, Aug. 2004.

[8] D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *Proc. of Crypto '05*, Aug. 2005.

[9] R. Chandra, P. Bahl, and P. Bahl. MultiNet: Connecting to Multiple IEEE 802.11 Networks Using a Single Wireless Card. In *Proc. of Infocom*, Mar. 2004.

[10] L. P. Cox, A. Dalton, and V. Marupadi. SmokeScreen: Flexible Privacy Controls for Presence-Sharing. In *Proc. of MobiSys '07*, June 2007.

[11] L. C. C. Desmond, C. C. Yuan, T. C. Pheng, and R. S. Lee. Identifying unique devices through wireless fingerprinting. In *Proc. of WiSec*, March/April 2008.

[12] J.-E. Ekberg. Implementing Wibree Address Privacy. 1st International Workshop on Security for Spontaneous Interaction, 2007.

[13] J. Franklin, D. McCoy, P. Tabriz, V. Neagoe, J. V. Randwyk, and D. Sicker. Passive Data Link Layer 802.11 Wireless Device Driver Fingerprinting. In *Proc. of USENIX Security*, July/August 2006.

[14] J. Geier. *Wireless Networks first-step*. Cisco Press, Aug. 2004.

[15] B. Greenstein, R. Gummadi, J. Pang, M. Y. Chen, T. Kohno, S. Seshan, and D. Wetherall. Can Ferris Bueller Still Have His Day Off? Protecting Privacy in an Era of Wireless Devices. In *Proc. of HotOS XI*, May 2007.

[16] B. Greenstein, D. McCoy, J. Pang, T. Kohno, S. Seshan, and D. Wetherall. Improving wireless privacy with an identifier-free link layer protocol. In *Proc. of MobiSys '08*, June 2008.

[17] M. Gruteser and D. Grunwald. Enhancing location privacy in wireless LAN through disposable interface identifiers: A quantitative analysis. In *Proc. of ACM WMASH*, Sept. 2003.

[18] M. Gruteser and D. Grunwald. Enhancing location privacy in wireless LAN through disposable interface identifiers: a quantitative analysis. *Mob. Netw. Appl.*, 10(3):315–325, 2005.

[19] IEEE Standard for Information technology — Telecommunications and information exchange between systems — Local and metropolitan area networks — Specific requirements — Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE Std 802.11-2007 (Revision of IEEE Std 802.11-1999), June 2007.

[20] ISO/IEC. Information technology — Security techniques — Entity authentication — Part 4: Mechanisms using a cryptographic check function, 1999. Reference number ISO/IEC 9798-4:1999(E).

[21] T. Jiang, H. J. Wang, and Y.-C. Hu. Location privacy in wireless networks. In *Proc. of MobiSys '07*, June 2007.

[22] A. Juels. RFID security and privacy: a research survey. *IEEE JSAC*, Feb. 2006.

[23] B. Kalinski. RFC 2898: PKCS #5: Password-Based Cryptography Specification Version 2.0, Sept. 2000.

[24] H. Krawczyk, M. Bellare, and R. Canetti. RFC 2104: HMAC: Keyed-Hashing for Message Authentication, Feb. 1997.

[25] J. Lindqvist, T. Aura, G. Danezis, T. Koponen, A. Myllyniemi, J. Mäki, and M. Roe. Privacy-preserving 802.11 access-point discovery. In *Proc. of ACM WiSec'09*, Mar. 2009.

[26] J. Lindqvist and L. Takkinen. Privacy management for secure mobility. In *ACM Workshop on Privacy in the Electronic Society (WPES)*, Oct. 2006.

[27] Meraki Inc. Meraki mini specification. http://www.meraki.com.

[28] R. Molva and G. Tsudik. Secret sets and applications. *Information Processing Letters*, 65, 1998.

[29] OpenWrt. http://openwrt.org/.

[30] ORBIT. Wireless testbed. http://www.orbit-lab.org/.

[31] A. Palekar, D. Simon, J. Salowey, H. Zhou, G. Zorn, and S. Josefsson. Protected EAP Protocol (PEAP) Version 2, Oct. 2004. Internet-Draft. Expired.

[32] J. Pang, B. Greenstein, R. Gummadi, S. Seshan, and D. Wetherall. 802.11 user fingerprinting. In *MobiCom'07*, Sept. 2007.

[33] J. Pang, B. Greenstein, D. McCoy, S. Seshan, and D. Wetherall. Tryst: The Case for Confidential Service Discovery. In *Proc. of HotNets-VI*, Nov. 2007.

[34] J. W. Rittinghouse and J. F. Ransome. *Wireless Operational Security*. Digital Press, Mar. 2004.

[35] T. S. Saponas, J. Lester, C. Hartung, S. Agarwal, and T. Kohno. Devices That Tell On You: Privacy Trends in Consumer Ubiquitous Computing. In *Proc. of USENIX Security*, Aug. 2007.

[36] D. Stanley, J. Walker, and B. Aboba. RFC 4017: Extensible Authentication Protocol (EAP) Method Requirements for Wireless LANs, Mar. 2005.

[37] R. Stanley. *Managing Risk in a Wireless Environment: Security, Audit and Control Issues*. Information Systems Audit and Control Association, 2005.

[38] Wifi Alliance. Wi-fi protected setup specification, version 1.0h, Dec. 2006.

[39] E. Wilding. *Information Risk And Security: Preventing And Investigating Workplace Computer Crime*. Gower Publishing, 2006.

[40] F.-L. Wong and F. Stajano. Location Privacy in Bluetooth. In *Proc. of ESAS '05*, July 2005.

APPENDIX

A Formal models

A.1 Authentication and confidentiality

```
free c.

(* Symmetric key cryptography *)

fun enc/3.
reduc dec(enc(m, k, iv), k) = m.

fun prf/4.

(* First security property:
   the R-SSID is authenticated. *)

query ev:rx(m) ==> ev:tx(m).

(* Second security property:
   the attacker does not learn the R-SSID *)

query attacker: SSID_a.

let Client =
  new N1_c;
  out (c, N1_c);
  in (c, (N2_c, Cipher_c, Mac_c));
  let SSID_c = dec(Cipher_c, K) in
  if Mac_c = prf(N1_c, N2_c, Cipher_c, K) then
  event rx(SSID_c).

let AccessPoint =
  new SSID_a;
  in (c, N1_a);
  new N2_a;
  new iv;
  let Cipher_a = enc(SSID_a, K, iv) in
  event tx(SSID_a);
  out (c, N1_a);
  out (c, (N2_a, Cipher_a,
      prf(N1_a, N2_a, Cipher_a, K))).

process
  new K;
  (!Client | !AccessPoint)
```

A.2 Privacy

```
free c.

(* Symmetric key cryptography *)

fun enc/3.
reduc dec(enc(m, k, iv), k) = m.

fun prf/4.

let Client =
  new N1_c;
  out (c, N1_c);
  in (c, (N2_c, Cipher_c, Mac_c));
  let SSID_c = dec(Cipher_c, K) in
  if Mac_c = prf(N1_c, N2_c, Cipher_c, K) then
  event rx(SSID_c).

(* Third security property:
   a client using K1 is observationally
   equivalent to one using K2. Trivially true. *)

let KnownAccessPoint =
  in (c, N1_a);
  new N2_a;
  new iv;
  let Cipher_a_1 = enc(SSID_a_1, K1, iv) in
  event tx(SSID_a_1);
  out (c, N1_a);
  out (c, (N2_a, Cipher_a_1,
    prf(N1_a, N2_a, Cipher_a_1, K1))).

let UnknownAccessPoint =
  in (c, N1_a);
  new N2_a;
  new iv;
  let Cipher_a_1 = enc(SSID_a_1, K1, iv) in
  let Cipher_a_2 = enc(SSID_a_2, K2, iv) in
  event tx(SSID_a_1);
  out (c, N1_a);
  out (c,
       choice[
         (N2_a, Cipher_a_1,
           prf(N1_a, N2_a, Cipher_a_1, K1)),
         (N2_a, Cipher_a_2,
           prf(N1_a, N2_a, Cipher_a_2, K2))
             ]).

 (* Fourth security propery:
    an access point using key K1 is
    observationally equivalent to one using K2. *)

process
  new K1;
  new K2;
  new SSID_a_1;
  new SSID_a_2;
  (!Client|!KnownAccessPoint|!UnknownAccessPoint)
```