

Mechanisms to Mitigate Wireless Privacy Threats

Jeffrey Pang
Carnegie Mellon University
jeffpang@cs.cmu.edu

ABSTRACT

The proliferation of mobile wireless devices enables or magnifies several privacy threats that traditional link layer confidentiality mechanisms, such as payload encryption, do not protect against: user tracking, profiling, and traffic analysis. For example, it is well known that the exposure of long-lived, unique device addresses can be used to track users over time. Although these addresses can easily be changed, more subtle features exposed in encrypted link layer traffic can be used to identify and profile users as well. These features, which we call *implicit identifiers*, include identifiers used for service discovery, characteristics that encryption does not obscure, and protocol information in unencrypted headers. We propose mechanisms to remove these features from wireless link layer protocols without the loss of important network functionality.

1 INTRODUCTION

The proliferation of mobile wireless devices, such as headsets, portable game stations, and mobile phones, raises new privacy concerns. In the context of network architecture, privacy has traditionally been taken to mean the confidentiality of messages, i.e., ensuring that message contents are encrypted when they traverse untrusted links or routers. However, this new class of devices enables or magnifies several privacy threats that traditional confidentiality mechanisms do not protect against: user tracking, profiling, and traffic analysis [4].

These new risks are primarily a consequence of increased user mobility and wireless communications. These devices communicate with each other in places where users have preconceived assumptions about privacy, such as in homes and in personal vehicles. With wireless communications, any nearby observer can intercept low-level identifiers in transmissions. This makes it trivial to locate users via the presence of their devices, and to follow them as they move from place to place.

To mask low-level device identifiers, researchers have proposed changing device addresses periodically [6, 8]. However, masking select fields leaves other fields exposed. Information such as identifiers used for service discovery, identifying characteristics that encryption does not obscure, and protocol information in unencrypted headers can also accurately fingerprint a device.

In previous work [11], we identified several such *implicit identifiers* exposed in 802.11 traffic and showed that they can accurately identify users. These identifiers are exposed for two main reasons: to enable service discovery and due to side-channels in encrypted traffic. In this abstract, we discuss mechanisms to remove these identifiers without sacrificing important functions of mobile wireless devices.

Service discovery. An important class of implicit identifiers are those exposed by discovery and rendezvous mechanisms, such as network/device discovery protocols in 802.11 and Bluetooth and local name resolution protocols (e.g., NetBIOS and multicast DNS). These mechanisms, which create bindings between different levels of names and addresses, often reveal identity information, either as a side-effect or by disclosing the binding, e.g., Bluetooth discovery advertises device identity, and 802.11 discovery often involves broadcast queries for familiar network names. Such mechanisms are an essential bootstrapping component for applications in wireless environments and thus can not be removed without loss of crucial network functionality.

We have designed and implemented an efficient discovery mechanism that enhances existing discovery and rendezvous protocols so that they are anonymous; that is, they do not expose sender or recipient identities to third parties. Because anonymous discovery protocols involve cryptographic keys, a central challenge in their deployment is automating key exchange between clients and services that may not have met before. We discuss our work in addressing this challenge in Section 2.

Side-channels. Another important class of implicit identifiers are those exposed by analyzing the implicit characteristics of encrypted messages, e.g., inter-arrival times and message sizes. In addition to identity, analysis of these implicit traffic characteristics can reveal other sensitive information, including passwords used [14], webpages visited [9], videos watched [13], languages spoken [17], and applications used [18].

These implicit characteristics are induced by applications that were not designed with them in mind. These channels are often subtle and go unnoticed until after applications are deployed. Thus, we argue that there should be a uniform interface to mask sensitive characteristics as they are discovered in the same way that anti-virus soft-



Figure 1: Two existing service discovery mechanisms.

ware and intrusion detection systems provide interfaces to prevent attacks on previously unknown (or simply unpatched) software vulnerabilities as they are discovered. Although these systems merely provide mechanisms to prevent known exploits and do not fix underlying vulnerabilities themselves, they have proven to be extremely useful in practice. No such mechanism yet exists to prevent known side-channel information leaks that threaten privacy.

We propose a rule-based system that selectively masks sensitive characteristics on wireless links while meeting application performance constraints. When sensitive side-channel characteristics are discovered (*e.g.*, packet sizes in some context), users can add rules that indicate that such characteristics should be masked (*e.g.*, ensure that packet sizes appear uniform). Based on specified performance constraints (*e.g.*, maximum jitter), our system will mask these characteristics in the best feasible way (*e.g.*, by adding packet padding). This proposal is discussed in Section 3.

2 PRIVATE SERVICE DISCOVERY

Previously, we presented the design and evaluation of *Tryst* [5, 12], a mechanism that enables a client to discover a service without exposing either client or service identities to third parties.¹

2.1 Challenges

Removing identifiers completely from the process of service discovery and binding (*e.g.*, link establishment) is hard for two reasons. First, wireless clients and services typically rendezvous by broadcasting an agreed upon identifier (Figure 1). A service might be willing to expose its identifier through announcements to save potential clients from having to expose it in probes. No such straightforward solution exists to conceal both the client’s and the service’s identity. A solution for this latter case is increasingly desired as personal devices often offer ad hoc wireless service themselves (*e.g.*, wireless game stations). Second, clients and services need to authenticate each other if they want to ensure that a trust relationship exists. However, cryptographic authenticity is difficult to provide without identifiers. Message recipients typically need to know which cryptographic key to

¹*tryst* (noun) - a secret rendezvous (especially between lovers).

use to verify a message, and it is hard to tell the recipient which one without explicitly identifying it.

2.2 Tryst

Private service discovery requires that discovery messages, such as probes, have the following security properties: third parties should not be able to link messages sent at different times to the same senders or intended recipients, message recipients should be able to verify their authenticity and integrity, and message contents should be confidential.

Theoretical protocols that have these properties have been proposed before [1], but they rely on public key cryptography. These protocols essentially encrypt entire packets, including the addresses of the recipients, with public keys. Therefore, in order to determine whether a packet is destined for a device, it must attempt to decrypt it, a computationally expensive operation. Symmetric key variants are also possible, but packets can not reveal which key should be used to decrypt the packet or else they would not be unlinkable. As a consequence, a receiver must try to decrypt a message with all its keys—one for each potential sender—before discarding it. Public and symmetric key protocols of this kind can incur so much overhead in practice that discovery and association fails in the face of background traffic [5].

Tryst overcomes these limitations by using temporary, unlinkable addresses. That is, suppose we wish that any two messages sent more than I time apart should not be linkable by third parties. If a sender and a intended recipient share a symmetric key k , which they exchanged at time T_0 , then at time $T_i = T_0 + i \cdot I$ (*i.e.*, the i th time interval of length I after key establishment) they can independently compute:

$$addr_i = AES_k(i), \quad \text{where } i = \lfloor (T - T_0)/I \rfloor$$

i.e., the encipherment of i . The address $addr_i$ can be used to deliver packets during time interval i , since both sender and receiver know the address, but addresses used during different time intervals can not be linked by third parties without the key k . Receivers simply compute a table of n addresses for the n potential senders that may send them packets, each time interval I . Unlike the public and symmetric key protocols discussed above, receivers can quickly determine whether a packet is destined for them by doing a table lookup.

Our evaluation of an 802.11 prototype that uses *Tryst* for discovery demonstrates that its overhead is comparable to that of WPA authentication [5]. For example, Figure 2 shows the mean time required for a client to setup an 802.11 link to an AP using different protocols. Using *Tryst*, the client actually performs faster than when using WPA-PSK (*wifi-wpa*) and achieves strictly stronger security. *Tryst* also scales as well as 802.11 without any

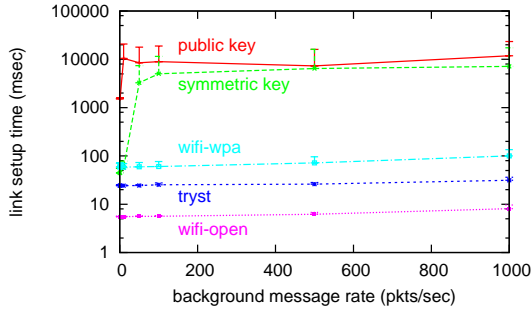


Figure 2: Link setup time for successful attempts as we vary the rate of background probe traffic not destined for the target AP. The target AP has 500 client accounts. Error bars indicate one standard deviation.

security (wifi-open) and scales much better than the public and symmetric key straw men discussed above.

2.3 Automating Trust Establishment

Tryst requires that clients and services have a pre-existing trust relationship in the form of shared symmetric keys. In some emerging uses of private service discovery, having to negotiate symmetric keys before discovering a service for the first time may hinder ease-of-use. For example, device pairing [15]—techniques used to establish keys on two personal devices that a user wants to connect (*e.g.*, Bluetooth peripherals)—typically assumes that users of the devices can identify them physically, which is may not be the case (*e.g.*, when trying to find an 802.11 AP). Moreover, these mechanisms assume that a client *already knows* the specific service it wants to discover. Service discovery is often useful because it enables users to find services they do not yet know about but still have reason to trust. Therefore, we are developing mechanisms that could enable these keys to be established automatically based on previously known trust relationships [12]. For example:

New Devices in Trusted Domains. This first mechanism enables a client to establish keys with a service for the first time with only knowledge of its “name.” By using a common naming convention clients can probe for devices belonging to users that they trust even if they do not know if those devices exist. For example, suppose devices are named using the convention owner-email/device-type. If Bob trusts Alice, he can enter Alice’s email address `alice@mac.com` into his iPod. His iPod can then privately probe for Alice’s iPod, which would be named `alice@mac.com/ipod`, if it existed. This mechanism leverages anonymous identity based encryption [2], which allows us to use these names as encryption keys. Although this mechanism uses public key cryptography, which is computationally expensive, it is only necessary the first time that two devices meet.

New Transitively Trusted Devices. A second mechanism enables devices to establish keys with any device transitively via a trusted friend who trusts it. This mechanism leverages a private social proximity test [3] to automatically establish keys with “friends-of-friends.” For example, this test enables Bob’s iPod to send an unlinkable message that can only be deciphered by devices that Alice has allowed to trust her friends, without having to explicitly tell Bob about any of them.

To evaluate the utility of transitive trust establishment mechanisms based on social networks, we plan on conducting a user study to evaluate the relationship between social networks and mobility and contact patterns.

3 MASKING SIDE-CHANNELS

To address the second class of implicit identifiers, we are developing a system that masks side-channel characteristics that act as implicit identifiers. This is a rule-based system that selectively masks sensitive characteristics of encrypted traffic at the link layer while meeting application performance constraints. For example, the identity of streaming videos can be revealed in encrypted traffic because packets are encoded using a variable bitrate [13]. Thus, a sequence of packet sizes can serve as a fingerprint for a particular video. The sequence of packet sizes over a time window is a side-channel.

A user may specify a rule to our system that this characteristic be masked, *i.e.*, that observable packet sizes be uniform. There are multiple ways packet sizes could be masked: by inserting extraneous cover traffic, by adding padding to packets, or by reorganizing the boundaries of packets in the packet stream. Each of these options has differing overheads and delay penalties. Our system intends to find an option that meets application specified constraints (*e.g.*, video playback typically has a maximum tolerable jitter).

In contrast to most previous work in traffic analysis prevention (*e.g.*, [7, 10, 16]), our goal is not to prevent all information leaks or even all leaks during a time interval, but rather to provide a framework to plug specific leaks when they are discovered, filling a niche similar to anti-virus software. This enables more efficient masking rules since only constant rate cover traffic can mask all information. Moreover, employing selective cover traffic only on wireless links, rather than end-to-end, is advantageous because these links often have more capacity relative to end-to-end paths and thus can tolerate some overhead.

3.1 Masking Mechanism

We now present a high level overview of our masking architecture and discuss challenges for its implementation. Applications or users specify *performance constraints*. The user or a “privacy vendor” (*e.g.*, like an

anti-virus signature vendor) specifies *masking rules* to protect against known side-channel attacks. Constraints and masking rules may be specified in conjunction.

An optimizer takes this specification and finds a set of outbound padding and delay settings that sufficiently mask the desired characteristics. The specification is also sent to the access point so that it can apply appropriate inbound padding and delay settings. In addition to this basic design, it is also possible to apply dynamic masking rules (*e.g.*, that only turn on when certain applications are running) and constraints (*e.g.*, constraints via per-packet annotations).

The primary challenge in realizing this architecture in practice is designing a masking rule language that is sufficiently expressive to capture all possible countermeasures to side-channel leaks. In addition, combinations of these countermeasures, which are necessary to mask multiple different side-channel leaks, should not typically degenerate to constant rate cover traffic. That is, if I have one rule that hides the movies I am streaming and another rule that hides the webpages I am browsing, applying the sum of those rules should not incur much more overhead than the sum of their overheads when applied in isolation.

We believe that we can meet this challenge because attacks on side-channels work at different time granularities: some identify aggregate characteristics of traffic samples that are tens of minutes long [13], while others identify more precise packet patterns in samples of less than one second [9]. Longer time scales are required when too much noise exists at shorter time scales (*e.g.*, due to background traffic or application variation), while shorter time scales are required when samples over longer time scales are roughly independent. Under the assumption that noise at different time scales is independent, these rules can be employed independently.

3.2 Learning Masking Rules

So far we have proposed a general mechanism for preventing known side channel attacks, but that does not guarantee that different users are indistinguishable. Indeed, if different users apply different masking rules, the rules themselves may induce implicit identifiers that distinguish them. Therefore, we are also developing a method to learn the rules to use in different locations to prevent location tracking. This will thwart most tracking attacks because traffic will become *location* dependent rather than *user* dependent.

4 SUMMARY

Implicit identifiers pose significant privacy threats in the wireless era. We presented mechanisms to address two important classes of implicit identifiers exposed in wireless traffic: those exposed in service discovery protocols

and those exposed via side-channels in encrypted traffic. Our current work involves two primary directions: First, we are evaluating the utility of using social networks to automatically establish trust between devices for private service discovery. Second, we are developing a flexible masking language to “patch” sensitive side-channels. We believe our work is increasingly important as we become more dependent on wireless devices in our daily lives.

REFERENCES

- [1] M. Abadi and C. Fournet. Private authentication. *Theor. Comput. Sci.*, 322(3):427–476, 2004.
- [2] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. MaloneLee, G. Neven, P. Paillier, and H. Shi. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. In *CRYPTO*, 2005.
- [3] M. J. Freedman and A. Nicolosi. Efficient private techniques for verifying social proximity. In *IPTPS*, 2007.
- [4] 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 *HotOS XI*, 2007.
- [5] B. Greenstein, D. McCoy, J. Pang, T. Kohno, S. Seshan, and D. Wetherall. Improving wireless privacy with an identifier-free link layer protocol. In *Mobisys*, June 2008.
- [6] M. Gruteser and D. Grunwald. Enhancing location privacy in wireless LAN through disposable interface identifiers: A quantitative analysis. *ACM MONET*, 10, 2005.
- [7] Y. Guan, X. Fu, D. Xuan, P. U. Shenoy, R. Bettati, and W. Zhao. Netcamo: Camouflaging network traffic for qos-guaranteed mission critical applications. *IEEE Transactions on Systems, Man, and Cybernetics*, 31(4), July 2001.
- [8] T. Jiang, H. Wang, and Y.-C. Hu. Preserving location privacy in wireless LANs. In *MobiSys*, 2007.
- [9] M. Liberatore and B. N. Levine. Inferring the source of encrypted http connections. In *CCS*, Oct. 2006.
- [10] R. Newman-Wolfe and B. Venkatraman. Performance analysis of a method for high level prevention of traffic analysis. In *Computer Security Applications Conference*, Dec. 1992.
- [11] J. Pang, B. Greenstein, R. Gummadi, S. Seshan, and D. Wetherall. 802.11 user fingerprinting. In *MobiCom*, Sept. 2007.
- [12] J. Pang, B. Greenstein, D. McCoy, S. Seshan, and D. Wetherall. Tryst: The case for confidential service discovery. In *HotNets*, 2007.
- [13] S. T. Sponas, J. Lester, C. Hartung, S. Agarwal, and T. Kohno. Devices that tell on you: Privacy trends in consumer ubiquitous computing. In *Usenix Security*, pages 55–70.
- [14] D. X. Song, D. Wagner, and X. Tian. Timing analysis of keystrokes and timing attacks on ssh. In *USENIX Security*, 2001.
- [15] J. Suomalainen, J. Valkonen, and N. Asokan. Security associations in personal networks: A comparative analysis. Technical Report NRC-TR-2007-004, Nokia Research Center, Jan. 2007.
- [16] B. Timmerman. A security model for dynamic adaptive traffic masking. In *Workshop on New security paradigms*, 1997.
- [17] C. Wright, L. Ballard, F. Monrose, and G. Masson. Language Identification of Encrypted VoIP Traffic: Alejandra y Roberto or Alice and Bob? In *USENIX Security*, 2007.
- [18] C. Wright, F. Monrose, and G. Masson. On inferring application protocol behaviors in encrypted network traffic. *Journal of Machine Learning Research*, Aug. 2006.