THESIS TITLE: Providing Privacy from the Residential Cloud

AUTHOR: Aldo Cassola

Ph.D. Thesis Approved to complete all degree requirements for the Ph.D. Degree in Computer Science.

Thesis Advisor: [Signature]
Date: 4/21/2015

Thesis Reader: [Signature]
Date: 4/22/2015

Thesis Reader: [Signature]
Date: 4/21/15

April 21, 2015

GRADUATE SCHOOL APPROVAL:

Director, Graduate School: [Signature]
Date: April 22, 2015

COPY RECEIVED IN GRADUATE SCHOOL OFFICE:

Recipient's Signature: [Signature]
Date: 4/22/2015

Distribution: Once completed, this form should be scanned and attached to the front of the electronic dissertation document (page 1). An electronic version of the document can then be uploaded to the Northeastern University-UMI website.
Providing Privacy

from

the Residential Cloud

Aldo Cassola

April 23, 2015

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

to the

Faculty of the College
of Computer and Information Science
Northeastern University
Boston, Massachusetts
This document was typeset using \LaTeX, with classicthesis\footnote{http://code.google.com/p/classicthesis/} developed by André Miede in the style of Robert Bringhurst’s “The Elements of Typographic Style”, and trying to keep a similar look to Aaron Turon’s impressive Ph.D. thesis. The bibliography was processed by Biblatex\footnote{http://www.ctan.org/pkg/biblatex}.

Robert Slimbach’s Minion Pro is the main text typeface. Sans-serif text is typeset in Steve Mattenson’s Opensans, Bitstream Vera Mono by Jim Lyles for monospaced text, and Donald Knuth’s Computer Modern for other purposes throughout.

Providing Privacy from the Residential Cloud
© April 23, 2015, Aldo Cassola
Abstract

The growth of mobile devices and computing has continued in recent years to the point where mobile network providers must not only upgrade their networks to serve the new traffic-intensive content that their users demand, but to actually turn to alternative methods of delivery, namely Wi-Fi hotspots and femtocells. This demand is driven in part by the surge in streaming services, but also by the demand for ubiquitous access to data hosted in cloud services.

The increased connectivity has changed user expectations for access to their data. Cloud service providers have seen similar increases in their user bases as clients migrate from laptops and desktop computers to tablets and smartphones. The amounts of data and computation performed in cloud services has always been of interest to eavesdroppers, and their growth can only make it more valuable to them. In such a world, a growing dependency on centralized providers makes them single points of failures for privacy threats.

With residential broadband Internet becoming more commonplace, always-connected home devices can collaborate to build cloud services closer to the network edge to mitigate the threats, creating more diffuse targets of attack. In this work we propose to leverage these networks to provide improved privacy for network access control, and edge cloud storage with little to no administrative overhead to home network users. In particular our main aim is to provide IP location privacy, anonymous access control, and efficient network performance for residential edge services. We show our prototyping work implemented over our own testbed for residential devices that we will also use for implementation and evaluation, and describe the proposed work to solve the research problems emerging in this context.

In TrafficJam [34] we illustrate the vulnerability of what is considered the most secure implementation of Wi-Fi. We devise a novel attack using a multi-layered targeted approach that makes the attack hard to detect yet having long range and effectively invisible to the victim.
We present the Open Infrastructure system, a home-broadband research platform built on off-the-shelf components and open source software deployed over 30 Access Points through Boston, Houston and San Francisco urban areas. The data obtained over the course of the project since February 2011 confirms the ongoing trends in home broadband Internet access, and highlights the feasibility of providing network access over home installations.

Privacy in network access is illustrated in TracEdge. As Internet Service Providers rely on open access points deployed over large numbers of installations, most of them controlled by other ISP subscribers or merchants. With this service ubiquity comes the risk for leakage of location information of mobile users as they associate, dissociate and move through these public access points. With TracEdge, an ISP can protect its users location information when connecting to Wi-Fi with the novel use of Private Information Retrieval techniques over installations of tens of millions of users. We implement and evaluate the performance of the PIR as an EAP protocol extension, making it readily usable by providers and clients.
My greatest concern was what to call it.  
I thought of calling it ‘information,’  
but the word was overly used, so I decided to call it ‘uncertainty.’  
When I discussed it with John von Neumann, he had a better idea.  
Von Neumann told me, ‘You should call it entropy, for two reasons.  
In the first place your uncertainty function has been used in  
statistical mechanics under that name, so it already has a name.  
In the second place, and more important,  
no one really knows what entropy really is,  
so in a debate you will always have the advantage.’  

— Claude E. Shannon
Acknowledgements

There is probably nothing more important when embarking on the road of studying something deeply than having a good mentor. I was extremely lucky to have found a great one in my advisor, Professor Guevara Noubir. Without his ideas, support, and encouragement none of the pages of this work could have happened. I am forever grateful to him for being my guide on this journey. In a very similar manner, I could never have handled the end of the road without the members of my thesis committee: professors David Choffnes, Alan Mislove, and Omprakash Gnawali. Their patient reading and listening of the proposal of this work, its defense and its written incarnation have made these pages the best they can possibly be.

I also want to thank my coauthors during my stay at Northeastern University, professors Will Robertson, Engin Kirda, and Erik-Oliver Blass, doctors Bishal Thapa, and Tao Jin, as well as my friends and colleagues Deep Singh, Harsh Kumar, Kamal Sharma. Working with you has been an honor and a delight.

The support I have received from my friends both at Northeastern and outside through the years has inspired me both through their dedication and love for their work, as by their exceptional humor and kindness. Thank you Hooman Javaheri, Fangfei Zhou, Ana-Maria Vişan, Eric Miles, Dimitri Kanoulas, Triêt Võ Hủ'u, Amirali Sanatinia, Sevtap Duman, Asmita Mukherjee, Paola Malerba, Antonella Papa, Pedro Córdova, and Fernando Sánchez.

Much of this work is also based on the data gathered from the home networks of anonymous volunteers, friends and colleagues. Some of them may or may not already be mentioned above, but I cannot ignore their patience, and willingness to assist in this project by letting our devices sit in their homes running experiments and gathering statistics, and evidence. Without you the very ground truth undergirding this thesis is lost.
To my family, who have supported and cared for me from long before I even thought of embarking on this path. My parents Rafael Cassola and Ana María Loor, and my siblings Alejandro, Isabella and Julie, and all my extended family: you have shaped me, you have made me better, and I am so grateful to have you.

And to my wife, Kristen, without whose loving support and encouragement, day after day during good days and bad all these years, none of this would make sense. I don't know whether I will be able to support you as much as you did, but I will certainly give it the old college try. I love you.

Aldo Cassola
Boston
April 2015
## Contents

1 **INTRODUCTION** ............................................. 1  
1.1 The State and Trends of Wireless Network Access ............ 2  
1.1.1 Advent of Free Cloud Services .......................... 3  
1.1.2 Security and Privacy Concerns .......................... 3  
1.1.3 Tapping the Residential Space ......................... 5  
1.2 Focus of this Work .......................................... 6  
1.2.1 Threat Models ......................................... 8  
1.2.2 Residential Entry Points .............................. 8  
1.3 This Thesis .................................................. 10  
1.4 Providing Residential Data Delivery .......................... 10  
1.4.1 Goals and Services ...................................... 11  
1.4.2 Evaluation ............................................... 12  
1.5 The **Open Infrastructure** Testbed .......................... 13  
1.5.1 Testbed Hardware, Software, and Scope ................. 13  
1.5.2 Results Summary ...................................... 15  
1.6 Wireless Access ............................................. 16  
1.6.1 The State of Wireless Access Control .................... 16  
1.6.2 Goals and Services ...................................... 17  

2 **RELATED WORK** ........................................... 19  
2.1 Characterizing Home Broadband Usage ....................... 19  
2.2 Weaknesses in Wi-Fi ....................................... 20  
2.3 Location Privacy and Anonymous Authentication ............. 23  

3 **THE FEASIBILITY OF RESIDENTIAL-BASED ANONYMITY NETWORKS** ................................ 27  
3.1 Open Infrastructure Testbed ................................ 29  
3.1.1 System Infrastructure ................................... 29  
3.1.2 Features .................................................. 30  
3.2 Delivery Throughput in a Residential Setting ............... 31  
3.3 Uplink Management ........................................ 32  
3.3.1 Provisioning of home uplinks ........................... 33
5.2.2 Improving Detection of Misbehaving ISPs .... 78
5.2.3 Correctness and Privacy ..................... 78
5.3 Private Information Retrieval .................. 80
5.3.1 Overview ................................... 80
5.3.2 NTRU-PIR Multiple Row Selection ........... 82
5.3.3 NTRU Encryption ............................. 82
5.3.4 Parallel NTRU-PIR ......................... 83
5.4 System ........................................ 84
5.4.1 The EAP-TE method ........................ 84
5.4.2 The Supplicant ............................... 85
5.4.3 The Authentication Server .................... 86
5.4.4 Optimizations ................................. 86
5.5 Evaluation Results .............................. 88
5.5.1 Evaluation Parameters ........................ 88
5.5.2 Database Generation and Communication Complexity ................. 89
5.5.3 Response Computation ...................... 91
5.6 Summary ...................................... 92

6 CONCLUSION .................................... 97
6.1 Feasibility .................................... 97
6.2 Attacking 'Secure' Wi-Fi Installations ............ 98
6.3 Private Wi-Fi Authentication ...................... 99
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The rise of mobile devices and the demand for cloud services have changed the landscape of network access.</td>
</tr>
<tr>
<td>2</td>
<td>Network access and traffic for cloud services today does little to protect privacy.</td>
</tr>
<tr>
<td>3</td>
<td>Using residential devices to push services to the edge allows for more robust privacy protection.</td>
</tr>
<tr>
<td>4</td>
<td><strong>Open Infrastructure</strong> testbed devices: Buffalo WZR-HP-G300NH</td>
</tr>
<tr>
<td>5</td>
<td>Web-Based Open Infrastructure AP Management Portal</td>
</tr>
<tr>
<td>6</td>
<td>Testbed device and management console</td>
</tr>
<tr>
<td>7</td>
<td>Latency measurement issued from Open Infrastructure Testbed</td>
</tr>
<tr>
<td>8</td>
<td>Bandwidth Usage PDF for Downlink.</td>
</tr>
<tr>
<td>9</td>
<td>Bandwidth Usage PDF for Uplink.</td>
</tr>
<tr>
<td>10</td>
<td>tc-based traffic prioritization for <strong>Open Infrastructure</strong>. User packets are sent when available. Served data has small guarantee, but can borrow available bandwidth.</td>
</tr>
<tr>
<td>11</td>
<td>Downlink behavior with and without experimental traffic. No impact is apparent in the presence of competing uplink traffic.</td>
</tr>
<tr>
<td>12</td>
<td>Uplink observed with and without experimental traffic. The low priority class backs off as soon as normal traffic starts.</td>
</tr>
<tr>
<td>13</td>
<td>Testing packet spacing over uplink saturation</td>
</tr>
<tr>
<td>14</td>
<td>Probe spacing over increasingly loaded uplink</td>
</tr>
</tbody>
</table>
Figure 15 Mean uplink throughput for each experimental home router. Several devices stand out with over 6 Mbps. Uplink throttling can go as low as 1.5 Mbps.

Figure 16 Cumulative distribution of normalized uplink throughput. Most of the experimental traffic reached the limit set by the ISP, while the user uplink was low.

Figure 17 Wireless network list for Microsoft Windows-based operating systems. Here, there are two seemingly identical entries for the SSID neutrino.

Figure 18 Wireless network list for Mac OS X. Again, there are two seemingly identical entries for the SSID free-wifi.

Figure 19 In the attack, an attacker leverages reactive jamming techniques to coerce a victim client to authenticate to a rogue network that appears identical to a target wireless network. A hash of the victim’s authentication credentials is then captured. The hash is cracked using a high-performance password cracker (not shown). Once the plaintext password has been recovered, the attacker uses it to authenticate to the target network with the privileges of the victim client.

Figure 20 An example of reactive jamming, where an attacker conditionally jams a target based upon the contents of a wireless frame. In this example, the attacker detects that a target is issuing a Probe Request frame. Before the target finishes transmitting the frame, the attacker issues a Probe Response frame to jam the target. Note that because the attacker can conditionally apply this jamming technique, the attack can be precisely targeted at a particular client and type of transmission.
Figure 21  Overview of hardware used in the prototype implementation of the attack. From left: desktop, antenna, router, USRP2 software-defined radio, victim client, signal analyzer (for evaluation), and attenuators  

Figure 22  A demonstration of the reaction time of our jammer. On the left, the signal analyzer shows an uninterrupted $800\mu s$ frame transmitted by a wireless client. On the right, the signal analyzer shows a Probe Response frame jammed at $300\mu s$. The horizontal grid width denotes $200\mu s$ divisions  

Figure 23  GPU hashing speed as a function of threads per CUDA block for dictionary generation. In particular, we compare the relative efficiency of four different hardware implementations  

Figure 24  Jamming and association probability over 1,000 trials as a function of distance. Note that attackers are not limited to a maximum of 1,000 trials, and that each trial takes approximately one second to execute  

Figure 25  Experimental setup for jamming tests. The access point and jammer are co-located in this experiment, while the client distance from the AP and jammer is varied at 50m increments. The jamming and association success probabilities against the client are recorded over 1,000 trials at each distance  

Figure 26  Results of a self-assessment of familiarity with computers and wireless networking performed by participants in our user study. Users tended to view themselves as technically sophisticated with respect to computing in general, while familiarity with wireless networking exhibits a bimodal distribution. In the latter case, most viewed themselves as sophisticated, while a smaller group classified themselves as less competent
| Figure 27 | SNEAP overview .................................. 68 |
| Figure 28 | SNEAP Architecture and Flow .................... 70 |
| Figure 29 | 1. supplicant 2. Client Authentication Portal 3. AP Registration Facebook app ................. 70 |
| Figure 30 | Private Information Retrieval overview .......... 80 |
| Figure 31 | Architecture of the EAP-TE protocol ............... 85 |
| Figure 32 | Throughput observed over ISP public hotspots in urban area .................................. 90 |
| Figure 33 | CUDA parameter impact in PIR computation time ........................................ 92 |
### List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1</td>
<td>Maximum TCP Throughput with 64 KB TCP Buffer Size on a Lossless Path</td>
</tr>
<tr>
<td>Table 2</td>
<td>Uplink Experiment Data Summary</td>
</tr>
<tr>
<td>Table 3</td>
<td>Total Uplink Throughput during the experiment</td>
</tr>
<tr>
<td>Table 4</td>
<td>Password search time, ordered by elapsed time</td>
</tr>
<tr>
<td>Table 5</td>
<td>A breakdown of the cost for each component of our attack prototype</td>
</tr>
<tr>
<td>Table 6</td>
<td>Identity Leakage vs. Misbehaving Server Detection</td>
</tr>
<tr>
<td>Table 7</td>
<td>Implementation choices</td>
</tr>
<tr>
<td>Table 8</td>
<td>Choice of $q$</td>
</tr>
<tr>
<td>Table 9</td>
<td>Evaluation systems</td>
</tr>
<tr>
<td>Table 10</td>
<td>Table creation time (s) ± $\sigma$</td>
</tr>
<tr>
<td>Table 11</td>
<td>Query generation on client</td>
</tr>
<tr>
<td>Table 12</td>
<td>Query extraction on client</td>
</tr>
<tr>
<td>Table 13</td>
<td>Unoptimized Polynomial Operations with degree $N = 439$ (times in $\mu$s)</td>
</tr>
</tbody>
</table>
Introduction

The mobile arena has seen trends of sustained growth through recent years. Smartphone manufacturers push their products to accommodate growing demand for always-connected wireless devices. Mobile network operators upgrade their infrastructure to support the increasing amounts of traffic, and even offload some of such traffic to Wi-Fi hotspots and femtocells. These trends in connectivity have driven the development of cloud services, making computing resources like storage and processing power a commodity. With them, mobile users can access data stored in the Cloud around the world, and can share it with ease, often at little to no cost.

As the reliance on cloud services deepens, so do security and privacy concerns. Wireless network access offers little privacy protection. Mobile networks allow the operator full access to handset traffic as well as location data. Wi-Fi access control can be performed in plain text with captive portals or through standard mechanisms over encrypted channels, but both strategies are susceptible to network impersonation, and always reveal the identity of the user to the authenticator. Cloud services, on the other hand have been known to tap into user-stored information to increase their revenue, and access to their offerings can leak information about user location or data access patterns to adversaries.

While growth in the user base can push cloud providers to protect entrusted data—most services encrypt traffic between the client and the service or even offer client-side encryption for data—the privacy landscape in wireless and cloud network access seems to take privacy as an afterthought.
Figure 1: The rise of mobile devices and the demand for cloud services have changed the landscape of network access.

Users increasingly rely on mobile devices for network access. As smartphones keep up with the latest generations of mobile network systems and Wi-Fi connectivity, people expect to be online at all times, especially in urban areas. The increase in user demand has prompted providers to manage their capacity by discontinuing unlimited plans and imposing caps on clients’ data usage enforced by additional service fees or reduced network performance. The growth in demand also prompts providers to upgrade their systems to faster and more efficient network protocols, and increasing channel usage per cell, forcing cell size to decrease. Today, subscribers can install small femtocell stations in their vicinity to improve their network performance. In other cases, strategically placed Wi-Fi hotspots serve mobile subscribers, effectively offloading cellular network capacity. But bringing base stations closer to subscribers is not easy, as this means significant investment on the provider’s end.

Wi-Fi network access in hotspots is normally managed over access points operating in Open Authentication mode, serving traffic in plaintext, and authenticating users over a captive portal running on over TLS. Other network access methods exist, such as WEP and WPA variants, that authenticate users with pre-shared keys or over the IEEE 802.1x standard. Using pre-shared keys for access control is most commonly
seen in the realm of home networks, where members of the household have configured their devices with the common keys. The use of 802.1x for network access is generally limited to more savvy users or professional settings, as the complexity in setup, overheads of account management, and differences in client interoperability make it impractical outside this context.

1.1.1 *Advent of Free Cloud Services*

Offerings such as Amazon’s EC2 make large CPU and GPU processing power available to anyone with an Internet connection and a credit card. Dropbox has penetrated not only the home—for which it offers free storage—but corporate culture with their always-available storage service, reaching over 200 million users [50, 89]. Even Google and Microsoft have jumped in the bandwagon by offering web-versions of their office suites and storage [78, 114] platforms for free, each with 120 and 250 million users respectively [30, 66].

Elastic services have contributed to the cloud service landscape. By offering computing, network and storage as a commodity, where customers only pay for actual usage, has made convenient for cloud providers to host their customer’s content on these third parties. This allows cloud providers avoid large investments on specialized hardware, in-house maintenance. In addition it is easier for cloud providers to react to spikes in service demand flexibly.

1.1.2 *Security and Privacy Concerns*

The architecture of mobile networks produces data that can be used to deduce user location using a variety of techniques. While its use normally occurs in the context of law enforcement, public concern regarding the technology has shifted due to recent revelation involving mass surveillance [68].

Cloud services often make use of established security techniques, protocols and algorithms to provide authentication, authorization, data transport confidentiality, and redundancy. Individual offerings however, typically only offer a subset of the services that make business
sense to the provider. For instance, free services like the ones offered by Google typically do not encrypt stored user data while their network exchanges are protected. GMail user messages, for instance, are known to be scanned in order to serve relevant ads to the user. Redundancy on the other hand, is normally under control of the same entity offering the service. Wuala, for instance offers client-side data encryption and redundancy, but only within its own network.

Generally IP address privacy is not considered in the design of cloud systems. Clients connect through browsers or specific mobile applications, which can and do reveal information about the user to the service provider. Location of a client can be approximated with only simple IP address database lookups, or client applications can report sensor data from the user’s device back to their server. Using anonymizing techniques is often not enough to protect against the above when application level data or even DNS lookups can leak IP addresses to adversaries.

Mobile network providers, on the other hand, have tried in recent years to reduce their network load by switching to more ubiquitous and faster to deploy Wi-Fi hotspots and femtocells. The growth of the mobile handset market has been the driver of such decisions, and its upward trend continues. The cellular space offers its own challenges in privacy, for users can be easily tracked using base station location information and identity is linked to a handset from the start. As things stand, the trend to offer open Wi-Fi access points for subscribers only adds to the existing privacy risks.

Figure 2: Network access and traffic for cloud services today does little to protect privacy.
Network providers such as AT&T and Comcast have in recent years deployed Wi-Fi access points across urban areas to shift load away from their networks while bringing subscribers closer to ubiquitous Internet access. Access is normally controlled through captive web portals to the provider’s network. The absence of lower layer authentication primitives results in clients broadcasting plain text traffic, impersonation, or even risking connections to Evil Twins. New initiatives for hotspot secure access are being developed by the Wi-Fi Alliance [164], which provides WPA-level protection to hotspot deployments in addition to handoff and roaming.

While technologies to counteract the risks in open Wi-Fi networks like WPA Personal and Enterprise have been in use for some time, their utility is limited. WPA-Personal provides encryption keys to any client that knows the shared passphrase, making it impractical for environments where clients are added and revoked periodically. WPA-Enterprise allows user management to happen behind the scenes, as clients authenticate securely through 802.1x with a central server in the network behind the Access Point before access is allowed. However, flaws in User Interface design, outdated protocols like MSCHAP-v2, and Public Key certificate impersonation make current deployments of WPA-Enterprise vulnerable to Evil Twin attacks.

The dependency on elastic services, such as those offered by Amazon, also raises privacy concerns. If user content is hosted on third-party devices, what guarantees exist over the privacy of the content? As the trends move in this direction, it is apparent that mechanisms to protect hosted data are needed.

1.1.3 Tapping the Residential Space

As broadband access to residential areas becomes ubiquitous, the number of home Internet subscribers has kept its upward trend globally, and with it, the deployment of home routers with ever greater hardware capabilities. Such growing pool of inexpensive, always-connected devices has the potential for becoming the basis for the privacy-aware and distributed applications that we need today.
Just as in services like TOR or FON [61], where users band together for a common goal, we believe that tapping into the growing number of small residential devices to provide network access and services can provide similar gains in the privacy space. Ideally, mobile network coverage can be served by home Wi-Fi devices whose owner has authorized use and may define payment amounts to himself without either mobile user or AP operator knowing their identities. The new class of edge cloud services would offer higher standards of protection and privacy without being tied to monolithic providers. Such services must be robust but also must operate within the constraints of limited and asymmetric network connections careful not to interfere with normal home usage. We cover our previous work on residential networks and the results of the Open Infrastructure testbed that confirm the feasibility of edge services on Section 1.5.

![Figure 3: Using residential devices to push services to the edge allows for more robust privacy protection.](image)

1.2 FOCUS OF THIS WORK

Leveraging large numbers of small residential sites to provide network services is a departure from the current privacy landscape of network and cloud access. For a privacy preserving system, infrastructure must be in place that allows usage of services with minimal leakage of user data. In the realm of network access wireless user may not want to reveal her identity to the owner of the access point, but he in turn may require payment or proof that she is authorized to access the network. Once connected, the user may request access to her documents in the cloud.
without revealing her identity to the provider, and protecting her data access patterns from potential eavesdroppers.

We address two aspects regarding privacy in the Edge Cloud: how to build a robust and easily-managed entry point, and implementing a well-behaved storage system. Because the potential number of services that can be offered by each component can be large, we focus in providing what we believe are a sufficient set of capabilities in each realm that can easily be generalized and extended. The following is the overview of services we aim to provide, and we define later in each section:

**Identity Protection:** Both clients and providers should not be able to deduce the identity of one another during operation. Needless to say, neither should potential eavesdroppers.

**Traffic Data Confidentiality:** No data should be disclosed to unauthorized parties.

**Availability:** For storage, data must be retrievable even in the presence of adversaries.

**Protection from Impersonation:** Clients should be able to detect provider impersonation attempts.

**Minimal Invasiveness:** No operations should disrupt the Access Point owner.

**Ease of Use:** Minimal assumptions can be made on the technical capabilities of residential Access Point owners and network clients. Management overheads and administrations task must be minimal and have a flat learning curve.

Throughout this work we use simplicity as a driving principle in the design of the above services. Any use case and interaction with the system should have well-defined steps that leave the system in a consistent step, minimizing surprises. A Network Authentication method that prompts a user for a password repeatedly after the credentials had been stored can be exploited to trick the user into providing its sensitive data. In general there is evidence that the decisions taken during service and interface design have an impact in the overall security of the systems. In the following sections we summarize both proposed frameworks, and discuss related work.
1.2.1 Threat Models

We mainly assume two slightly different threat models for this work on each section. In Chapter 4 we assume an active attacker that can create access points that impersonate the network, with a fixed power budget. This attacker can read and redirect traffic to the network it is impersonating, and more importantly it can trick the user into giving away its authentication credentials under a weak authentication protocol. We illustrate how to build such an attacker in WPA-Enterprise, the most widely deployed infrastructure that is largely considered secure. An adversary that is able to break such deployments is clearly strong enough to break non-encrypted Open Wi-Fi Hotspots as deployed in the wild.

As we explore the strategy to provide secure network access that is resilient to the attacker above, we introduce a new authentication method that also protects the authenticating user’s identity. In this case we extend the adversary to the authentication server itself. The main idea is for client authentication conversation to be indistinguishable among all clients, and the server computation to be opaque in terms of which entries in the user table is being accessed. This adversary is able to create an authentication table that flags each user to deanonymize the authentication.

1.2.2 Residential Entry Points

Building cloud services on top of residential devices implies that residential users become service providers, and thus take on administrative roles. For instance, a user may need to decide who has access to the service he offers in her home device. This authentication/authorization problem has typically been handled by the use of secret passwords (such as Pre-Shared Key in WPA for sharing the wireless network resource) or by the use of authentication servers, as in WPA-Enterprise.

Creating a local database of users and maintaining it is a task well beyond what a non-technical user may be willing to perform to share a service. Doing so would entail a learning curve for the tools she would need to use to implement it, and the maintenance tasks that come with it are error prone. Compounding this problem to the number of po-
tential residential users makes it clear that a naïve solution of keeping an account on every possible device is undesirable. Instead, the system should provide the capability for the owner to perform access control in the simplest manner possible, leveraging data and services that are already available.

Providing authorization and authentication for third-party services on the Internet is not a new notion. Services like Groupon, Slideshare, and even news sites use OAuth to permit end-users to access protected content on their service without the need to access user credentials, often in exchange for access to the end users’ social network posts and lists of friends. With the popularity of social networking sites like Facebook, their user base has made the site a global directory of billions open to be searched and crawled. Using social media services as authentication back-ends offers clear advantages over locally-maintained user tables: Setup and learning curve are virtually eliminated. Large bases of users are common in social media, and all the heavy-lifting is offloaded to the social media provider. Using a social networking site as a discovery mechanism for users seems a natural tool to use in providing authentication, the challenge is to protect user privacy and to avoid information leakage, given the issues in such networks. To do so, we design our services without placing any trust on the online-social network.

*Storage in the Residential Cloud*

By building a cloud storage service we seek to address the lack of privacy in current cloud storage offerings. The idea of distributed cloud storage is not new, as services like Wuala offered a variation of it in the past, but the goal of our work is to provide a similar system that minimizes user exposure to the service provider.

In addition, a distributed storage service can serve as a basis for Content Delivery Networks such as the ones used for video services, providing benefits to both end-users and ISPs. If data is accessible from within an ISP network users see less delay and throughput, and ISPs may see less traversal to external links to access content.
1.3 THIS THESIS

This work aims to illuminate design of location privacy-preserving network access control for Wi-Fi and edge cloud storage on residential broadband devices.

Our goal is to implementing both over a real-world testbed on actual residential installations. To this end we designed an implemented Open Infrastructure, the residential platform for home broadband routers that facilitates experiment deployment and data gathering.

The remainder of this work is organized as follows. We begin with Chapter 3 discussing the Open Infrastructure testbed, the residential deployment where we will implement our work. Chapter 4 shows multi-layer vulnerabilities of Wi-Fi deployments with stricter security protections than hotspots. Our improved Wi-Fi network access service that solves the above vulnerabilities and provides location privacy is discussed in Chapter 5. We discuss related work in Chapter 2. We conclude and sketch future work on Chapter 6.

1.4 PROVIDING RESIDENTIAL DATA DELIVERY

To reduce dependency on centralized storage systems, the edge cloud leverages the aggregate power of smaller always-on devices. Every individual device must be capable of not only serving content efficiently, and preserving privacy, but also with minimal impact to the normal residential traffic.

Building privacy into a system is not a straightforward procedure. For instance, naïve solutions to privacy in storage would add a confidentiality layer in the form of client encryption and anonymization through the use of onion routing to communicate with the cloud provider (e.g. Dropbox + EncFS + TOR, or https:// + TOR). Such solutions do not address leakage of information to the provider inside the encryption layer. Because the identity can be revealed to the provider by the client software—it may include IP address or location information in the process—a tighter set of privacy guarantees need to exists to protect privacy. In addition TOR performance, although increasing over time, is limited...
by the number of participating nodes and available node bandwidth, adding difficulty to the design of a good performance solution.

1.4.1 Goals and Services

Content Protection

Data Confidentiality Data in third party storage providers must not be accessible to others than the data owner or parties authorized by him.

Data Availability Once content has been committed to storage, it should be retrievable. We consider the case of storage providers becoming adversaries after data has been stored in their space (e.g. the provider holding data hostage, forcing payment not previously agreed to, modifying the data, cheating by removing data).

Access Pattern Protection Even encrypted data can be subject to information leakage. An adversary noticing that region of the data is accessed more frequently than others may imply it is more valuable to owner. Oblivious storage techniques exist to solve this problem and their performance is evolving, but we consider it out of the scope of this work. Current best techniques involve expensive mixing procedures requiring much more power than what we assume is available on residential devices. We cannot however dismiss the importance of having a system that protects access patterns, so we choose to modularize the storage strategy so SafEdge can be extended as better privacy mechanisms are developed.

Untraceability

We define traceability as the capacity of deducing either geographic location or identity of a data owner or storage provider after data exchange has occurred. While anonymity overlay networks like TOR can help protect against the former, the latter depends on the application itself. To be able to provide access control without revealing the identity of either storage provider or data owner to themselves or a third party is a challenge.
Convenience

**ubiquity** The current state and popularity of cloud storage services allow users to access and share their files no matter the network or devices used. **SafEdge** should provide at least as good access as current technologies, but because its capacity is directly related to the number of participants, special care must be taken to make it as user-friendly as possible.

**High throughput, low impact** Fast transfers are necessary, but we are limited by the residential uplink. A user willing to offer storage to others will be deterred from doing so if she notices participating impacts her normal browsing of video streaming.

**Low overhead** No additional administration tasks should be necessary from the data owner or storage provider further than what is normally required in other non-edge, centralized systems.

**Incentive mechanisms** User participation hinges on the perception of value on signing up to the system. Value does not need to be purely monetary; participation in FON [61] provides user the right to use other member’s devices for network access. Monetary compensation may still be possible through decentralized monetary systems like Bitcoin [117], but its lack of true anonymity would mean a redesign that is out of the scope of this project.

1.4.2 *Evaluation*

Previous work on available bandwidth problem [90] focuses on finding the minimum capacity link in a path and estimating its value in a time frame of seconds to minutes. However, we believe user experience expectations, change as their connections’ speed evolve, and decreasing estimates for user thresholds of delay acceptability [11] support this conclusion.

To evaluate feasibility residential data delivery we characterize the residential uplink using a long-term monitoring study. We examine usage asymmetry in the residential link, and the effect of latency on data throughput. In addition we experiment with congestion control perfor-
mance in each of the two possible deployment scenarios: direct access and indirect access to the last mile. In addition, we will measure the effect of data upload on background traffic under high load.

1.5 THE OPEN INFRASTRUCTURE TESTBED

Figure 4: Open Infrastructure testbed devices: Buffalo WZR-HP-G300NH

The Open Infrastructure testbed is a set of hardware and management tools designed to host new applications and experimental projects on a multitude of residential-grade Wi-Fi Access Points (APs). The testbed’s hardware deployment is comprised of off-the-shelf wireless routers running customized firmware, network monitoring and administration tools, and experiment management scripts. We first give an account of the testbed itself, its deployment and scale, followed by our preliminary results obtained from it.

1.5.1 Testbed Hardware, Software, and Scope

The devices in our deployment are Buffalo WZR-HP-G300NH [150] routers with 400MHz Atheros CPU, 64MB RAM, 32MB Flash, five 1Gbit ethernet ports, one USB port, and Atheros AR9132/AR9103 Wi-Fi Network Processor. Every device deployed has access to either a 16GB USB Flash or a 250GB hard disk, as shown in Figure 4. The devices run a modified version of the OpenWrt embedded Linux distribution [154] that includes a suit of management and experimental tools which we will describe below. OpenWrt supports a large hardware base with 27 architectures available, and over 100 well-supported router models [155].

We have deployed 30 customized APs in urban areas in Boston and Houston, serving around 100 individual users since February 2011. Our
user base is comprised of mainly graduate students and young professionals between 20 and 40 years of age, and of diverse backgrounds. We have plans of expanding the scale of our testbed to 150 nodes in the next year.

We monitor network usage through a custom heartbeat client program that aggregates data about the number of associated devices, average bandwidth grouped over different traffic types, or any configurable custom measurement. The data is aggregated over 10 second intervals and then sent over the network to our heartbeat server. All the reports are stored in our back-end MySQL server for further processing. Since the start of our deployment, we have obtained over 113 million records of data. In addition, the firmware includes a suite of SSH-based remote management tools to support remote firmware upgrades, update the AP configurations, schedule experiment tasks, etc. In order not to disturb our test users’ normal network usage, any and all data stored is scheduled for upload over off-peak hours such as midnight on weekdays.

We have also developed a web-based Testbed Management Portal as a front-end for administration purposes. See Figure 5 for screenshots.

![Stats Portal](image1)

(a) Stats Portal

![Access Point List and status](image2)

(b) Access Point List and status

Figure 5: Web-Based Open Infrastructure AP Management Portal
The testbed was built to provide first-hand information of urban Wi-Fi and residential network usage patterns over extended periods of time. Because urban homes network usage differs from other network deployments, like in academic and enterprise contexts, the **Open Infrastructure** testbed provides us with real-world data that is more granular in nature than deployments spanning ISP network segments. It also provides the flexibility to measure network characteristics from the edge of several network providers, and to potentially expand monitoring over diverse geographic areas.

### 1.5.2 Results Summary

Feasibility in crowdsourcing network access and edge storage services is directly related to the amount of backhaul capacity each device can contribute. To understand residential bandwidth usage, we observed network usage reported by the testbed between February and December 2014. From our results we find that during peak hours the probability that traffic is less than 10Kbps is greater than 65%.

We also find devices are capable of sustaining saturated uplinks without affecting the traffic of the owner, provided a sensible traffic control policy is in place. We set up deployed devices to initiate uplink UDP traffic simulating high background usage from the edge cloud. After a few seconds, a new UDP uplink transfer is started simulating traffic from the owner, such that both transfers compete for the backhaul. We then repeated a saturated uplink vs. owner upload.

Previous results in interconnectivity between devices through round-trip time (RTT) measurements among deployed devices, and through RTT measurements during wardriving show that latency between network subscribers of the same provider fall between 20 and 50ms at city level, which is enough to reach the provider’s uplink limit. Even with recent upgrades on the provider uplink, this result still holds.

The density of residential networks, unused backhaul, low RTTs, and high sustainable throughput support the feasibility for coordinated residential edge devices to provide network access and well-behaved edge storage.
1.6 Wireless Access

Protection on the wireless link layer constitutes the first line of protection against eavesdropping and impersonation. Current trends and issues in user privacy motivate us to propose improvements to Wi-Fi access control.

1.6.1 The State of Wireless Access Control

As network demand keeps its upward trends, mobile network operators try to offload traffic on their networks whenever possible. Several mechanisms for wireless LAN offload have been proposed in the standards [2, 5], and providers have implemented Wi-Fi offload in recent years. The 3G and 4G mobile networking standards also contemplate data protection mechanisms for air transmissions and end-to-end communication [3, 4].

The governing standard in Wi-Fi [87] includes mechanisms for authentication, data confidentiality and integrity. One of these mechanisms, colloquially referred to as WPA-Enterprise is the de-facto standard for medium to large network deployments requiring user authentication, and is based on the Extensible Authentication Protocol framework (EAP) defined in [86]. Even though standard Wi-Fi security techniques are routinely applied to home devices and business deployments, mobile operators who provide Wi-Fi hotspots do not normally implement them. Mobile users who wish to use Wi-Fi hotspots provided by their network operator commonly interact with captive portals over an SSL-enabled website and unencrypted link layer to authenticate to the network.

While the current traffic protection standards have held up better than the previous work [20], they are still vulnerable to certain types of attacks that exploit small multi-layer flaws in the implementations. With the introduction of attacks that leverage the lack of a trust relationship between a Wi-Fi SSID and the security certificate presented to clients [34], the need for a mechanism to provide a better Wi-Fi access control framework becomes apparent.
In this work we present a flexible authorization scheme for distributed services. We examine the minimal information necessary to perform authentication for the parties both in the context of the framework and of the implementation, and give a detailed account on its workings.

**Wi-Fi Privacy-preserving Authentication**

The process of authentication usually entails establishing the identity of a client and search on an authorization matrix to determine the services the user is allowed to access. Conversely, anonymous services protect the identity of users, but offer limited authorization capabilities.

We propose the use of a privacy-preserving authentication method that protects client and AP owner identity in the context of Wi-Fi network access. The challenge is to create a method that protects from impersonation without identity leaks over the limited capabilities of residential network devices.

Several cryptographic schemes can be used as building blocks for anonymous authentication. With group signature schemes, a set of users belonging to a group can sign messages and can convince a verifier that the signature was created by a member of the group without revealing its identity. Group signatures may include the notion of a group manager capable of deanonymizing a signature if necessary. Ring signatures, a simpler scheme do not require a manager and forego several registration an setup procedures associated with group signatures. Both of these schemes require signatures linear of size linear in the number of member of the anonymity set.

Private Information Retrieval (PIR) allows a client to retrieve bits from a database in a way that does not reveal the queried bits. A trivial PIR has the client transfer the entire database to its local storage, which then queries the database himself. However, PIR schemes exists with worst-case communication complexity $O(n^c)$ for $0 < c < 1$. A simple anonymous authentication scheme using PIR would work as follows: The AP uses key $K$ to allow authentication to the set of users $S = \{U_1, U_2, \ldots, U_n\}$, and has access to a database of $n$ entries $E_{pub_{U_1}}(K)$. If a discovery mechanism exists that allows verification of identities and
Public Keys, a user can request identity by sending a PIR request for the entry associated with its public key in the database. The server's reply returns the value to the user, along with a hash of $K$, to force the AP to commit to a value of $K$ for all its users. Once the user decrypts $K$, both client and AP can perform mutual authentication.

_Evaluation_

PIR schemes require both the database server and client to perform a series of computations to perform a query and to extract the requested value, respectively. We will examine current AP and client capabilities in performing our scheme. In addition to performance vs database size, user load, and comparison against current WPA schemes, we will also examine whether service of ongoing transfers to the home are affected by load.
2

Related Work

2.1 CHARACTERIZING HOME BROADBAND USAGE

In the past 2 to 3 years, the Internet traffic has been undergoing drastic changes. For example, the latest technical report from Sandvine Intelligent Broadband Networks shows that Netflix made up around 20% of Internet traffic during the prime time [101]. Other reports from Cisco's global IP traffic forecasting indicates that the global Internet video traffic surpassed global P2P traffic in 2010, and predicts that by 2012 Internet video will account for over 50 percent of consumer Internet traffic [139].

At the same time, the research community has shown a growing interest in profiling residential Internet traffic. A recent study analyzed a data trace collected from an ADSL ISP representing around 2 K subscribers, showing that HTTP dominates Internet traffic again, at the expense of the P2P traffic [129] due to Web Streaming. They also found in their dataset that P2P and Web Streaming are almost never used simultaneously. Earlier work in 2009, characterized the Internet traffic of 100 K US residential broadband subscribers [58], showing that HTTP traffic accounts for 68% of the total downstream traffic and that 34% of it is multimedia. It also showed that multimedia content over HTTP exhibits a 83% annual growth. An additional interesting finding is that CDN traffic is more efficient than P2P content delivery in terms of Air Miles. Similar findings indicating a major change of traffic breakdown were presented in [116, 144], as well.

More recently, some work focused on profiling residential traffic, primarily investigating the impact of FTTH deployment on the characteristics of residential traffic [27, 43]. Both papers show that FTTH net-
works show higher upstream volume thanks to the symmetric uplink and downlink capacity. Other recent work monitored the network traffic for more than 20K residential DSL subscribers in an urban area [106]. Their analysis reveals that HTTP traffic dominates by a significant margin, while P2P contributes only roughly 14%. They also observed from their RTT analysis that the latency from the DSL-connected host to its first Internet hop dominates the WAN path delay; and that the DSL lines are frequently not the bottleneck in bulk transfer performance.

Older related work, introduced a measurement methodology to conduct large scale measurement without requiring cooperation from the remote broadband hosts [54]. They carried out a rigorous characterization of residential broadband networks, indicating important differences between residential networks and academic networks.

There has been a consistent interest in literature examining the possibility of offloading distribution tasks to end users. Some of recent studies [108, 158, 160, 172] examine the feasibility and advantages of such system compared with current main stream consolidated storage services. Asides from providing the online service to end users, the Wi-Fi APs can also function as a Content Delivery Network. Because urban Wi-Fi APs are generally closer to the user and consume little energy, and thus have the potential of forming a P2P style CDN for better throughput to users, less operational cost, and less energy cost [44, 56, 85, 96, 108, 158–160, 170, 172].

2.2 WEAKNESSES IN WI-FI

Our attack against WPA Enterprise incorporates a combination of cross-layer vulnerabilities, from the physical layer to the human-computer interface. In this section, we place the attack in the context of related work.

Protection against impersonation and man-in-the-middle attacks in wireless networks has garnered interest in the networking community in recent years. Techniques include providing visual feedback to the user such as light or sound for secure pairing [77, 134], using correlated motion [35], using ambient signals around the transmitter to authenticate it [16, 74, 112], and special packet coding techniques to detect jam-
Anti-jamming systems have also been studied for decades, ranging from spread spectrum communication [145] to newer forms that allow key establishment for spread spectrum techniques in hostile environments [105]. While such techniques provide useful building blocks for securing wireless networks, they are limited in applicability for the existing WPA Enterprise standard and its deployment constraints.

Most evil twin attacks in the literature contemplate the impersonation of unsecured networks and do not include jammers, while our attack relies on a composition of cross-layer techniques to exploit weaknesses in UI design, authentication protocols, and the physical layer, achieving stealthy and targeted attacks. Work in the literature against evil twins fall into several categories, none of which is enough to prevent our attack. Secure device pairing [77, 112, 134] seeks to use properties in or around the devices that will be communicating to establish their identity. These approaches require line-of-sight access to the access point, or assume the attacker can’t co-locate with the device. Although our attack is capable of attacking long-range targets, it can also be placed close to an AP or client. Dedicated hardware could render the attack less conspicuous, and could allow for deployment with plug devices or smaller, making its physical presence hard to detect.

Protocols relying on trust-on-first-use assume that the first time a network is configured, the environment is secure. Our attack, however, exploits this assumption by creating a new network to trust as needed and, as such, trust-on-first-use approaches and similar are vulnerable [16, 74].

Wireless intrusion prevention systems (WIPS) capture packets and search for attack patterns [47]. When one is found, the WIPS alerts a supervisor and may terminate the overheard exchanges. However, our attack involves posing as a legitimate access point to the client and can even involve spoofing a jammed access point hardware address. Therefore, from the perspective of a WIPS, our attack looks no different from a normal association.

Device fingerprinting assumes the evil twin hardware differs from the victim deployment and sends malformed probe packets to elicit responses from the twin, which will be compared against a device table [26]. This approach requires the evil twin to respond to packets from
the probe. Our targeted jamming and response approach requires the probe to know the attack target beforehand, while our use of directional antennas forces the probe to co-locate with the victim, limiting its usefulness. In addition, an attacker using the same hardware as the victim is undetectable under this approach.

A substantial body of work in the security literature has studied attacks against password-based authentication, from the popular John the Ripper \cite{125} that relies of lists of common words, to time-space trade-offs approaches culminating in the well-known rainbow table \cite{83, 122} to probabilistic approaches such as Markov models \cite{119} and context-free grammars \cite{163} derived from public password lists. In the WPA world, wpacracker is a recent commercial effort focused on bringing cloud resources to bear on the problem of cracking WPA2-PSK passwords \cite{110}. We note, however, that in wpacracker’s target challenges are exchanged between two nodes with a shared secret. Our work, however, is an end-to-end attack against WPA Enterprise networks. The most significant similarity between the two efforts is their use of cloud computing to parallelize the plaintext recovery process.

While our attack relies upon a robust password cracking component to successfully recover WPA Enterprise pass-phrases, it is agnostic and, therefore, orthogonal to the underlying technique used. In our current prototype, we make use of parallel DES cracking techniques on GPUs and can use cloud computing nodes.

The attack we describe leverages vulnerabilities in user interfaces that fail to convey important security-relevant information to the user. Attacks in this vein have been known since the early days of multi-user computing, where mechanisms such as secure attention sequences—e.g., the now-infamous CTRL-ALT-DELETE—were introduced to establish a trusted path between the user and the operating system.

The particular vulnerabilities exposed by our work bear resemblance to a number of attacks that have been launched against the web browser, where the URL plays a similar role to a wireless network SSID in that users make trust decisions based upon the reputation of a particular domain or network name. In particular, homograph attacks \cite{64} have been used to mount phishing attacks against users that expect to visit a trusted domain by tricking them into visiting a site with a similar-appearing domain name by exploiting similarities between glyphs in
a character set—e.g., paypa1.com vs. paypal.com—or across character sets [156]. Our attack uses similar techniques, although an important difference in the context of wireless SSIDs is the general lack of delimiters, allowing for the use of invisible and non-printable characters.

Subverting the SSL/TLS PKI infrastructure relied upon by https to verify the authenticity of web servers is an important class of web security attacks that mirrors—to some extent—our user interface attacks in the context of WPA Enterprise. SSL/TLS has recently suffered a number of issues, such as the questionable trustworthiness of some certificate authorities [49, 65] that has led to the issuance of malicious, but correctly signed, certificates. Another interesting class of attack involves browser URL spoofing by, for example, creating SSL certificates that spoof trusted domains in vulnerable browsers by injecting null bytes in the certificate common name field [109].

2.3 LOCATION PRIVACY AND ANONYMOUS AUTHENTICATION

Work on anonymous credentials spans several decades [18, 24, 31, 40]. In anonymous credentials, clients create independent identities called pseudonyms with organizations who will authenticate them, and in turn clients receive credentials which they use on Zero-Knowledge proofs to authenticate themselves to organizations. Pseudonyms are created such that they do not reveal anything about the user apart from ownership of some credential, and two pseudonyms belonging to the same user do not reveal his underlying identity. In addition to the above properties, proposed anonymous credentials in the literature include other features such as protections against user sharing of credentials, user revocation, and delegation of credentials. While user revocation is practical in past work, it still requires relatively costly computation to perform, and the identity of the user can be retrieved either by the system’s CA, or deduced due to reuse of credentials. Our work provides immediate and unconditional credential revocation and user deregistration while maintaining the user identity hidden from the authenticating access points.

Authentication protocols providing proof of membership have been present in the literature for several decades. Group [41] and ring [133]
signatures allow members of a group to sign a message such that any third party can verify the message was signed by a member of the group, but not its identity. In both of these schemes signature size is linear in the number of group members, which does not scale. In addition, members entering and leaving the group in these schemes require new keys to be generated—an expensive operation—and to be provided to the members, limiting their practicality. Anonymous authentication by Schechter et al. [137] is also linear in the size of the group, but allows for dynamic group membership. However, in optimizing the scheme for large groups, a trade-off in privacy must be made by authenticating smaller client subsets. Jarecki et al. [93] allow members of groups to authenticate each other when they belong to the same group and without revealing affiliation or the identity of the group, but still depends on key redistribution when members leave the group.

A substantial body of work [19, 46, 67, 98, 142, 143] on Location-Based Services has been constructed over the years due to their widespread deployment on smartphones. The main idea behind much of these schemes relates to hiding location data to thwart the adversary through various means, some of them relying on collaborating with other clients to perform queries. While such strategies protect against parties that have no access to the location service itself, this would not be the case in the scenario of network access, where the mere act of authenticating to the hotspot provider already leaks user location and identity.

Broadcast encryption [22, 55, 60, 81] considers the distribution of protected content to authorized viewers, part of its interest being due to its applications in digital copyright management. User collusion is prevented by careful distribution of keys to clients, limiting practicality of key sharing. Part of our protocol may be formulated as a case of broadcast encryption, however existing schemes such as Fiat-Naor are designed for one-way channels with limited collusion protection.

Similarly, Logical Key Hierarchy (LKH) schemes [76, 118, 161, 167] seek to distribute a secret among $n$ recipients such that revoked members cannot decrypt new messages. For this the recipients are logically organized as leaves a tree structure and store their own key as well as the keys of every node in the way to the root. To update keys after the revoking node $x$, the root sends new keys to every node on the path to
x, reducing broadcast costs to $O(r \log (n/r))$ where $r$ is the number of revoked devices. Broadcast encryption and LKH require re-keying and communication with users, typically necessitating a channel that does not leak user location, and do not protect against a cheating server. Our work in contrast, does not make such assumptions about the channel, makes a contribution on how it may be constructed in practice, and provides clients with proof in the case of server wrongdoing.

Private Information Retrieval has been an active area of research starting with [45], where Chor, et al. showed information retrieval protocols over database replicas guaranteeing client privacy as long as at least some servers do not collude. These protocols provide information theoretic security. Single-database Private Information Retrieval (PIR) [8, 9, 69, 70, 99, 103, 113], in contrast, can only provide privacy against computationally bounded adversaries. PIR schemes seek to provide as much privacy as the trivial construction of retrieving the entire database, but with smaller than linear communication complexity in the number of entries. While traditionally PIR protocols use homomorphic cryptosystems like [51, 126] to compute the PIR response to queries, their practicality and scalability is limited due to the expensive operations involved. New lattice-based PIR has received significant attention recently [7, 8, 69, 84, 100] for their improved speed in comparison to number-theoretic algorithms, allowing for faster processing.

Oblivious-RAM (ORAM) techniques [72, 75, 130, 148], in contrast to read-only PIR, allow client read and writes to be hidden from the server. While the added protection to write patterns allows for greater flexibility, the overhead for write operations has little utility in the one-sided key distribution strategy we use for this work.

There has been significant interest in the industry on improving Wi-Fi client security for public hotspots. Protocol attacks ranging from key discovery to multi-layer Evil Twin impersonation [16, 20, 74, 82, 151, 162] are periodically being discovered and mitigated. New protocols and services that improve on the client experience and security have also been proposed [102, 164]. Yet the number of available anonymous authentication services for hotspots remains low, and is subject to the same limitations for group re-keying and message size as the schemes above.
The Feasibility of Residential-based Anonymity Networks

A growth of cloud applications aimed at end users has been ongoing in recent years. Video-on-Demand services like Hulu, YouTube and Netflix, cloud storage such as DropBox or iCloud, and social networks such as Facebook are ubiquitous, and have driven HTTP traffic the top once again [129]. As the demand of such services increases, privacy awareness among users has come to the forefront due to high-profile privacy violations and the concern for the unbalanced control data hosted in the application provider—e.g. pictures, video or social interactions. As demand for such services increases, content creators turn to Content Delivery Networks (CDNs), which place copies of the content in key locations to ease the load on networking infrastructure and to improve end-user latency and throughput. This strategy however, far from resolving privacy concerns, only serves to create more copies to the data in the network, often within a third party’s infrastructure. In time, CDN installations on major ISPs reduce the distance between the end user requesting content and the physical machines storing it, but at the same time increase the redundancy and attack surface of potentially sensitive information.

There has been increasing research interest in combining Peer-to-Peer (P2P) distribution and CDN for better scalability [85], reduction of cross AS traffic cost, better energy efficiency [160], user privacy [108], etc. Prior work has shown the potential gain of P2P CDNs and the locality of content demand [38, 136], justifying the P2P CDN model. How-
ever, there has been little work to determine whether current residential broadband provisioning is sufficient for content distribution, and what its achievable performance may be.

In this section, we present a testbed based on off-the-shelf hardware to gather residential usage statistics and a study of it to analyze the feasibility of residential data delivery though bandwidth aggregation. Using wireless home routers as the storage unit makes available an always-on device with moderate computing power and storage capacity. Since February 2011, we have deployed our testbed to 30 residences around the United States, covering around 100 users, and collecting 51GB in 153 million records of uplink and downlink usage data over a period of 4 years. A summary of our contributions and lessons learned is provided below:

- We study uplink usage patterns. Our analysis indicates residential broadband services are idle at least 40% of the time and up to 80% for certain periods of the day. Average throughput is less than 100 Kbps at least 70% of the time. Our measurements are consistent with previous studies, showing over 60% of HTTP traffic as the largest traffic type.

- We run live experiments on our testbed simulating content delivery to better understand the impact, and achievable levels of service when providing content. We show experimentally that with a proper traffic shaping configuration, uplink bandwidth can be shared fairly and without impact even while serving content for over 2 hours at 4 Mbps, the most common uplink cap in our current user base.

- We show anonymity can be built on top of our platform by building flows from a selection of cloud storage applications across our devices to serve data privately similar to networks like TOR [131]. Our results show devices can fill up available uplink up to capacity —4Mbps as in the experiments above—without disruption of flows originating from the LAN.
In this paper we consider traffic information from residential Cable and DSL subscribers over several ISPs in the United States. The data collection happens in off-the-shelf wireless routers running a customized firmware; data is later relayed to a central control server. Our participating demographics is mostly young (under 30 years old) students and professionals. We provided each participant with a wireless router to be connected to their home network and serve as its main wireless router. Once in operation the home devices associate to the access point to access the Internet. In order to collect link usage information, our APs log the amount of traffic traversing the WAN interface facing the ISP and sending this information periodically using our UDP-based heartbeat system. In addition to the above, we developed a suite of tools to ease the management of the devices. The remainder of this section describes the hardware and firmware choice, software tools developed for the system, and the collection methodology.

3.1.1 System Infrastructure

Our testbed consists of 30 Buffalo WZR-HP-G300NH wireless home routers deployed within the continental United States. The devices are 400MHz Atheros chips, with 64 MB storage and 32 MB RAM. In addition, the router provides a USB port we use for additional storage with 16GB USB flash drives. We centralize our server infrastructure at our Department in a computer whose role is to aggregate statistics observed in the home routers, and to present the management console. The server runs on a Core2 Quad Q9300 CPU at 2.5 GHz, 16 GB RAM, with MySQL as storage backend.

Figure 6: Testbed device and management console
The firmware running on the routers is based on OpenWrt, a Linux distribution customized for wireless access points [154]. To the user the router is fully configurable, with options such as web access control, WPA-PSK protection, WAN and Wireless LAN configuration. In addition, we encouraged the users to keep security tight by using their own passwords. We retain control of the devices through an ssh channel using RSA-2048 public-key-only authentication on the WAN port. All commands, experiments, data transfers and updates are performed through this ssh interface.

3.1.2 Features

Given the router nodes are geographically distant, we developed a set of tools to ease node administration and manage experiments. The source code for the basic image and customized tools are made available to other researchers for download and use [166]. The features we implemented are summarized below:

SOFTWARE MANAGEMENT Our system uses OpenWrt’s software management system with .ipk files cross-compiled on our central server. New versions of packages and custom tools built for our system can be deployed to routers in seconds.

BATCH FLASHING We automate the task of upgrading the router firmware by implementing our centralized batch flashing tool. Groups of access points can have their firmware upgraded at once without user intervention.

UPTIME MONITORING We track router health and online status with our heartbeat monitor, a daemon on the access point that periodically reports the current IP address, firmware version, number of clients, and bandwidth usage to the central server’s database. Heartbeat packets are small and have little or no impact on the user traffic.

VISUALIZATION TOOLS We graphically display status, health, and statistics of the router nodes based on the heartbeat information. The
visual tools also serve as a front-end to the management, firmware flashing, and software installation.

### 3.2 Delivery Throughput in a Residential Setting

One of the key concepts behind Content Delivery Networks is that providers replicate content on a set of distributed servers which are geographically close to the end users, keeping latency between content servers and end users small. For a set of geographically close residential devices within the same ISP to serve content amongst themselves the same requirements apply. Since latency is inversely related to throughput, low latency allows end users to fetch content from the source (be it a CDN or a peer device) with high throughput. In reality however, CDN performance is a more complex issue, and depends on a large number of factors, including server uptime, content hit ratio, server load, patterns of content demand, etc. These factors can often be far more important than mere packet latency. How to measure CDN performance has been a challenging research topic, and is beyond the scope of this work. Instead, our goal is to study the feasibility of residential data delivery services, and we are primarily interested in how fast can home routers serve their peers through their uplink.

Previous work with Open Infrastructure [94] examined the maximum theoretical latency required for the most common uplink speed available at the time (5Mbps) on a 64KB TCP buffer size using Mathis’ equation [111]. For a packet Round-Trip Time as high as 96ms, it is still possible to fully utilize such uplink. In fact, the observed intra-ISP RTT for our deployment falls below 30ms with high probability, allowing over 16Mbps theoretical upper bound, while 12Mbps is the fastest uplink within our deployment. This means latency is not yet a limiting factor for uplink utilization although subscriber uplink upgrades may make this the case in the future. Maximum throughput and latency within the Open Infrastructure are shown in Table 1 and Figure 7.
Table 1: Maximum TCP Throughput with 64 KB TCP Buffer Size on a Lossless Path

<table>
<thead>
<tr>
<th>Distance [12]</th>
<th>RTT (ms)</th>
<th>Throughput (Mbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regional: 500 – 1,000 mi</td>
<td>32</td>
<td>16</td>
</tr>
<tr>
<td>Cross-continent: ~ 3,000 mi</td>
<td>96</td>
<td>5.33</td>
</tr>
<tr>
<td>Multi-continent: ~ 6,000 mi</td>
<td>192</td>
<td>2.66</td>
</tr>
</tbody>
</table>

Figure 7: Latency measurement issued from Open Infrastructure Testbed

3.3 UPLINK MANAGEMENT

As discussed in previous section, latency is not the bottleneck limiting the uplink throughput performance, but ISP throttling. Thus, in residential CDN, how much uplink throughput that each home router can contribute heavily depends on the end users’ network usage pattern, and ISP’s network provisioning. We start this section by showing uplink usage statistics gathered on our testbed. We summarize the experiment details in Table 2.

In order to determine the impact of uplink sharing on the user device we setup an environment on our testbed that simulates the conditions of clients serving content to a remote machine. As the test runs we monitor the impact on client traffic caused by the uplink usage. The remainder of this section details the considerations, implementations and results of the test.
Table 2: Uplink Experiment Data Summary

| Home APs | 30 |
| ISPs     | Comcast (26), RCN (4) |
| Data Collection Time | Feb 1–Dec 13, 2014 |
| Mean AP Online Hours | 5183 |
| Throughput Samples | 55,987,203 |

3.3.1 Provisioning of home uplinks

In this section, we examine the statistics collected from our heartbeat monitors on the bandwidth utilization in the residential networks. Uplink provisioning of home users plays a central role in content distribution of peers because a slow link may reduce the downlink of a client drastically. As described in Section 3.1, each heartbeat message contains the total uplink and downlink bandwidth information and is reported to the server every 10 seconds. We classify the bandwidth utilization into 6 different classes of traffic load. Figure 8 and 9 show, for each traffic load class, the average bandwidth utilization—taken from approximately 40 million heartbeat records—for each 10-second interval in the 24 hour time window over everyday in 15 months and over all access points. We see from Figures 8 and 9 that the bandwidth is idle during most times of the day.

3.3.2 Backoff to Prioritize User Traffic

Data served by home users places a burden on the limited uplink available to devices. It is therefore important to ensure that served traffic does not slow down regular user traffic to a significant degree. Techniques and tools for traffic shaping, such as GNU/Linux’s tc are widely available, and are also supported on our setup by OpenWrt. However, use of traffic shaping tools assumes the AP is connected to the ISP-throttled interface, which is not always the case. In scenarios where the device sits behind another—through a Network Address Translation router, for instance—traffic shaping only sees the high-speed LAN connection between the primary and secondary devices and the ISP-
throttled link may still saturate, hurting the user’s performance. We discuss both cases in this section.

3.3.3 ISP Link Management

When the device has full access to the ISP link, it may use traffic prioritization. Without loss of generality, for our experiments it is sufficient to simply classify traffic in two classes: normal user traffic, and background upload traffic. The former will have the highest priority, and will be able to use the WAN link without contention. The experimental traffic will have a small traffic guarantee to keep it from stalling, and will borrow available bandwidth if available but it will yield immediately should any user data arrive. To implement this, we use classful Hierarchical Token Buckets, and Stochastic Fair Queuing within them to fairly distribute traffic belonging to the same class. We illustrate the setup in Figure 10.

To validate the correctness of the traffic shaping mechanism we ran two tests on real home router, to verify the background upload traffic will not affect end user’s normal traffic. In test 1, we first generate pure user download traffic for 10 minutes to obtain a baseline of AP’s download throughput. Then the background upload starts, emulating the
CDN content distribution, and is sustained for 20 minutes. As the background upload goes on, we generate another user download traffic. As shown in Figure 11, the user download throughput is not affected, because the TCP ACK streams in the uplink have been prioritized, and consumes limited bandwidth. In test 2, we examine the analogous case for uplink. We first starts background upload. In the middle, we issue a user upload traffic. As shown in Figure 12, the experimental uplink backs off properly, and the user uplink traffic takes over more of the uplink bandwidth. In this experiment, we configure tc to ensure a minimum 500Kbps bandwidth for background upload. Thus, the experimental upload does not back off completely.

Having full access to the ISP link allows the device to directly manage data as it arrives. The tc tool allows for a wide range of configurations using different queuing algorithms, and filters, so even more detailed guarantees could be achieved.

3.3.4 Backoff Without the ISP Link

When the device that must handle the backoff does not have access to the throttled interface, it must rely on external indications of link saturation. This is a common strategy in protocols like TCP in which packet
loss is used as the trigger to cut the congestion window, and thus the
sender rate. Another example comes in the literature in the field of
available bandwidth estimation in which packet probes are sent to the
destination on a fixed interval, and the time skew is measured on the
destination to obtain an estimate.

We examine the residential uplink behavior over varying loads by set-
ing up the test depicted in Figure 13. Test host A behind a given Open
Infrastructure node without any traffic control policy probes a well-
provisioned server with large UDP-packets 1-second apart. Test host
B, which shares uplink with host A generates uplink traffic, increasing
monotonically over 20 second intervals to a different host. We choose
this value to provide a relatively long term view of the link and to by-
pass any possible high-burst provisioning by the ISP. Probes are maxi-
mum size UDP packets, which will be fragmented by the routing hosts
on their way to the destination. The intention is two-fold: to provide
a fast baseline from the fragment spacing, and to detect packet losses
quickly at the probe receiver, as lost fragments result in total packet loss.
Figure 14 shows the change in packet spacing observed at the probe
receiver. The increased variance happens just as the traffic-generating
host saturates the uplink. Fragment spacing remained constant and on
the order of 1ms or less.
Figure 11: Downlink behavior with and without experimental traffic. No impact is apparent in the presence of competing uplink traffic.

Figure 12: Uplink observed with and without experimental traffic. The low priority class backs off as soon as normal traffic starts.

Congestion control protocols such as LEDBAT [141] use the difference between a target and the actual delay of the flow to adjust their congestion window to back off within one RTT interval. The window adjustment is linear in the difference of delays, positive when the target is greater, and negative when the measured delay overtakes. These protocols are implemented in recent versions of BitTorrent’s client and BTSync tools [21, 48].

3.3.5 Pushing the Home Uplinks to the Limit

In order to examine uplink performance of home devices under an anonymity network structure we simulate onion routing behavior. At
the start of experiment two devices are chosen uniformly at random from our pool to perform a transfer, and three other devices to serve as relays. We build a circuit similar to TOR [131], encapsulating traffic in a new layer for every relay hop. At every relay, the packet is unwrapped and forwarded to the next device. Flows from source to destination are TCP transfers lasting 10 minutes and are subject to shaping in favor of user traffic.

We estimate ISP uplink and downlink for every device before the experiment by running flows on both direction on each router with shaping disabled during off-peak hours to maximize throughput. Then we applied the traffic shaping policy described above.

During the experiment we monitored traffic uplink with our heartbeat monitor. As shown on Figure 15, all routers in the set achieved a mean uplink of close to the uplink cap set by the ISP, with small variances. This is consistent with our observations about the uplinks being idle most of the time. Not all our routers have the same uplink throttling, due to users choosing different plans with their ISP at the time of signing up for service.

Table 3: Total Uplink Throughput during the experiment.

<table>
<thead>
<tr>
<th>Bandwidth</th>
<th>Online Time (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 1 Mbps</td>
<td>0.34%</td>
</tr>
<tr>
<td>1 – 2 Mbps</td>
<td>26.1%</td>
</tr>
<tr>
<td>2 – 4 Mbps</td>
<td>34.9%</td>
</tr>
<tr>
<td>&gt; 4 Mbps</td>
<td>38.66%</td>
</tr>
</tbody>
</table>

To measure the uplink utilization on every circuit, we normalize the exit throughput on the last relay by the baseline uplink of the slow-
Figure 14: Probe spacing over increasingly loaded uplink

sest router in the chain. The behavior of the system is shown in Figure 16. Note how even though user uplink has the highest priority, it contributes little to the overall volume of traffic. Assuming the traffic shaping mechanism works correctly, we can see there is little impact in user traffic overall, and full link utilization is achievable provided similarly-powered uplinks are chosen in the circuit.

3.4 SUMMARY

In this Section we have presented the case for the feasibility of urban broadband installations for traffic delivery. We presented Open Infrastructure, which we designed to examine performance on urban home broadband with a deployment of 30 Access Points over the United States. Our long-term study shows upward trends in uplink provisioning from ISPs, as well as increased usage by clients over time.

We have shown using our Open Infrastructure platform how ISP throttling is the throughput bottleneck for transfers within an ISP. Even though uplink usage patterns show an upward trend over time, follow-
Figure 15: Mean uplink throughput for each experimental home router. Several devices stand out with over 6 Mbps. Uplink throttling can go as low as 1.5 Mbps.

Despite ISP service upgrades, underutilization of uplink is still prevalent, with almost 60% of traffic below 100Kbps over peak hours.

Content delivery from peers can make full use of the available uplink with little to no impact on flows originating from inside the home network using traffic shaping. Our tests are able to obtain close to full usage of the uplink even on a relayed structure, similar to the one in onion routing, with over 90% of flows reaching 95% of uplink capacity.
Figure 16: Cumulative distribution of normalized uplink throughput. Most of the experimental traffic reached the limit set by the ISP, while the user uplink was low.
The Fragility of Wi-Fi

Only a decade ago, gaining unauthorized access to a local network in an organization typically required physical access to the wired network. Today, the situation has dramatically changed. Wireless networks are ubiquitous, and allow users to have any-time, any-where access to information. At the same time, this convenience comes at a cost with respect to security: Wireless signals are not physically confined to the perimeter of an organization, but may be received by adversaries from very long distances. Therefore, although wireless networks have become an indispensable technology, they serve to increase the attack surface of an organization, and can potentially allow attackers to gain access to sensitive information over the æther.

Since their popularization and wide deployment, wireless networks have had a long history of security vulnerabilities. The initial effort to provide data confidentiality and client authentication—namely, Wired Equivalent Privacy (WEP)—suffered a widely publicized series of progressively more efficient attacks that necessitated WEP’s replacement by the WPA family of security protocols \[20\]. Today, WPA Enterprise is widely used to protect large enterprise wireless networks against unauthorized access. Trust in the security of WPA Enterprise stems from the use of proven security protocols for authentication (e.g., SSL/TLS), and the centralization of client authentication and authorization on well-protected authentication servers (e.g., RADIUS). However, while the individual protocols that comprise the WPA Enterprise security suite are well-tested and widely regarded as being secure, the composition of different features, components, and protocols has received less scrutiny.

In this paper, we present a novel, stealthy, and effective variant of the evil twin attack \[16, 52, 74\] against WPA Enterprise networks. The key
insight behind our attack is that the combination of cross-layer capabilities such as stealthy jamming using software radios, the inadequacy of wireless user interface mechanisms in popular commodity operating systems, and the insecure trust model used in wireless authentication makes real-world end-to-end attacks against wireless network authentication feasible. Our experiments with 17 technically-sophisticated users show that the attack is highly effective in practice and very difficult—if not impossible—for victims to detect.

Our targeted attack consists of four main phases. In the first phase, the attacker uses a software-defined radio to target the victim’s specific device, without impacting other users of the network. To accomplish this, the attacker uses targeted, stealthy reactive jamming techniques to deny access to the legitimate wireless network the victim would like to connect to. While recent work in the wireless community has made significant progress in developing mitigation techniques against jamming [28, 42, 104, 105, 127, 149, 165, 171], most of these techniques are still not part of wireless standards or deployed systems. Through reactive jamming, the victim is first disassociated from the legitimate network, and her probe requests are partially jammed to prevent legitimate access points from receiving them. We then send spoofed probe responses from a rogue wireless network. The combination of high-gain antennas and software radios makes such attacks possible from locations hundreds of meters distant from the targeted network.

In the second phase, the attacker takes advantage of predominantly unstudied and inadequate security mechanisms in popular commodity operating systems for disambiguating similar wireless network SSIDs. For example, the attacker can use character substitutions or invisible characters to create visually similar SSIDs—e.g., “Enterprise” vs. “Enter prise” —to trick users into connecting to a rogue network that is under the control of the attacker. This phase is an extension of the evil twin attack, where an attacker deploys a rogue access point that spoofs a legitimate wireless network.

In the third phase, the attacker presents a legitimate-appearing public key certificate, which can often be obtained for a cost of less than 200 USD. Since the certificates used in WPA Enterprise are not strongly bound to the network SSID, the victim’s device that connects to the
rogue network setup by the attacker has no basis for enforcing strict verification of certificates in popular commodity operating systems.

The fourth phase of the attack leverages the fact that WPA Enterprise deployments rely on the MSCHAPv2 [173] protocol for authentication, which has vulnerabilities that have been well documented [138]. MSCHAPv2 was initially designed for wired networks, and despite its use of outdated DES encryption, design flaws, and the availability of multiple automated cracking tools [39, 125, 168], MSCHAPv2 continues to enjoy wide usage, with nearly every major operating system and wireless infrastructure device supporting it. This can be partially explained by the fact that MSCHAPv2 is believed to be sufficiently secure when tunneled within an SSL/TLS session. However, the lack of a verifiable chain of trust from a CA to the network SSID allows an attacker to impersonate a trusted wireless network to capture victim authentication credential hashes.

In the final part of the attack, the attacker recovers plaintext authentication credentials by leveraging parallel password cracking techniques.

4.1 WPA Enterprise Background

In this section, we discuss relevant background information on WPA Enterprise. In particular, we focus on the WPA Enterprise authentication procedure, as well as implementation behavior when no known networks are available or when a new wireless network profile is created at the client. The reader is referred to the WPA Enterprise standard for further details [86].

4.1.1 Devices and Authentication

The WPA Enterprise authentication procedure involves several different devices: a client, an access point, and an authentication server. The client is a device with a 802.11-compliant network interface that requests access to the network. To connect to a network, the client communicates with an access point, which serves as a point of entry to one or more wireless networks. The authentication server is used to authenti-
cate users of the wireless network, and typically runs a network authentication protocol such as RADIUS.

WPA Enterprise authentication proceeds in three distinct phases: Discovery, Key Exchange, and Authentication. The most common methods used to perform the Key Exchange and Authentication phases are PEAP [115] and MSCHAPv2 [173]. With PEAP, the client initiates a TLS tunnel with the authentication server. During tunnel establishment, the authentication server presents its certificate, which ideally has been signed by a trusted certification authority (CA) that the client can verify and serves as the means of authenticating the network to the client. Once the TLS session has been established, MSCHAPv2 is then used to authenticate the client. Even though MSCHAPv2 has known flaws, the authentication procedure is commonly thought to be secure since it is encapsulated by the TLS session.

### 4.1.2 GUI Implementations

Network SSID lists on modern operating systems show only printable characters, with no way for the user to distinguish between identifiers that merely look similar. In addition, WPA Enterprise support differs considerably between platforms. Every client presents the user a different set of options when creating a new wireless network profile, some fields may be set automatically depending on user choice, and certificate management ranges from restrictive to permissive. We describe relevant behavior of common WPA clients below.

**Windows (XP and Above)** Figure 17 shows the wireless network selection list for Windows-based operating systems. Note that Windows displays network names with no visual aid to distinguish similar SSIDs. Instead, the system shows seemingly identical networks as separate entries in the list of available networks. For instance, in Figure 17, neutrino is displayed twice.

When a Windows client receives an authentication server certificate during the authentication phase, a summary of the certificate’s fields appears for verification. If the user accepts the certificate, the corresponding CA certificate can be used to verify the identity of the server for
Figure 17: Wireless network list for Microsoft Windows-based operating systems. Here, there are two seemingly identical entries for the SSID neutrino

Figure 18: Wireless network list for Mac OS X. Again, there are two seemingly identical entries for the SSID free-wifi

this network under “Trusted Root Certification Authorities.” The user may also specify that the network should be authenticated only when the certificate’s common name field matches a particular name. By default in Windows 7, the server name field is set to the value seen in the certificate, if accepted.
APPLE  Figure 18 shows the wireless network list for Mac OS X. Similar to Windows-based systems, Mac OS X also shows available SSIDs without visual aids. When creating a new network profile, the system selects most of the network parameters automatically – e.g., the use of PEAP. Once the authentication server presents its certificate, the client will present the user a summary of the certificate’s fields for inspection. Mac OS X provides a visual aid in the form of quotation marks to delimit the SSID of the chosen network. In contrast to Windows clients, however, it is unclear how to restrict connections to a specific server name for a new network profile from the OS X GUI. Like Mac OS X, iOS uses quotation marks to visually delimit network SSIDs in the username and password dialog.

GNU/LINUX FAMILY  GNU/Linux systems offer a variety of WPA client configuration interfaces, NetworkManager and wpa_gui being the most common. In both cases, the new network configuration dialogs are similar, and none offers visual aids to distinguish similarly-named networks. We note that text-based configuration tools allow checking for the correct authentication server name, and to distinguish similar SSIDs by displaying them within quotation marks. However, text-based configuration remains an advanced task out of reach for the common user.

4.2 TARGETED, STEALTHY EVIL TWIN ATTACKS

In this section, we sketch an overview of our advanced, stealthy evil twin attack. The goal is to subvert the WPA Enterprise authentication described in Section 4.1, such that an attacker tricks a victim client into unwittingly authenticating to a rogue network with their credentials for a real, trusted network. The capture of these credentials allows the attacker to then authenticate to the trusted network with the privileges of the victim client. We note that this attack, when correctly performed, is completely transparent to the victim – that is, the victim will be entirely unaware that their network authentication credentials have been leaked.
In the following, we first detail the threat model we assume for the attack. Then, we present an exposition of the details of the attack, including several variations that increase both its effectiveness and stealthiness.

4.2.1 Threat Model

The attack we present in this work makes a number of realistic assumptions regarding the configuration of the wireless network and victim clients, as well as the capabilities of the attacker. We enumerate these assumptions here.

1. The Wi-Fi network uses WPA Enterprise for authentication, and uses MSCHAPv2 to authenticate clients over RADIUS. This is a common configuration; for instance, it is the default on recent versions of Windows.

2. The attacker can successfully communicate with a target wireless network, and transmit with sufficient power to successfully jam legitimate clients of the network. Section 4.4 shows the distances satisfying these requirements.

3. The attacker has sufficient resources to mount the attack. We demonstrate in Section 4.4 that the attack is feasible on common, high-end servers.

4. The victim clients run one of several commodity operating systems, including: Windows XP or later; Apple's Mac OS X or iOS; or, GNU/Linux with common GUI-based configuration tools.

4.2.2 Attack Description

In the following description of the targeted evil twin attack, let \( C \) be a legitimate client of the victim network \( N \) that uses an access point \( AP_N \) advertising SSID \( S_N \). Let \( R_N \) be the victim network’s authentication server that has been set up to perform PEAP with MSCHAPv2 for authentication with a certificate signed by certification authority \( CA_N \), and can access the user database for \( N \). In this scenario, \( C \) has a stored
Figure 19: In the attack, an attacker leverages reactive jamming techniques to coerce a victim client to authenticate to a rogue network that appears identical to a target wireless network. A hash of the victim's authentication credentials is then captured. The hash is cracked using a high-performance password cracker (not shown). Once the plaintext password has been recovered, the attacker uses it to authenticate to the target network with the privileges of the victim client.

The attacker A sets up an access point \( AP_A \) in range of the client C, using the same channel as \( AP_N \). \( AP_A \) uses an authentication server \( R_A \) with a certificate similar to \( R_N \).

The attack is illustrated in Figure 19, and proceeds as follows.

1. \( AP_A \) sends Disassociation frames to C, forcing C to reassociate. These frames can be spoofed easily to appear being issued by \( AP_N \) using standard tools such as aircrack-ng [10].

2. C sends Probe Request frames to scan for active access points in order to reassociate with a suitable wireless network.

3. \( AP_A \) reacts to every Probe Request frame from C by jamming it and broadcasting Probe Response to C with SSID \( S_A \). This has the effect of preventing reception of C’s Probe Request frames at \( AP_N \), allowing \( AP_A \) to impersonate \( AP_N \).

4. C connects to \( S_A \) advertised by \( AP_A \), thereby creating a new network entry in the client’s wireless network configuration database.
In effect, the attacker abuses the user’s trust in the legitimate SSID $S_N$.

5. $R_A$ presents $C$ with a certificate signed by a trusted CA to avoid arousing suspicion.\(^1\)

6. Normal authentication proceeds, such that $C$ discloses a hash of his authentication credentials to $R_A$.

7. $A$ transfers the captured hash to cloud-based password cracker to recover the plaintext authentication credentials.

In the following sections, we elaborate upon each of the steps of the attack outlined here. In particular, we describe the jamming strategy required to coerce $C$ to communicate with $AP_A$. We then provide a procedure for choosing a suitable value for $S_A$ and the properties necessary for $R_A$’s TLS certificate. Finally, we outline a procedure for efficiently recovering authentication credential plaintext.

### 4.2.3 Reactive Jamming

Jamming a radio signal is often described as a physical operation where the jammer’s transmitter outputs energy into the medium to disrupt communication without further knowledge of the data it is trying to

---

\(^1\) We discuss issues surrounding obtaining trustworthy certificates later in the section.
disrupt. In the context of our attack, however, the jammer needs additional higher-layer knowledge of the victim's transmitted data in order to react properly. For the purposes of our work, a reactive jammer is a device that is capable of A) examining higher-level network protocol data contained in wireless frames, and B) conditionally jamming a target based on intercepted data prior to that frame's end of transmission. A depiction of this technique is shown in Figure 20.

In our attack, the access point $AP_A$ operates by waiting for Probe Request frames from a client. Once a frame is detected, but before transmission ends, $AP_A$ sends a continuous train of Probe Response frames that do not obey SIFS, DIFS, or PIFS timings in the 802.11 standard and lasts long enough to ensure that $C$ receives it. If the original frame is jammed before it reaches $AP_N$, there will be no response from it, removing its entry from the SSID list at the client $C$.

Successful jamming requires the attacker to respond quickly to transmissions from $C$. Let $t_o$ be the transmission time of $C$’s Probe Request frames, $T_C$ the time required to transmit one Probe Request frame, $T_T$ the radio turnaround time of $AP_A$, $T_b$ the transmission time of 1 bit, and $t_d$ the time at which $AP_A$ detects a Probe Request frame from $C$. To successfully jam the Probe Request frame, $T_T$ must satisfy the inequality

$$t_d + T_T < t_o + T_C - T_b.$$  \hspace{1cm} (1)

For instance, with a rate of 1.0Mbps and frame size of $\approx 600$ bits, $T_C < 600\mu s$.

Even though the above technique only works for a fixed channel, Section 4.2.7 discusses the case where other channels must be jammed. Elimination of competing access points in the client’s SSID list serves as the first step of the attack.

4.2.4 SSID Selection

As alluded to in Section 4.1.2, wireless network configuration interfaces provided by common operating systems can render maliciously-named

---

2 SIFS, DIFS, and PIFS are inter-frame spaces of different magnitudes. Shorter inter-frame spacing gives a transmitter greater transmission priority on a wireless channel. By necessity, reactive jammers ignore such inter-frame spacing standards.
networks indistinguishable from legitimate networks to the average user. In the simplest case, all that is needed to exploit this shortcoming is to use an SSID with a trailing or leading non-printable character in its name – e.g., “Enterprise” vs. “Enterprise.”. However, as in the case of browser URL spoofing, substitution of similar glyphs can produce spoofed network names. The standard defines SSIDs as an arbitrary string of up to 32 characters, and leaves the interpretation open to the implementation \[86\]. Given the abundance of glyphs available, especially in extended character sets, this implies that the set of possible spoofed SSIDs available for use by an attacker is potentially large.

When the victim receives a Probe Response frame with a spoofed SSID, a new network profile will be created. None of the security settings for the legitimate network will apply to the new profile. Creation of a new network profile is a critical step of the attack, and it illustrates how subtleties in user interface designs can enable serious attacks despite the use of theoretically strong authentication protocols.

4.2.5 Authentication Server Certificates

Once a new profile exists in the client for the spoofed network, the attacker’s authentication server must present a valid certificate to the client. To accomplish this task, the attacker can register a domain that appears similar to a trusted domain, or one that is related to wireless networking in some way (e.g., in our experiments, we registered the domain openinfrastructures.net). Since many wireless security products ship with self-signed certificates, users may be conditioned to accept any certificate that has a semblance of validity. We test this conjecture in Section 4.4 with user experiments, and show that it holds in practice.

The attacker uses the same CA used by \( R_N \) to sign her own certificate and bind it to a name in the registered domain. For instance, a deployment that uses VeriSign to sign its certificates can be targeted by an attacker registering a domain, and signing her own certificate with VeriSign as well.

Choosing a certificate may even be made simpler if a deployment chooses to use its own internal CA to sign certificates. If this is the case,
then any CA in the client’s trust store that is willing to sign certificates—e.g. VeriSign—is enough to prevent the operating system from reporting an error to the user.

Once the user accepts the attacker’s certificate for the new network profile, the TLS tunnel between the victim client and the attacker’s authentication server is established, and authentication commences. Crafting a reasonable certificate constitutes the final step in giving the attacker access to the less secure authentication protocol—MSCHAPv2—that is encapsulated by TLS.

4.2.6 Recovering Authentication Credentials

Once the client starts authentication, the attacker can record exchanged frames, or even act as a full man-in-the-middle, forwarding frames between the client and the victim network. In both cases, the attacker can intercept the victim client’s authentication credentials.

Breaking MSCHAPv2 is well documented [138], and tools such as asleap [168], mschapv2acc [39], and John the Ripper [125] can recover the plaintext secret given the conversation handshake. The speedup over brute-force search claimed by exploiting MSCHAPv2 weaknesses is $2^{16}$, and this can be improved in practice by distributing the search over a pool of high-speed computation nodes. For instance, GPU computing platforms such as CUDA [121] allow the attacker to massively parallelize the key search.

To summarize the attack against NTLM, given a key $K = K_1 || K_2 || K_3$, the attacker must first obtain the values for either $K_1$ or $K_2$; obtaining $K_3$ is trivial, and only provides 2 bytes. Cracking DES once yields one of $K_1$ or $K_2$, giving a total of 9 out of the 16 bytes of the NTLM hash. The last 7 bytes of the hash can be obtained by running a dictionary attack against NTLM. In Section 4.4, we show how we were able to crack real-world passwords in a short period of time, and also empirically evaluate the computational effort required to completely execute this step of our attack.
4.2.7 Optimizations

While the basic procedure outlined above contains the essential details of the attack against WPA Enterprise, a number of optimizations are useful to consider when mounting this attack in the real-world. We consider possible improvements in the following.

The basic version of the attack might require the attacker to be in close proximity to the victim. Depending on the target, this may not be possible without arousing suspicion in the victim’s activities. However, inexpensive directional antennas can make a remote attack possible from great distances. We examine the performance of one such antenna in Section 4.4.

The version of the attack presented above can only target a single wireless channel, and is limited in practice as Wi-Fi deployments move towards higher access point density; this has the effect of making more channels available to clients. For these scenarios, the attacker could use an additional jamming device for each of the available channels, using the same principle of operation as the basic attack. Another strategy would involve jamming the non-overlapping channels not targeted by the reactive jammer, essentially forcing the victim client to use the desired channel covered by the reactive jammer.

In order to convince more security-minded users to accept the rogue certificate, other social engineering techniques are possible. Consider an attacker AP that contains its own certificate and the one the legitimate authentication server presents. In the first iteration, \( AP_A \) uses the legitimate certificate and advertises \( S'_N \), where \( S'_N \) is a visually equivalent variation of \( S_N \). The user can inspect this certificate and since it will be the one expected, he will accept it. TLS session establishment will fail, and \( AP_A \) switches SSID to \( S''_N \), another visually similar variation of \( S_N \), creating yet another network. At this point the user may be more likely to accept the rogue certificate, as he has already inspected and approved one.

Using arrays of GPUs to parallelize the computation can shorten the time needed to crack DES. Buying and managing such an array can be costly, but our evaluation in Section 4.4 demonstrates that current offerings such as Amazon EC2 [13] can greatly reduce these costs.
In the current section, we describe our implementation of the advanced, targeted evil twin attack. Our prototype, shown in Figure 21, uses a desktop computer running Gentoo Linux for coordination. The machine contains an Intel Core 2 Quad Q9650 3GHz processor, 4GB of RAM, and an NVIDIA GeForce 9800GT graphics card with GTX280 GPUs. The reactive jammer is implemented using two USRP2 software-defined radio boards from Ettus Research [59], each with one RFX2400 daughterboard. A Buffalo WZR-HP-300NH wireless router serves the rogue wireless network, and two HyperGain HG2419G 2.5GHz 19dBi parabolic grid antennas are used to increase the range of the attack.

The reactive jammer is a GNURadio-based software defined radio (SDR) [71], running an 802.11b module developed by BBN [17] on our desktop host. The two USRP2 boards and daughterboards, one for reception and one for transmission, connect to the host through Gigabit Ethernet adapters. The USRP2s are connected to the directional antennas.

The reactive jammer is provided with the target client and access point MAC addresses, and the desired spoofed SSID. With these parameters, the jammer builds a Probe Response frame that is then passed to the modulation blocks for 802.11. The returning data is the train of signals to repeat on the sender, which is stored in memory. Once the receiver thread starts, the frame decoding function fills the received data buffer from the incoming signals, checking for the target client MAC address. When the desired Probe Request frame is encountered, the jammer transmits the stored Probe Response train.

For our rogue authentication server, we use FreeRADIUS [63] “Wireless Pwnage Edition” [169], a patch that maintains a challenge and response authentication log. Also connected to the host is the Buffalo router running the OpenWrt 10.03 firmware [154]. The router connects to the desktop computer, and is configured for WPA Enterprise using our desktop as a RADIUS server. We bought, registered, and obtained signed TLS certificates for openinfrastructures.net and radius.openinfrastructures.net. Certificates are signed by a reputable certification authority trusted by all common operating systems.
We explored four separate approaches for recovering plaintext from captured authentication hashes. We evaluated the use of a 9800GT card, a GTX 280 card, a Tesla S870 cluster with 8 GPU boards, and an Amazon EC2 Cluster GPU Quadruple Extra Large Instance with two M2060 Fermi CPUs. We evaluate the relative impact of each approach on the efficiency of our attack in Section 4.4.

In this section, we report on an evaluation of our attack prototype. First, we establish bounds on the reaction time of our jammer, and the computational overhead required to recover authentication credential plaintexts. We evaluate the overall effectiveness of our prototype against a production WPA Enterprise-protected wireless network of our own. We quantify the range of our prototype in a separate experiment conducted using high-gain antennas in an urban environment. We report on the real-world effectiveness of the attack with user experiments with 17 technically-sophisticated participants. Finally, we present an economic, worst-case analysis of the attack in terms of the cost of the hardware and the software required to guarantee the success of the attack in practice against a particular victim.
Figure 22: A demonstration of the reaction time of our jammer. On the left, the signal analyzer shows an uninterrupted 800µs frame transmitted by a wireless client. On the right, the signal analyzer shows a Probe Response frame jammed at 300µs. The horizontal grid width denotes 200µs divisions.

4.4.1 Ethical Considerations

All of the experiments described in this section only target devices under our control, or ones for which we have obtained prior consent for testing. The experiments were performed in a wired environment whenever possible. However, to demonstrate the effectiveness of the attack, it was also necessary to perform it over the air. Note that fine-grained control of the attack is possible due to the nature of our jammer implementation that reacts only to specific MAC addresses. For test devices that we did not control, we obtained prior consent from the targeted users, and debriefed them after the experiments.

4.4.2 Jamming Speed

Recall from Section 4.2.3 that for the reactive jammer to successfully block transmission of the victim client’s Probe Request frames, the radio turnaround time of the jammer must satisfy Eq. (1). This turnaround time effectively determines the maximum reaction speed to detected signals. Therefore, we performed the following experiment to measure the reaction time of our jammer.

In this experiment, we RF-wired a USB wireless dongle with an external antenna adapter to our jammer through a pair of 30dB attenuators. The output of the dongle and jammer transmitter is displayed on our signal analyzer.
Figure 22 illustrates the reaction time of our jammer. On the left panel, a single Probe Request frame lasting $800\mu s$ is shown while no jammer is active. On the right panel, the same Probe Request frame is interrupted by the jammer $300\mu s$ after transmission starts, establishing an upper bound on the reactive jamming capability of our prototype.

Because 802.11g network management packets are sent at the lowest rate (1 Mbps) to ensure delivery [86], our prototype’s minimum response time of $300\mu s$ means that our reactive jammer is able to jam 802.11g management packets at around byte 38 using our software-radio implementation. Additionally, we note that while this reaction time is too long to allow for reliable jamming of 802.11n-based wireless networks, hardware is available that would render the attack possible in that context as well.

4.4.3 Plaintext Recovery

As we mention in Section 4.3, we explored several options for recovering authentication credential plaintexts. In particular, we deployed GPU-optimized password crackers on a low-end NVIDIA 9800GT card, a medium-end GTX 280 card, a Tesla S870 cluster with 8 GPU boards, a Tesla C1060 cluster with 8 nodes, and an Amazon EC2 Cluster GPU Quadruple Extra Large Instance with two M2060 Fermi GPUs. To empirically demonstrate the required computational overhead to recover authentication credential plaintexts, we measured MD4 hashing speed by performing a full dictionary search on each device.

Figure 23 presents a hashing performance comparison between different hardware and thread configurations. To generate this plot, we ran an exhaustive dictionary search over a space of 8 alphanumeric characters with increasing threads per CUDA block to a maximum of 512 blocks. The dictionary build times range from a little over a week to approximately 13 hours.

As expected, the number of hashes computed per second initially increases with increasing parallelism, but eventually levels out or decreases due to thread contention. We note that for the 9800GT, the drop in performance at 512 threads is due to the configuration being too large
Figure 23: GPU hashing speed as a function of threads per CUDA block for dictionary generation. In particular, we compare the relative efficiency of four different hardware implementations for the card memory. We also note that a more careful implementation of the distributed cracker might improve performance at scale.

Once a full dictionary is generated, plaintext recovery requires at least one DES encryption per entry in the dictionary. An efficient GPU implementation [6] would allow the recovery to be fully-parallelized. However, the speed gain factor of the above implementation against a single CPU is around 10, meaning that a host with multiple cores is a reasonable substitute for the GPU implementation.

4.4.4 Attack Range

To quantify the maximum distance from which our attack could be mounted by our prototype, we deployed our high-gain directional parabolic antenna on the 16th floor of a building in a large U.S. city. The antenna was connected to a computer running Kismet through a TL-WN722N USB wireless dongle. We recorded the MAC addresses of a number of access points, and queried these addresses using the Skyhook Wireless Positioning Service [146] to give an approximate distance from the placement of the antenna. The results show that our 19dBi antenna is able to communicate with networks 800m in distance.
To further quantify the performance of our prototype, we measured the jamming and association success probability of our system. We placed our prototype on the 4th floor of the same building as in the previous experiment to limit its range, and co-located an access point for a target wireless network. Then, we varied the position of a test client at locations 50 meters apart in line-of-sight of the transmitter. As a test client, we used a GNU/Linux laptop containing an Atheros-based wireless interface and the factory antenna. For each position, we ran 1,000 network scans. Figure 25 sketches the relative position of all the components for the experiment.

For the chosen channel, we consider a single jamming attempt to be successful if the scan reveals only our system’s MAC address. Failure to see our system in the scan, or seeing other transmitters in the channel, indicates a failed jamming attempt. Similarly, we consider an association to be successful if the test client associates with our rogue access point. If, on the other hand, the client connects to the legitimate wireless network, then we consider the association attempt to be a failure. The results for both tests are shown in Figure 24.
The results indicate that our prototype is able to jam and force client associations with close to 100% probability at ranges under 100m. As expected, success probabilities decrease with increased range; however, they do remain non-negligible up to 400m. We note that although these probabilities might indicate that the attack becomes ineffective at larger distances, the attackers are not constrained to 1,000 trials. Indeed, an attacker could quite feasibly perform hundreds of thousands or millions of trials in relatively short periods of time. Though this strategy comes with a corresponding increase in the risk of detection, it is not necessarily the case that the true risk is greater since one would have to be looking for signs of the attack in the first place.
4.4.5 User Experiments

Similar to work conducted by Jakobsson et al. [91, 92], we believe that realistic experiments are the only way to reliably estimate success rates of attacks in the real world.

In order to assess the feasibility of the attack we describe in this paper, we performed user experiments with 17 volunteer computer science graduate students that we recruited in our department. All participants were technically-sophisticated – that is, the participants were expert Internet and knowledgeable wireless network users. We set up the attack prototype in the lab with small Antenova B4844-01 antennas to limit range. The system was modified to stop jamming and answering requests for clients already seen and captured.

Before the experiments, we informed all participants that we would be capturing traffic, but without revealing the concrete attack that we were launching. The participants did not know that we were performing a security experiment. Furthermore, we reassured our participants that we would not be accessing any personal information (e.g., email contents, Facebook messages, etc.). Also, we anonymized captures when cracking such that no user could be mapped to a password, and only processed information automatically (i.e., we did not manually look at it).

For each participant, we assigned a series of common, innocuous tasks such as browsing the web, sending email through a web interface, and CAPTCHA solving after authenticating with the university wireless network that we were targeting. Establishing a connection to the wireless network we were targeting was, therefore, not an end in itself, but was instead the means of accomplishing another unrelated goal. This was to ensure that the users were not aware of the attack that we were launching so that we could determine how effective the attack would be in a realistic, real-world setting. When available, the participants performed the tasks using their personal computers. Otherwise, we provided an Asus 1005PE netbook running Ubuntu 11.10.

After the completion of each user experiment, we debriefed the participant on the real purpose of the tests, asked if they had noticed any suspicious behavior during the experiment, and asked them to complete a self-assessment of their technical abilities. Figure 26 shows the
distribution of scores reported by the users. Note that most participants were technically-sophisticated. Hence, the results we report in this section demonstrate that the attack works effectively even against very knowledgeable users.

Our prototype was able to successfully perform a man-in-the-middle attack against all of the users who participated in our user experiments, including capturing the MSCHAPv2 conversation. All users authenticated with the wireless network that we were targeting, and only one reported seeing suspicious activity (even so, she still provided her password to the system).

To determine how long it would take for us to crack a password to gain illegal access to the network, we performed a brute-force search of character alphanumeric passwords over the anonymized captures using John the Ripper on a Xeon-CPU server at 2.4GHz in our lab. We parallelized the search to use one candidate password on each CPU. Table 4 shows that we were able to successfully crack a password after about 30 seconds, and then another one after about two hours. Hence, our attack is feasible in practice and can be used to gain unauthorized access to WPA Enterprise networks.

<table>
<thead>
<tr>
<th>Password</th>
<th>Time to find (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>27</td>
</tr>
<tr>
<td>2</td>
<td>7,510</td>
</tr>
</tbody>
</table>

Table 4: Password search time, ordered by elapsed time

4.4.6 Economic Analysis

We split the estimated cost of building a system capable of launching our attack into two parts. First, we quantify the cost of the hardware we used, shown in Table 5. The total cost is dominated by the USRP2 SDR boards, at $3,000 USD, which is unsurprising given the relatively specialized nature of the hardware. Nevertheless, the final cost of $4,470.45 USD is well within the reach of many motivated attackers.

Next, we analyze the cost of the plaintext recovery components, including the MD4 dictionary generator to map strings to hashes, and the
Figure 26: Results of a self-assessment of familiarity with computers and wireless networking performed by participants in our user study. Users tended to view themselves as technically sophisticated with respect to computing in general, while familiarity with wireless networking exhibits a bimodal distribution. In the latter case, most viewed themselves as sophisticated, while a smaller group classified themselves as less competent.

Table 5: A breakdown of the cost for each component of our attack prototype.

<table>
<thead>
<tr>
<th>Component</th>
<th>Cost (USD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Desktop Core 2 Quad 4GB RAM</td>
<td>580.00</td>
</tr>
<tr>
<td>2 USRP2 boards @ $1500 ea.</td>
<td>3,000.00</td>
</tr>
<tr>
<td>2 RFX2400 boards @ $275 ea.</td>
<td>550.00</td>
</tr>
<tr>
<td>1 Buffalo WZR-HP-300NH AP</td>
<td>66.00</td>
</tr>
<tr>
<td>2 Parabolic grid ant. @ $47.99 ea.</td>
<td>95.98</td>
</tr>
<tr>
<td>1 Standard TLS certificate</td>
<td>178.47</td>
</tr>
<tr>
<td>8 Tesla C10 computing devices @1,050 ea.</td>
<td>8,400</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>$12,870.45</strong></td>
</tr>
</tbody>
</table>

DES-cracking implementation if the attacker would like to have guaranteed success in cracking a specific victim’s password. Agosta et al. [6] and Guneysu et al. [86] examine the use of equivalent GPU and FPGA clusters respectively for performing key searches in the $10,000 USD range. A cluster of 48 NVIDIA GTX260-216 boards completes a single DES in 18 days, while COPACOBANA does so in 12. Taking the former as an upper bound for the time required to completely explore the possi-
ble space of keys, the budget for such a plaintext recovery component in under 20 days comes to less than $15,000 USD when using the Amazon EC2 GPU extra-large instance for dictionary generation.

From the above cost analysis, our attack can be performed for under $20,000 USD if a specific user is targeted rather than when the attacker is interested in gaining general access to a WPA Enterprise network as we demonstrated in our user experiments. We believe that this renders the attack feasible for a wide range of attackers who are interested in launching targeted attacks—e.g., criminal gangs, well-funded corporations, and nation-states. Note that for those entities with significant resources—e.g., the security agency of a country—the attack could be performed in a far more efficiently than above by investing in faster resources.

4.5 COUNTERMEASURES

Preventing our attack requires coordination by defenders on multiple layers. Several countermeasures are required to mitigate such attacks. In this section, we present and discuss some possible mitigation strategies.

A central component of our attack concerns the spoofing of network SSIDs in wireless network user interfaces. Implementations that can provide visual cues to the user that a rogue SSID is distinct from a similar-appearing trusted SSID could neutralize or warn against the attack. For example, SSIDs might be displayed between visible delimiters, or visible placeholders might be displayed in place of non-printable characters. More sophisticated approaches might check for SSID similarity, similar to phishing heuristics integrated into modern browsers, and alert the user in such cases. Note, however, that educating users to take notice of security-relevant information is a notoriously difficult problem, especially in a wireless setting where most users are not aware of the risks. For example, session-hijacking tools such as Firesheep have received much attention because of their simplicity and effectiveness in wireless environments.

Unfortunately, flaws in MSCHAPv2 and the wide availability of distributed computational resources make it feasible to recover plaintext
user credentials. Attacks against it [138] are well-known and are implemented in a variety of tools. Perhaps worst of all, these attacks have negligible parallelization overhead. Our work supports the fact that more secure alternatives to MSCHAPv2 need to be deployed in WPA Enterprise networks. Even small changes to MSCHAP, such as using a single AES encryption for the client response, could make the attack we describe infeasible. Also, using client certificates for authentication would avoid relying on password-based authentication protocols altogether.

In this work, we show that the lack of a strong binding between the SSID and the authentication server certificate is a significant security problem. Tying the wireless network SSID to the authentication server certificate gives the victim a chance to detect the attack and determine that she is actually contacting a rogue access point. WPA clients should also require certificate authority selection and common name verification to protect from SSID spoofing attacks.

To close the gap between the TLS certificate and the host network a system such as SNEAP [33] could be used. With a flexible authentication framework that relies on the protection inherent in web browser’s SSL implementation and the OAuth framework for websites we create a link between the AP identity and its authentication server.

The key insight behind SNEAP is the leveraging of the SSL configuration in the user’s browser to establish the identity of the authenticator. Figure 27 gives an overview of SNEAP’s architecture and operation.

We implemented our SNEAP prototype under windows and Linux clients, and Open Infrastructure router devices running our modified supplicants and RADIUS servers, respectively. Our implementation uses Facebook as the authentication backend, but the framework can be easily extended to any other OAuth-type system. The system is an EAP (802.1x) extension for WPA supplicants, RADIUS and authenticator can prevent the Evil Twin attacks by linking the identity of the authenticating party (Facebook in our implementation) with the specific network the user tries to use. Figure 27 summarizes SNEAP’s architecture and operation.

SNEAP itself does not provide for anonymous authentication however, and in fact the social network identities of the access point owner and client are revealed in order to find a relationship between the two. Additionally, the very use of an online social network, means we give
up on privacy and instead place trust directly on the social network. We present a better solution in Chapter 5 that can protect identities and authenticate clients using a special form of Private Information Retrieval.

Allowing users to configure their own network profiles to be able to connect to an organization’s wireless network may be convenient from a management and organizational point of view. However, this flexibility creates the potential for attacks such as ours. For deployments using WPA Enterprise, we believe that it might be better for a central authority to distribute wireless profiles to clients, and disallow dynamic profile creation.

While our attack makes use of directional antennas to increase the range of the attack, physical security techniques (e.g., secure pairing) have recently been proposed to mitigate rogue device attacks [29, 112]. Work in this vein, although mainly focused on short range handsets, could potentially be adapted to newer 802.11n multi-antenna wireless clients and WPA Enterprise networks. This could severely limit the range of the attack, and force the jammer to be co-located with the victim clients, reducing the problem to one of physical security.
4.5.1 **SNEAP: The Social Network-Enabled EAP**

As mobile devices evolve, end users have ever increasing demand for ubiquitous network access. Wi-Fi, being now commonplace, has the potential to fulfill this demand. Apart from Wi-Fi hotspots deployed by ISPs, home users start showing interest in sharing bandwidth with others (e.g. Fon bases its business model on users sharing open access points.) However, all existing Wi-Fi sharing approaches are unsecured due to the difficulty of distributing access keys, discouraging potential users. Meanwhile, social networking provides a large scale, well established social graph, which is an attractive candidate for authentication services. Previous work on social networks as authentication mechanisms such as Social Wi-Fi requires users to first register with a third-party server through an open Access Point (AP), which adds an unneeded step to the process, and complicates user management at the third-party.

Our Social Network-Enabled EAP method (SNEAP) integrates the authentication services in social networks with the widely-adopted EAP framework. In addition, the extensibility of EAP and our software-based solution allow easy incremental deployment, and our chosen platform offers broad hardware compatibility.

4.5.2 **System and Protocol Design**

Our system consists of a SNEAP-extended supplicant, AP, Radius server, and a Facebook application. Any AP running our custom OpenWrt firmware can be registered by the owner through our Facebook application to obtain a SNEAP ID, associated with the owner’s Facebook profile at the Radius server.

SNEAP authentication consists of two phases shown in Figure 28. In Phase I (steps 1 and 2), the requesting client authenticates the Radius server by setting up a TLS tunnel. After a WPA key exchange, Radius instructs the AP to authorize the client with restricted access, allowing only traffic pertinent to authenticating with the social network. Thus, the client obtains a secure link to the AP early in the process.
Phase II (steps 3 onward) starts with the supplicant opening a browser to authorize our Facebook application to obtain the client’s authentication code (AC), which will be used by the Radius server to access his profile. Since Facebook authentication is browser based, our application redirects the AC to a tiny web server integrated into the supplicant, which in turn will forward it to the AP. The AP will send the (AC, SNEAP-ID) pair to the Radius server for friendship verification. Upon success, the Radius server will instruct the AP to instate full access for the client.

Figure 28: SNEAP Architecture and Flow

Figure 29: 1. supplicant 2. Client Authentication Portal 3. AP Registration Facebook app
In showing the vulnerability of what is considered the most secure Wi-Fi deployment we illustrate how even constructions of secure components can be attacked separately to subvert them, exposing user data. The key insight behind our attack is that the combination of cross-layer capabilities such as stealthy jamming using software radios, the inadequacy of wireless user interface mechanisms in popular commodity operating systems, and the insecure trust model used in wireless authentication makes end-to-end attacks against wireless network authentication feasible in practice.

It is clear that we need to explore new methods of authentication mechanisms that offer more than transmission confidentiality. As we propose a set of residential devices to provide network access services, the new mechanisms need to guard against attacks like the one we describe here, and also where Access Points are not trusted entities. Such is the subject of the next chapter.
Protecting hotspot user’s location

5.1 Introduction

Recent trends in offloading mobile traffic to Wi-Fi hotspots solve traffic load problems for providers at the cost of client identity and mobility patterns. Hotspots are largely implemented as unencrypted untrusted Access Points (AP) with a captive portal backend—often owned by other ISP subscribers. This setup allows a dishonest ISP to track the location of users as they connect to hotspots, and from it further deduce other sensitive information, as was shown in [120]. With the ongoing growth in mobile network access, extensive quantities of data on users’ mobility patterns are being generated, and few countermeasures exist to protect this leakage of private and sensitive information.

Under these conditions, some providers have reacted to protect location information. For instance, smartphone vendors (e.g., Apple iOS 8) have included MAC address randomization [14] to prevent some types of location tracking. However, mobility patterns can still be deduced from clients with dynamic addresses by the access points or hotspot connectivity providers.

While the problem of anonymous credentials has received a significant amount of attention for a long time, the literature does not cover the cases where an anonymous side channel is not available for rekeying, changes to group membership are common, with users entering and leaving the system constantly, and that protects against a server attempting to deanonymize clients.

In this work we propose TracEdge, an anonymous authentication protocol that hides user’s identity from both AP operators and authentication server. Our proposed scheme rekeys users by letting them re-
retrieve their access keys anonymously when needed, and as such removes the need for an anonymous side channel. Credentials in our protocol are not bound to the current set of authorized users, making membership changes a constant-time operation that does not require long-term user rekeying. In the case of dishonest providers, our protocol interaction gives users proof that a server has attempted to identify them, even before the authentication is complete. The proof of misbehavior can be disclosed and verified by any other third party, thereby exposing malicious servers’ activities. This capability serves as deterrent for dishonest servers, and it gives honest ones a mechanism to keep client trust.

TracEdge, as any security mechanism, does not come for free. It implies additional costs for the ISP, and client authentication time may increase. Still, there exists a strong incentive for providers to adopt such anonymizing mechanisms for two reasons: First, recent news on privacy breaches and wiretapping has created demand for anonymous communication on the customer side. Secondly, TracEdge’s costs are reasonably small as we will see later in Section 5.5.

5.1.1 Problem Statement

A service provider offers network access to users through a set of Access Points (e.g., Wi-Fi hotspots) connected to its system. Users subscribe to the service at any time by contacting the provider, and agreeing to some terms of service. By the end of the subscription process, the provider knows identifying information about the subscribed client. Likewise, clients may unsubscribe from service, ending their authorization to the system.

After subscription, the client associates to an AP in the provider’s network near to their current location and uses it to authenticate to the provider’s server. The server must decide whether the authenticating user is one of the subscribers. The user however, does not wish to leak their identity to either the ISP’s server or AP operator when connecting. Specifically, the authentication exchange between this user and the server must be indistinguishable from any authentication exchange for the server, AP, and any eavesdropper. In other words, we view both the access point and the authentication server as adversaries. We assume
the server to also be an active adversary in the sense that it can assign different keys to each user to deanonymize them.

We define an anonymous authentication protocol as an exchange between user and server that has the following properties. Let \( \text{UID} = \{ C_i : 1 \leq i \leq n \} \) be the set of \( n \) client identities known to provider \( P \).

1. Given an authentication exchange with client \( c \), the server can decide if \( c \) is an element of \( \text{UID} \).

2. For any server, access point, and external observer, authentication exchanges for some client \( C_i \) are indistinguishable from those for client \( C_j \), for all \( 1 \leq i, j \leq n \).

3. An authorized client \( C_i \) knows with a certain probability of anonymity \( P_a \) whether presenting her credentials will leak her identity to the provider.

4. If a malicious server can identify \( C_i \), the authentication exchange provides proof that it has targeted \( C_i \)’s identity.

We further constrain our scenario by noting users do not have access to an anonymous channel over which to interact with the server without revealing their identity. Such is the case when users build and destroy links to the server network over time.

To the best of our knowledge, TracEdge is the first anonymous authentication scheme that allows clients to detect identity leaks and has sub-linear communication complexity in the number of clients in a dynamic client membership context. Our underlying NTRU-based PIR makes key revocation immediate, with no communication cost, and with constant authentication key sizes.

This chapter is organized as follows. We present our anonymous authentication method TracEdge in Section 5.2. Section 5.3 shows our single-database Private Information Retrieval scheme. Section 5.4 describes our system of TracEdge as a Wi-Fi authentication method over 802.1X [86] Extensible Authentication Protocol for large databases. We present the evaluation of our scheme in Section 5.5.
5.1.2 Notation

We denote $K\{m\}$ as the symmetric-key encryption of message $m$ with key $K$. Public key encryption and decryption of $m$ using key $PK$ is written as $E_{PK}(m)$ and $D_{PK}(m)$ respectively. We use $E(m)$ and $D(m)$ for an homomorphic encryption scheme to distinguish it from the above. Signatures are noted as $\text{Sig}_{PK}(m)$. The concatenation of $a$ and $b$ is noted as $a \parallel b$, and the symbol $\bot$ represents an invalid key value.

5.2 Anonymity from Private Information Retrieval

As a consequence of the privacy mechanisms of this section, no traffic, access patterns, nor identities are disclosed to the server. We assume our adversaries to include servers that cannot be trusted to keep client data or identities private at time of authentication, but that can be partially trusted to protect access patterns.

Clients demand their individual privacy to be protected against tracing attacks, or attacks that link authentication events. For instance, an adversary eavesdropping on communication should not be able to tell whether a client seen previously is currently authenticating again at a specific server. No authentication server should be able to find a link between authentication events, as it can lead to identification of individuals [15, 120]. Not only does this require encryption of communication between the authenticating client and the server, but it also calls for protection against the server itself.

The authentication scheme proposed in this section is appropriate for access control of individuals or members of groups. The authentication will be anonymous to an eavesdropping adversary including the server itself. More formally, given a group of $n$ clients, and access to a simulator of authentication exchanges, an adversary is not able to guess the client identity for a given exchange with probability $P > \frac{1}{n} + \varepsilon$, where $\varepsilon$ is “negligibly” small, depending only on the security parameters of underlying cryptographic primitives. In addition, further executions of the protocol provide no advantage to the adversary. At the end of the authentication, the only information the server learns is whether the connecting client has proved membership to the group.
5.2.1 PIR Authentication

Algorithm 1 shows the basic authentication mechanism. We assume a Public Key encryption scheme $\mathcal{S} = (K, E(), D())$ exists and for which the server and users know $\mathcal{U}$, the set of subscriber’s public keys. To make keys publicly available, a simple, integrity protected public key directory like PGP keyservers [128], a public ledger such as the one in Namecoin [153], or a traditional trusted CA structure protecting identities may be used. We also assume every server has a public identity known to the clients, and that client certificates are marked to only be used on this scheme to prevent protocol composition. To authenticate a user, the server uses a random key $K$, for which it builds a table with rows $E_{A_{\text{pub}}}(K)$ for every user $A$ in $\mathcal{U}$. An authenticating user $A$ has to anonymously and efficiently retrieve the entry corresponding to their key. Then, once $K$ is recovered, they can use it to authenticate using a standard challenge-response mechanism.

The client $A$ can obtain the current key $K$ without revealing its identity by performing a PIR exchange [99] with the server. Given the table index $i_0$ where $A$’s entry resides, the client constructs a query vector $v$ using additive homomorphic encryption (detailed on Section 5.3.3). The server then computes the product of the vector with each entry in the database and adds the results to obtain a response for the client. For a detailed presentation of the new PIR scheme we use, see Section 5.3.

While this protocol hides the identity of the client effectively, a misbehaving server can still leak subscriber information by assigning a different key for different clients. For instance, the server may use two keys $K_1$ and $K_2$ for the same table, and build it encrypting $K_1$ with half the client public keys of clients, and $K_2$ with the others. Later mutual authentication would then reveal the group to which the client was placed. In the extreme case, a server may assign $n$ different keys in the table, one for each client, deanonymizing the group completely.

To mitigate this exposure, Algorithm 1 has the server commit to a value of the key at the start of the protocol by computing a fresh cryptographic signature of $K$ and sending this value to the client. After PIR, and once an authorized client has obtained the value of $K$, it can check the signature. If the check passes, the client can continue with the mutual authentication normally. If verification fails, the client performs
mutual authentication with an invalid key to keep the authentication exchange indistinguishable from an unauthorized client.

5.2.2 Improving Detection of Misbehaving ISPs

While a commitment to a key for the entire table allows clients with a value of $K$ different to the commitment to detect wrongdoing, clients with the same $K$ as the commitment however, can still leak a small amount of information when authenticating and need to rely on other users to detect the server misbehavior.

To mitigate this problem, a client who would like to increase his immediate confidence in the correct behavior of the server, can perform PIR retrievals of other clients’ entries and check whether the database holds the expected values. After retrieving their own $K$, the client retrieves $E_{\text{pub}}(K)$ from a sampling of $a$ different rows. As long as a deterministic algorithm is used to encrypt $K$, e.g. Elgamal with a “fixed” random coin, clients can verify the extra entries they have obtained, increasing their probability of finding a flagged key if one exists. In other words, on every table build, the server chooses a random $r$ to be used across all encryptions of $K$ and computes for all rows $i$

$$E_{\text{pub}}(K) = (rP, r_{\text{pub}}) = (rP, rY_i)$$

where $Y_i$ is user $i$’s public EC Elgamal key. Note that because $K$ is chosen uniformly at random, and is refreshed after a new table is ready to be used, there is no need to protect its encryption against chosen plaintext attacks.

5.2.3 Correctness and Privacy

Given a key $K$, the client public key for the client $A_{\text{PUB}}$ and a valid encryption $E_{A_{\text{PUB}}}(K)$, a mutual authentication proving knowledge of $K$ also proves $A$ belongs to the group of authenticated clients.

The privacy of TracEdge depends on two factors: the underlying PIR scheme, and the detection of flagged keys in the database if they exist. In secure PIR, any two queries exchanged are indistinguishable for the
server. Since the identity of the client is bound to an index in the table, a secure PIR will provide privacy at the query level to our scheme.

Because the client obtains a signed response for both the commitment to $K$ and the PIR result during the protocol, it can detect and prove to any third party when a server has sent a key different from the committed one, violating its client’s trust. A misbehaving server has to trade the amount of information he learns with the probability of being exposed. Assume that the server commits to key $K_1$ but sets $m$ entries in the table with key $K_2$. Table 6 summarizes the trade-off available to the misbehaving server. The server learns different amounts of information depending on whether the query matches the commitment. However, any time a user makes a query for an entry that doesn’t match the commitment, they detect the misbehavior of the server and can publicly expose it. While a misbehaving server learns a limited amount of information about the identity of a user, the probability of not being detected decreases exponentially as a function of the number of queries.

Let $DB = \{ E_{C_i}(K_i), 1 \leq i \leq n \}$ be a database of $n$ authorized clients, commitment $K^*$, and $m$ flagged keys (i.e., there exist $m$ indices $i$ where $K_i \neq K^*$.) A system receiving $a - 1$ queries to rows selected uniformly at random has a probability $p_{nd}$ of no client detecting a flagged key:

$$p_{nd} = \prod_{i=1}^{a-1} \left( 1 - \frac{m}{n - i} \right)$$

The maximum probability for a server to be undetected by a non-flagged client occurs when $m = 1$, in which case the minimal amount of information will be learned. The flagged client, on the other hand, will detect the flagging unconditionally, and will have proof the server has cheated.

Assuming queries are uniformly distributed in the set of users, the probability of being detected after $i$ queries is $1 - \left( 1 - \frac{m}{n} \right)^i$. To learn 1 bit about the identity of users, the misbehaving server will be exposed with probability $1 - \left( \frac{1}{2} \right)^i$ after $i$ queries. Given the typical number of queries and that when a server is exposed he loses the users’ trust irrevocably, a server cannot misbehave with impunity for any reasonable amount of time.
5.3 PRIVATE INFORMATION RETRIEVAL

In a PIR protocol, a client and a server storing $n$ records exchange messages such that by the end of the protocol the client learns the contents of a record of their choosing, while the server cannot guess which record was retrieved with probability greater than $1/n$. To achieve this, the server performs some operation related to the underlying encryption scheme over the database records. We describe a basic version of the scheme we use for TracEdge below.

5.3.1 Overview

In our scheme, the database is a set of $n$ records of length $\ell$ bits. Every row $i$ contains bits $DB_i = \{b_{i,j}, 1 \leq j \leq \ell\}$, and we use NTRUEncrypt $[84, 140]$ over the Ring $\mathbb{Z}_q / (X^N - 1)$ with polynomial multiplication $\ast$, and prime $N$. The use of NTRU allows for homomorphic addition $\mathcal{E}(m_1) + \mathcal{E}(m_2) = \mathcal{E}(m_1 + m_2)$, and an efficient query mechanism we call Multiple Row Selection (MRS) that we describe in Section 5.3.2. The main idea is to use the encryption of polynomials as the query component to select the desired row as well as rows within the same region. This gives the client the ability to audit multiple rows in the database to ensure he is not being targeted by the server (we describe this feature in Section 5.2.3). With a large enough polynomial degree $N$, the number
of expensive multiplications will be reduced by this factor. This stands in contrast to more traditional PIR in which one encryption selects a single row of the database.

The query-response protocol for the $i_0$-th database element is as follows (see Figure 30):

**QUERY GENERATION** The client computes a vector $v$ of $\lceil n/N \rceil$ encrypted polynomials of degree $N$, where all but one is an encryption of the zero polynomial. To query the database for element $i_0$, the client encrypts the polynomial $x^{i_0} \mod N$ and assigns it to the $\lceil i_0/N \rceil$-th element in $v$.

**RESPONSE GENERATION** To generate a reply, the server divides the database in regions of $N$ rows, grouping bits in the same region's column as a polynomial $e_{i,j} = \sum_{i=0}^{N-1} b_{i,j} x^i$ and computes $e'_{i,j} = v_i \ast e_{i,j}$. The meaning of $\ast$ depends on the additive homomorphism of $E(\cdot)$, which for NTRU is polynomial multiplication modulo $X^N - 1$ as described in Section 5.3.3).

The server then computes a result vector $r$ by adding the resulting entries $e'$ column-wise, giving $r_j = \sum_i e'_{i,j}$, which is then sent to the client.

**DECODING** The client decrypts the reply vector and extracts the information using the homomorphic properties of the underlying encryption scheme $E$. For additive homomorphic encryption

$$E(a + b) = E(a) + E(b)$$

the elements of $r$ can be written as

$$r_j = \sum_{i \neq i_0} e_{i,j} \ast E(o) + e_{i_0,j_0} \ast E(x^{i_0})$$

$$= E(e_{i_0,j_0} \ast x^{i_0})$$

Note that the values in the databases need not be in plaintext or even encrypted with the same $E(\cdot)$. As we discuss in the following section, homomorphic addition and our use of polynomial multiplication allows us to retrieve the selected row. Further, the communication com-
plexity of our scheme can be improved by using Kushilevitz’s PIR [99]. This technique reduces communication complexity to $O(\sqrt{n})$. While it can be generalized for $O(n^{1/d})$, we show in Section 5.5 that squaring gives enough savings for short communication time.

5.3.2 NTRU-PIR Multiple Row Selection

An NTRU-PIR query with Multiple Row Selection takes advantage of the properties of polynomial multiplication. For any polynomial $y = \sum_{i=0}^{N-1} y_i x^i$ in the Ring $\mathbb{Z}_q [X] / (X^N - 1)$, the product

$$x^{i_0} \ast y \mod (X^N - 1) = \sum_{i=0}^{N-1} y_{i+i_0} \mod N x^{i+i_0} \mod N$$

That is, the coefficients of $y$ are rotated by $i_0$ positions. When the NTRU-PIR operation multiplies the query vector with every column region in the table, the resulting operation is an NTRU encryption of either the zero polynomial for regions where the query vector $v$ was zero, or a rotation of the query region, where $v$ contained $x^{i_0}$. In fact, setting any single coefficient in a component of the query vector will result in a rotation of the column returned.

This is a useful mechanism for two reasons. First, it integrates well with our protocol’s multiple row audit feature, giving users protection against a malicious server. Second, the number of multiplications during the computation of the PIR is reduced by a factor of $N$.

5.3.3 NTRU Encryption

In this section we describe the properties of NTRU encryption and the modifications we implemented to make it suitable for PIR.

NTRU Encrypt is a public-key encryption scheme that operates on the ring of integer polynomials $\mathbb{Z}_q [X] / (X^N - 1)$ where $\mathbb{Z}_q$ is the ring of integers modulo $q$, a small power of two, and $N$ a prime. The element $p$ is a small polynomial relatively prime to $q$, usually $X - 2$ or 3. To support homomorphic additions without decryption failure, we select a suitable value of $q$ empirically in Section 5.5.
Multiplication in the ring is defined as the product of polynomials modulo \(X^N - 1\). If \(f\) and \(g\) are polynomials in the ring, then their product \(f \ast g = \sum_{i=0}^{N-1} h_i x^i\) where

\[ h_k = \sum_{i+j=k \mod N} f_i \cdot g_j \]

The private key \(f\) is chosen at random with the standard [152] recommending coefficients in the range \([-1, 1]\). The public key is \(h = p \ast f_q \ast g \mod q\) where \(f_q\) is the inverse of the secret key in \(\mathbb{Z}_q[X]/(X^N - 1)\).

The encryption of message polynomial \(P_m\) is

\[ c = r \ast h + P_m \mod q, \]

where \(r\) is a polynomial chosen at random modulo \(q\).

Decryption consists of two steps. First the value \(a = f \ast c \mod q\) is computed. Secondly, \(P_m = f_p \ast a \mod p\) is obtained to reveal the plaintext, where \(f_p\) is the inverse modulo \(p\) of the secret key.

When performing the response generation phase of our PIR protocol, every bit region \(e_{i,j}\) of the database will be multiplied with the query polynomial for that region. The result vector for column \(j\) is

\[ r_j = \sum_{i \neq i_o} E(0) \ast e_{i,j} + E(x^{i_o}) \ast e_{i_o,j_o} \]

so to extract the response, we note that multiplication \(g \ast x^{i_o}\) modulo \((X^N - 1)\) rotates the coefficients of \(g\) cyclically by \(i_o\) positions. In particular, coefficient \(k\) of \(g\) is

\[ (g \ast x^{i_o})_k = g_{k-i_o} \mod N \]

therefore, to calculate the bit associated with column \(r_j\) the client extracts the \(2 \cdot i_o \mod N\)-th bit of \(D(r_j)\).

### 5.3.4 Parallel NTRU-PIR

The PIR computation itself consists of expensive polynomial multiplications and the addition of columns into the result value. Since we intend to deploy this system on GPUs, we need to parallelize the NTRU-
PIR as much as possible. For this, we map the polynomial multiplications into point multiplications of the corresponding Fast Fourier Transforms (FFT). We sum over the whole table before applying the inverse FFT. There are two distinct advantages to this. First, it reduces the complexity of multiplications to $O(N \log N)$ on the degree of the polynomials, instead of $N^2$. Second, the component-wise multiplication of the coefficients in the frequency domain is an operation much more susceptible to parallel computation, as every result coefficient depends only on a single complex multiplication.

Addition the column polynomials in parallel treats polynomials as leaves in a binary tree, with every addition step removing children and writing the result in the parent. Because the value of every non-leaf node depends on the addition of its children there is a loss of roughly half the computation power on average. At the last step, there is a single process computing the addition of the last set of elements. However, because our elements are polynomials of degree $N$, every coefficient can be computed in parallel, even to the last addition.

5.4 System

We implement TracEdge as an extension to the Protected EAP [86], using the same TLS tunnel establishment in the common Wi-Fi WPA-Enterprise framework. This ensures the authentication server has a set of authentic public and private keys that can be verified with a Certification Authority. We describe the implemented protocol components and also offer a basic summary of the operation of 802.1X. In addition, usage of the TLS tunnel provides lower-layer fragmentation and reassembly of large (longer than 1020-byte) packets for the underlying method.

5.4.1 The EAP-TE method

Figure 31 shows the architecture of EAP-TE. After the client first associates to the AP, it starts a TLS tunnel with the Authentication Server, where standard checks are performed to ensure the identity of the server. After tunnel establishment is complete, both client and server share a session key $K_{CS}$ to encrypt all later communication. This protects
the conversation against eavesdropping. After this point, all protocol messages are encapsulated within EAPOL frames on the air, and re-encapsulated as RADIUS Access-Request/Access-Challenge packets on the back-haul. The authenticator initiates by sending an initial EAP-Request packet with an unused protocol identifier.

The first phase of the **EAP-TE** starts by sending an encoded timestamp as the identity to the authenticator in response to the first EAP-Request packet. Subsequent packets containing the commitment, PIR Query, PIR Response and key are exchanged similarly. The final mutual authentication we selected is a simple challenge-response for both parties using SHA-256 as the cryptographic hashing function. The final session keys are built by hashing the initial timestamp, the key value and the strings `server` and `client`.

![Architecture of the EAP-TE protocol](image)

**Figure 31**: Architecture of the **EAP-TE** protocol.

### 5.4.2 The Supplicant

Our client query generation is a wpa-supplicant [107] patch for Linux and Android systems. Query generation on the client involves comput-
ing \( n/N \) NTRU encryptions in polynomials. The size of the query is \( n \times |q| \) bits, or \(|q|\) bits per database entry. Using Kushilevitz and Ostrovsky’s [99] PIR allows us to reduce this to \(|q|\sqrt{n}\).

An important client parameter is its row in the database. We assume that every client obtains their fixed database index at the time of registration with the Service Provider. As noted in Section 5.1.1, the server may know identifying information about the client, including its fixed row number, but the PIR protocol prevents the server from learning it at authentication time.

### 5.4.3 The Authentication Server

The authentication server in **EAP-TE** provides two important functions in the system. First, it must generate the encrypted key tables that will be used during authentication. Because the server cannot know which row the clients may be accessing, it must be careful not to keep invalid entries even for rows with unassigned clients. For unassigned rows, it may encrypt the key with its own public key, which must be made available to the clients.

The PIR Reply generation is the most expensive operation in the protocol. It is potentially parallelizable, where computation of the result is divided among a large number of threads. We evaluate the performance gains from GPU parallelization in Section 5.5.

As clients enter and leave the system, index management must keep the invariance of the client entry index. Empty database rows are kept with a special encryption of \( K \) with the server’s public key. This allows auditing clients to check validity of the committed \( K \), while protecting the key itself from discovery. When a new user subscribes, the server chooses one of the empty rows and assigns it to the client. Later, when a client unsubscribes, the association is cleared and the row value is replaced with the special value from above.

### 5.4.4 Optimizations

Several expensive operations occur in **EAP-TE**. On the client side, vectors with \( \sqrt{n} \) elements with mostly \( \mathcal{O} \) are generated to form the
PIR query. While NTRU Encrypt is a fast operation compared to other number-theoretic schemes, polynomial multiplication is still intensive enough to add considerable time for large numbers of users if done naively. We show the impact of the implementation on the speed of polynomial multiplication on Section 5.5. Because no polynomial in the vector depends on others this an easily parallelizable operation. Moreover, pre-computation of zero-polynomials during idle times can aid in offsetting this cost in the handset.

While authentication key $K$ must be refreshed to both prevent unauthorized users from reusing keys, and to revoke access to unsubscribing clients, there is no need to recompute the table after every authentication. The authentication server can update its table periodically (e.g., every hour) resulting in changes being batched and becoming effective with a reasonable delay similar to today’s ISP delay in activating/canceling users accounts. Another strategy for optimization is to keep the public-key bit length small. Since $K$ is short-lived and it is discarded after new tables come to the pool, it is not necessary to use a large security parameter.

Resolving a PIR query has the server compute a value over every column of the table, so limiting the size of the key implies computation savings. We can obtain even larger savings by using a cryptosystem with small key sizes, like ECC. Table generation can be quite costly for the large number of clients we consider. For this construction we use a version of ECC-Elgamal encryption for every client entry with pre-computed ephemeral keys. This optimization pre-computes the value $H(K) \cdot P$ and uses it to build every entry in the table, i.e.,

$$(H(K) \cdot P, H(K) \cdot x \cdot P + K)$$

where $x \cdot P$ is the user’s public key. This halves the number of EC scalar multiplications.

Our system uses the parallel NTRU-PIR described in Section 5.3.4. A practical trade-off of this technique that data transfers between the host memory and the GPU device’s RAM must occur before the operations start. Fortunately, these transfers can be pipelined while the processors work on previously loaded data. Another trade-off is a more involved implementation that must deal with processor occupancy and
GPU memory access times carefully such that they do not offset the speed gains of running the code in parallel. We show the performance of our GPU implementation on Section 5.5.

Our implementation using Fast-Fourier transform-based polynomial multiplication over GPUs using CUDA. Every column region in the database is stored in FFT form after table generation, as is the query vector received by the client. The table is split column-wise among GPU devices to fill the device's available memory. Computation starts with component-wise multiplication of the FFT forms of the query vector and every column. Next every column region is added to obtain a final degree \( N \) polynomial in FFT form that is then transformed back into the time domain.

Addition is a two-step process to maximize the number of parallel operations. First, \( \beta \) CUDA blocks add two polynomials at a time in shared memory, producing \( \beta \) polynomials. Finally a single block adds the former to get the result for the column. We find the optimum value of the parameter \( \beta \) on Section 5.5.

5.5 Evaluation Results

We evaluate the efficiency of a TracEdge construction by benchmarking its components. For the server components we use a combination of off-the-shelf desktop machines with medium-range GPUs and production-grade Amazon EC2 instance. Clients are Android-based smartphones.

5.5.1 Evaluation Parameters

We choose our security parameters assuming a key lifespan of a few hours and an AP density consistent with a modern urban ISP hotspot deployment. We test the client query upload time assuming the wireless link to be the bottleneck in this operation. Our measurements for database building are done using two ECC curves: the smallest and the fastest of the publicly available \([36, 37]\). To test PIR result computations on the server we measure the time elapsed in computing the client's response from a query. Because this is the most computationally expensive operation we attempt to characterize the behavior of the basic
operations in two implementations: CPU and GPU. For system specifications see Table 9.

Table 7 summarizes the evaluation parameters for EAP-TE. We choose a 128-bit key for $K$ that fits within the EC Elgamal modulus size, and gives enough entropy to generate suitable WPA encryption keys. A small modulus size limits the width of the table, reducing the time for the PIR response generation. Conversely, a faster curve reduces table generation but impacts PIR response due to wider row length. While these may seem modest security parameters, we assume the lifetime of $K$ to be in the order of hours, as the server periodically generates new tables. Thus, we limit table size, generation time, and key lifetime.

We chose our NTRU modulus parameter $q$ by empirically testing the minimum number of additions observed before an encryption failure occurs. For every value of $q$, we add a ciphertext addition of $E(0)$ to $E(1)$. After every addition, we check whether the decryption returns the original value. We record the minimum number of additions observed over $10^8$ repetitions. Table 8 summarizes our results.

5.5.2 Database Generation and Communication Complexity

Given the parameters in Table 7 for TracEdge, the maximum size of the table using EC Elgamal over 163-bit security is $41 \times 10^7$ bytes, or 410MB. Our server code builds a database table by choosing a key $K$ uniformly at random, and using $H(K)$ as the scalar multiplier for the ephemeral Elgamal key. We take the mean time to create a table over 100 samples on a multi-threaded implementation, shown in Table 10.

Query generation in the client happens every time the client wishes to authenticate to the server. A query for our PIR scheme is a vector of $q \times \sqrt{n} = 4\sqrt{10^7}$ or 12.6KB, assuming 32-bit integers to store coefficients. We benchmarked our client’s query generation over 1000 repetitions and Table 11 summarizes the results.

Communication complexity describes the amount of data needing to be sent over the air. The scale of our client database makes it a good candidate to use the technique in [99]. By processing the table as a square with $\sqrt{n}$ rows and columns over two iterations, the server needs to send back $N\sqrt{\ell\sqrt{n}}$ elements back, containing the encryption of a database
element. Using 32-bit integers this yields a total of 1.25MB for the response. Assuming a 10Mbps data rate, this transmission completes in 1.0 seconds.

We measure rate quality over the air using 50 public Wi-Fi hotspots from a large ISP in an U.S. urban area. We associate our mobile client devices from Table 9 to the target APs and connect to a test server at a well-provisioned site. Afterwards we make a similar measurement with Speedtest [124] as a control. We measure downlink and uplink throughput over a period of 10 seconds. Figure 32 shows how over 50% of the sampled APs can sustain throughputs higher than 10 Mbps over 10 seconds.

![Hotspot Link throughput](image)

Figure 32: Throughput observed over ISP public hotspots in urban area

Response extraction at the client is performed by decrypting all the polynomials received from the server. Table 12 shows average times for our clients over 1000 samples.
5.5.3 Response Computation

The most computationally intense stage in TracEdge is response generation, which consists of two phases: polynomial multiplication modulo $X^N - 1$, and polynomial addition modulo $q$. Even though polynomial multiplication is a fast operation compared to other homomorphic cryptographic schemes, the large size of the database can make computation lengthy. Fortunately, the operation is suitable for parallelization, as discussed in Section 5.3.4. We evaluate two computation strategies: multi-threading over CPU cores and GPU computation using CUDA. Table 13 shows the average operation time for each server per database column on a straightforward polynomial multiplication and addition implementation.

GPU computations clearly outperform the threaded program as can be expected, and also scale well on the number of cores and devices. For the case of Server 6, a set of 8 mid-range GPUs (2300-core) could compute the response of 10 million rows in little more than 1.1 seconds key table. However, evaluation times still fall in the order of seconds due to the nature of the $O(N^2)$ multiplication algorithm used.

In contrast, using the Fast Fourier Transform strategy outlined in Section 5.4.4 significantly reduces the evaluation time per column (over an order of magnitude gain). Figures 33a and 33b show the average time taken per column multiplication and addition over 500 samples using different block/threading configurations. Every curve represents a block configuration for the first addition step, while the time shown includes both addition steps. From our results we find each device has different optimum block parameter (as described in Section 5.4.4) with $\beta = 64$ for the Amazon instance and $\beta = 96$ for the GTX780. The mid-range GTX780 device takes 1.077ms per-column computation for the FFT multiplication and addition versus 28ms with the former implementation. This implies an array of 8 GTX780 GPUs can compute the PIR response of 10 million 326-bit wide entries in 43.9ms.

Such improvement is due to the increased parallelism induced by the point-to-point product in the frequency domain. Every result coefficient depends only on a pair of factor coefficients, avoiding potential slowdowns from threads accessing the same memory position or non-coalesced memory access.
With the current landscape of Wi-Fi access being such that network access for mobile clients exposes their identity and location to ISPs and AP hosts, it becomes necessary to provide methods to protect the leaked information. A method for network access that protects users privacy becomes critical in our model, where users offering access are not trusted by the client.

With TracEdge we have shown that such a mechanism can provide protection for users efficiently, with authentication times of less than 44ms, with the added benefit of dishonest server detection. While the authentication overhead in terms of communication complexity is slightly larger than traditional methods, it brings an improvement over key sizes and key distribution without an anonymous key distribution channel.
Algorithm 1: **TracEdge** authentication.

**Data:** *audit*, the number of rows to check for flags, $h = g^y$ the server’s public key

1. $C \rightarrow S : t_s$
2. $C \leftarrow S : \text{Sig}_{SPR}(g^{kh_2} \parallel t_s) //server commits to an encryption of $K$ with random $r$ for all clients
3. $C \rightarrow S : \text{PIR} \_Q(A)$
4. $C \leftarrow S : \text{PIR} \_\text{Resp}(E_{A_{pub}}(K'), K \{r\})$
5. $\text{Sig}_{SPR}(\text{PIR} \_\text{Resp}(E_{A_{pub}}(K') || K \{r\} || t_s))$

$C : \text{Check signature, compute}$

1. $K' = \text{PIR} \_\text{Extract}(\text{PIR} \_\text{Resp}(E_{A_{pub}}(K') )$
2. $r' \leftarrow \text{Decrypt}_{K'}(K \{r\})$

$C :$ if $g^{kh_2}r' \neq g^{kh_2}$ then

1. $K \leftarrow \bot$ and report server

while $audit > 0$ do

1. $C : \text{row} \leftarrow \{1 \ldots n\} \setminus A$
2. $C \rightarrow S : \text{PIR} \_Q(\text{row})$
3. $C \leftarrow S :$
4. $\text{PIR} \_\text{Resp}(E_{row_{pub}}(K'' ))$
5. $\text{Sig}_{SPR}(\text{PIR} \_\text{Resp}(E_{row_{pub}}(K'' ) || t_s))$

$C : \text{Checks signature, computes}$

1. $X = \text{PIR} \_\text{Extract}(\text{PIR} \_\text{Resp}(E_{row_{pub}}(K'' )))$
2. $C : audit \leftarrow audit - 1$

$C :$ if $E_{row_{pub}}(K') \neq X$ then

1. $K \leftarrow \bot$ and report server

end

$C \leftrightarrow S :$ Mutual Authentication Protocol using $K$

---

Table 6: Identity Leakage vs. Misbehaving Server Detection

<table>
<thead>
<tr>
<th></th>
<th>Commit. matches</th>
<th>No matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>Srv. learns</td>
<td>$\log \left( \frac{n}{n-m} \right)$ bits</td>
<td>$\log \left( \frac{n}{m} \right)$ bits</td>
</tr>
<tr>
<td>Srv. exposed?</td>
<td>No, prob. $\frac{n-m}{n}$</td>
<td>Yes, prob. $\frac{m}{n}$</td>
</tr>
</tbody>
</table>
Table 7: Implementation choices

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>K</td>
</tr>
<tr>
<td>$E_{\text{PUB}}(K)$</td>
<td>EC Elgamal [57]</td>
</tr>
<tr>
<td>EC Curves</td>
<td>sect131r1, sect163k1</td>
</tr>
<tr>
<td>$H(\cdot)$</td>
<td>SHA-256</td>
</tr>
<tr>
<td>$E(\cdot)$</td>
<td>NTRU-Encrypt [84]</td>
</tr>
<tr>
<td>NTRU Parameters</td>
<td>APR2011_439_FAST [152]</td>
</tr>
<tr>
<td>$N$</td>
<td>439</td>
</tr>
<tr>
<td>$q$</td>
<td>$2^{21}$</td>
</tr>
<tr>
<td>$n$</td>
<td>$10^7$ clients</td>
</tr>
</tbody>
</table>

Table 8: Choice of $q$

<table>
<thead>
<tr>
<th>$q$</th>
<th>Observed Additions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^{11}$</td>
<td>2</td>
</tr>
<tr>
<td>$2^{12}$</td>
<td>47</td>
</tr>
<tr>
<td>$2^{13}$</td>
<td>250</td>
</tr>
<tr>
<td>$2^{14}$</td>
<td>951</td>
</tr>
<tr>
<td>$2^{15}$</td>
<td>4408</td>
</tr>
<tr>
<td>$2^{16}$</td>
<td>18087</td>
</tr>
<tr>
<td>$2^{17}$</td>
<td>73047</td>
</tr>
<tr>
<td>$2^{18}$</td>
<td>367865</td>
</tr>
<tr>
<td>$2^{19}$</td>
<td>$1.12 \times 10^6$</td>
</tr>
<tr>
<td>$2^{20}$</td>
<td>$7.01 \times 10^6$</td>
</tr>
<tr>
<td>$2^{21}$</td>
<td>$2.21 \times 10^7$</td>
</tr>
</tbody>
</table>

Table 9: Evaluation systems

<table>
<thead>
<tr>
<th>Srv.</th>
<th>Hardware Setup</th>
<th>Graphics/WLAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Quad 3.0GHz, 8GB</td>
<td>9800GT (112c)</td>
</tr>
<tr>
<td>2</td>
<td>Quad 2.5GHz, 8GB</td>
<td>GTX280 (240c)</td>
</tr>
<tr>
<td>3</td>
<td>2x 8c. Xeon, 130GB</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>8x Xeon, 21GB</td>
<td>2x GF100 (448c)</td>
</tr>
<tr>
<td>5</td>
<td>8x Xeon, 21GB</td>
<td>1x GK104 (1536c)</td>
</tr>
<tr>
<td>6</td>
<td>Quad 2.5GHz, 8GB</td>
<td>GTX780 (2304c)</td>
</tr>
<tr>
<td>Cli. 1</td>
<td>Galaxy SII 1GB RAM</td>
<td>802.11bgn</td>
</tr>
<tr>
<td>Cli. 2</td>
<td>HTC One 2GB RAM</td>
<td>802.11abgn</td>
</tr>
</tbody>
</table>
Table 10: Table creation time (s) ± σ

<table>
<thead>
<tr>
<th>Server</th>
<th>sect163k1</th>
<th>sect131r1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1160 ± 4</td>
<td>3424 ± 9</td>
</tr>
<tr>
<td>2</td>
<td>1190 ± 9</td>
<td>3211 ± 13</td>
</tr>
<tr>
<td>3</td>
<td>261 ± 7</td>
<td>751 ± 12</td>
</tr>
<tr>
<td>4</td>
<td>11055 ± 3</td>
<td>2997 ± 10</td>
</tr>
</tbody>
</table>

Table 11: Query generation on client

<table>
<thead>
<tr>
<th>Device</th>
<th>Avg. Encryption (µs)±σ</th>
<th>Total time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>159.3 ± 0.12</td>
<td>1.86s</td>
</tr>
<tr>
<td>2</td>
<td>173.4 ± 0.08</td>
<td>1.98s</td>
</tr>
</tbody>
</table>

Table 12: Query extraction on client

<table>
<thead>
<tr>
<th>Device</th>
<th>Av. Decryption (µs)±σ</th>
<th>Total time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>63.7 ± 0.20</td>
<td>45.3</td>
</tr>
<tr>
<td>2</td>
<td>78.6 ± 0.13</td>
<td>55.9</td>
</tr>
</tbody>
</table>

Table 13: Unoptimized Polynomial Operations with degree \( N = 439 \) (times in µs)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4.28</td>
<td>251.6 ± 0.33</td>
<td>76.5</td>
<td>438s</td>
</tr>
<tr>
<td>2</td>
<td>2.67</td>
<td>228.2 ± 0.38</td>
<td>38.4</td>
<td>225ms</td>
</tr>
<tr>
<td>3</td>
<td>2.94</td>
<td>129.9 ± 0.05</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>2.97</td>
<td>130.2 ± 0.1</td>
<td>13.4</td>
<td>113.5ms</td>
</tr>
<tr>
<td>5</td>
<td>2.97</td>
<td>130.2 ± 0.1</td>
<td>5.9</td>
<td>48.8ms</td>
</tr>
<tr>
<td>6</td>
<td>2.67</td>
<td>228.2 ± 0.38</td>
<td>5.7</td>
<td>27.9ms</td>
</tr>
</tbody>
</table>
Conclusion

6.1 Feasibility

We have presented a testbed based on off-the-shelf hardware to gather residential data, and a throughput study of feasibility for peer data delivery, useful in the context of peer anonymity network. Since February 2011, we deployed 30 Access Points over residences around the United States, covering around 100 users, and collecting 53GB of link usage data over a period of 4 years. Using wireless home routers as the storage unit makes available an always-on device with moderate computing power and storage capacity.

- We study uplink usage patterns, flow volume, duration, and object sizes. Our analyses indicate residential broadband services are idle at least 30% of time and up to 80% for certain periods of the day. Average throughput is less than 100 Kbps at least 60% of the time.

- We run live experiments on our testbed simulating content delivery to better understand the impact, and achievable levels of service when providing content. We show experimentally that with a proper traffic shaping configuration, uplink bandwidth can be shared fairly and without impact even while serving content and up to 95% of uplink 90% of the time over a simulated onion routing scenario.

We looked at the behavior of user’s uplink utilization and found that idle links are both widespread, and—due to our uplink sharing tests—robust to sharing provided a sensible traffic shaping mechanism is in
place. Moreover, our test give evidence that home devices are able to sustain long periods of saturated uplink without disrupting normal user traffic. Given the performance a small base of small home devices can achieve, it is feasible to build data delivery system in this scenario.

6.2 ATTACKING ’SECURE’ WI-FI INSTALLATIONS

We made the following contributions in studying the security of Enterprise Wi-Fi installations:

- We present a practical, end-to-end, stealthy, and targeted evil twin attack against WPA Enterprise networks. The attack leverages novel, specific weaknesses in the human-computer interfaces of commodity operating systems for managing wireless connections that have not heretofore been discussed in literature.

- We are the first to demonstrate a significant weakness that exists in modern wireless authentication systems today – namely, that authentication server certificates are not strongly bound to network SSIDs. Using this fact, an attacker can use selective jamming techniques to trick unsuspecting users into connecting to a rogue access point without receiving an invalid certificate warning. Note that certificates are widely believed to be the most effective form of protection against evil twin attacks by practitioners [162].

- We describe a prototype implementation of the attack, present experiments with real users that demonstrate that the attack is feasible and effective in practice, analyze its cost, and discuss countermeasures that should be adopted.

We have shown a novel wireless attack against WPA Enterprise networks. The key insight behind our attack is that the combination of cross-layer capabilities such as stealthy jamming using software radios, the inadequacy of wireless user interface mechanisms in popular commodity operating systems, and the insecure trust model used in wireless authentication makes end-to-end attacks against wireless network authentication feasible in practice. Our user experiments demonstrate
that the attack is highly effective in practice and very difficult for vic-
tims to detect. We are the first to show significant deficiencies in wire-
less management user interfaces for commodity operating systems, and
also the first to highlight the weak binding between wireless network
SSIDs and authentication server certificates. We described a prototype
implementation of the attack, analyzed its effectiveness and cost, and
discussed countermeasures that should be adopted.

6.3 PRIVATE WI-FI AUTHENTICATION

We design an anonymous authentication scheme as defined above em-
ploying privacy-preserving techniques as building blocks, and evaluate
its performance. Our contributions are:

- A PIR-based authentication protocol augmented with a mecha-
nism to detect dishonest servers. This protocol allows a user to
prove that he is authorized to access the network without leaking
any information about its identity.

- A new underlying NTRU-based PIR technique we call Multiple
Row Selection (MRS) that allows the retrieval of $N$ rows with the
use of a single polynomial of degree $N$. This mechanism reduces
the number of multiplications on the server, is fast to evaluate,
and is easily parallelizable. This is a contribution of independent
interest, and it may be used as the PIR mechanism for other ap-
plications.

- We design and implement a Wi-Fi construction for TracEdge
compatible with the Wi-Fi Extensible Authentication Protocol
(EAP). Our implementation scales to databases in the order of $10^7$
clients. Our first optimization using GPU computations shows
client authentication times of 43.9ms.

We have presented an authentication protocol that protects client
identity, mitigates location tracing for general applications, and is able
to detect dishonest servers attempting to identify clients. In addition,
we show a proposal for a 802.11 implementation over EAP, and an es-
timation of its computational and communication costs. The key PIR
component can perform response computation in 43.9 milliseconds on a platform with 8 medium-range 1 year-old GTX780 GPUs. With devices in the market with over 5700 cores, computation times can be reduced further by augmenting hardware capabilities.

The nature of the underlying PIR multiplications and additions in our system, have an impact on our implementation in terms of memory access, and thread occupancy. Multiplications of complex numbers in the GPUs are fast, but memory accesses swap threads out in favor of others that have their data ready. Depending on the implementation and execution parameters, a GPU may end up waiting a long time without ready threads to execute, offsetting the advantage of parallel execution. Addition of polynomials on the other hand is less friendly to parallelization because the accumulated result becomes the bottleneck of the operation. To deal with this problem we use shared thread memory to perform the addition in a tree fashion, giving large savings in computation time. However, a more sophisticated implementation could make better use of the GPU’s shared memory and access patterns to further reduce PIR computation time.

While the table creation time is a relatively long operation due to its size, it does not need to be performed on the fly. The server may compute tables proactively and store them for later PIR consumption. Client keys may live in the database for several hours before being refreshed with a new table, a common timescale in ISP data update response time.

While there is still a probability of information leakage in this protocol if the server is willing to be discovered, we contend that the cost for the server to be discovered targeting clients is larger than the benefit obtained from identifying the client. This the case when trust is bound to the server’s business model, such as online banking, or identity certification for SSL. While this may not obviously be the case for mobile network servers and anonymity, allowing for clients to prove being targets provides a strong incentive for servers to be honest.

Prevention of user credentials pooling and sharing can be achieved with an implementation of TracEdge with a USIM component [1, 62, 95] and distance bounding protocols [23, 25, 32, 132, 135]. Currently, the EAP-AKA method [88] uses the subscriber module to authenticate to a network. Similarly, a TracEdge implementation would store the retrieved authentication key and use it to produce the necessary keying
material as in Section 5.4, preventing human access to this value. Distance bounding protocols such as in [25, 132] ensure that the distance between client and access point is kept within Wi-Fi range.

Although our system uses decentralized PKI, the protocol is not tied to it, and can easily be implemented using the usual CA model. However, it is worth noting that recent news [79] involving trust of actors in the networking and privacy ecosystem have sparked increase in its adoption. There is evidence of a significantly growing number of keys available in public servers [123] as well as TOR nodes following these events [157]. Because our scheme only needs a public list of certificates and identities, either model could in principle be used. Also, while the use of dedicated key pairs for authentication implies users need another piece of data just to perform this authentication, there is nothing in our scheme preventing automatic key generation when subscribing to the service and management by the application.
Bibliography


[97] Fred Kilbourn and Jakob Eriksson. “Social WiFi - Leveraging Social Networks for Safe Wi-Fi Sharing”. In: MobiCom Demo ’10. ACM.


