Authenticating Privately over Public Wi-Fi Hotspots

Aldo Cassola
U. San Francisco de Quito
Northeastern University
acassola@usfq.edu.ec

Erik-Oliver Blass
Airbus Group Innovations
Northeastern University
erik-oliver.blass@eads.net

Guevara Noubir
Northeastern University
noubir@ccs.neu.edu

Abstract

Wi-Fi connectivity using open hotspots hosted on untrusted Access Points (APs) has been a staple of mobile network deployments for many years as mobile providers seek to offload smartphone traffic to Wi-Fi. Currently, the available hotspot solutions allow for mobility patterns and client identities to be monitored by the parties hosting the APs as well as by the underlying service provider. We propose a protocol and system that allows a service provider to authenticate its clients, and hides the client identity from both AP and service provider at the time of authentication. Particularly, the client is guaranteed that either the provider cannot do better than to guess their identity randomly or they obtain proof that the provider is trying to reveal their identity by using different keys. Our protocol is based on Private Information Retrieval (PIR) with an augmented cheating detection mechanism based on our extensions to the NTRU encryption scheme. The somewhat-homomorphic encryption makes auditing of multiple rows in a single query possible, and optimizes PIR for highly parallel GPU computations with the use of the Fast Fourier Transform (FFT).

In this work we lay out the operation of our protocol in detail, its security analysis, and propose an implementation compatible with the Wi-Fi Extensible Authentication Protocol (EAP) along with optimizations for deployments of over 10 million clients. We evaluate the performance of its mobile and provider components, and show that a client can be authenticated in 43.9 milliseconds on a GPU platform, giving an end-to-end authentication of 1.12 seconds.

1. INTRODUCTION

Recent trends in offloading mobile traffic to Wi-Fi hotspots solve traffic congestion issues for providers at the cost of client identity and mobility patterns. Hotspots are largely deployed as unencrypted untrusted Access Points (AP) with a captive portal backend—often owned and operated by other ISP subscribers. In this scenario a dishonest ISP may track which APs a client connects to and when, revealing client's mobility patterns and from them other sensitive information, as was shown in [40]. With the ongoing growth in mobile network access, extensive quantities of data on clients' mobility patterns are being generated, and few countermeasures exist to protect this leakage of private and sensitive information.

Under these conditions, some providers reacted to protect device identity information. For instance, smartphone vendors (e.g., Apple iOS 8) included MAC address randomization [3] to prevent some types of device tracking. However, mobility patterns can still be deduced from clients with dynamic addresses by the mobile provider by tracking client ID access on APs over time. Alternatively, one can imagine using a pass-through authentication scheme such as UMTS' EAP-AKA [31] (or it's GSM variant EAP-SIM) to prevent the AP from tracking users. However, this does not prevent the cellular operator from tracking the users mobility patterns.

While the problem of anonymity and particularly 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 re-keying, changes to group membership are common, with clients entering and leaving the system constantly, and that protects against a server attempting to deanonymize clients.

In this work, we propose an anonymous authentication scheme named TracEdge that hides client's identity from both AP operators and authentication server. Our proposed scheme re-keys clients by letting them retrieve a different access keys anonymously when needed, removing the need for an anonymous side channel. Credentials in TracEdge are not bound to the current set of authorized clients, making membership changes a constant-time operation that does not require long-term client re-keying. In the case of dishonest providers, our protocol interaction gives clients 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, is not free. It adds computational costs for the ISP, and client authentication time may increase. Privacy-conscious providers however have a strong incentive to adopt such anonymizing mechanisms for two reasons: First, honest service providers may find that the information their systems collect can become a liability, and that it is in their best interest to keep client information private on technical unavailability grounds. For instance, Lavabit [48], a secure email provider, found itself closing shop, rather than releasing client information. Secondly, TracEdge's costs are reasonabe as we will see in Section 6.

1.1 Problem Statement

A service provider offers network access to clients 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 client is one of the subscribers. The client however, does not wish to leak their identity to either the ISP's server or AP operator when connecting. Specifically, we view both the access point and the authentication server as active adversaries trying to deanonymize the client at authentication time. The exchange between this client and the server must therefore be indistinguishable from any authentication exchange not just to third-party eavesdroppers but to the authentication peers: the server and AP.

We define an anonymous authentication protocol as an exchange between client and server that has the following properties. Let $\mathrm{UID} = \{C_i : 1 \leq i \leq n\}$ 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 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 clients do not have access to an anonymous channel over which to interact with the server without revealing their identity. Such is the case when clients build and destroy links to the server network over time.

1.2 Contributions

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

- A PIR-based authentication protocol enabling a client to prove that they are authorized to access the network without leaking any information about their identity. Our protocol is secure against fully malicious, covert adversaries [4], allowing detection of dishonest servers.

- A new underlying NTRU-based PIR technique we call Multiple Row Selection (MRS) that allows the retrieval of the sum of any subset of rows by using arbitrary N-degree polynomials with a reduced number of multiplications on the server. This is a contribution of independent interest, and it may be used as the PIR mechanism for other applications.

- We design and implement a Wi-Fi construction for TracEdge compatible with the Extensible Authentication Protocol (EAP). Our implementation scales to databases in the order of 10^7 clients. Our FFT implementation using parallel GPU computations enable us to reduce the server PIR computation time to 43.9ms.

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.

The subject of full client anonymity is a complex one, as information leaked by each communication layer can be used to deanonymize clients. For instance, physical layer fingerprinting techniques can use variations of behavior of hardware implementations to identify devices [33]. On higher layers, traffic patterns of the client after authentication succeeds could be observed by the AP operator to deanonymize them. While the above are real and interesting problems, TracEdge does not claim to address them, and it is designed to be a solution to the problem of anonymous Wi-Fi authentication. We offer some discussion on the subject in Section 7.

This paper is organized as follows. Section 2 shows an overview of related work. We present our anonymous authentication method TracEdge in Section 3. Section 4 shows our single-database Private Information Retrieval scheme. Section 5 describes our system of TracEdge as a Wi-Fi authentication method over 802.1X [30] Extensible Authentication Protocol for large databases. We present the evaluation of our scheme in Section 6. We conclude and discuss our results on Section 7.

1.3 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 $\mathrm{E}_{\mathrm{PK}}(m)$ and $\mathrm{D}_{\mathrm{PK}}(m)$ respectively. We use $\mathcal{E}(m)$ and $\mathcal{D}(m)$ for an homomorphic encryption scheme to distinguish it from the above. Signatures are noted as $\mathrm{Sig}_{\mathrm{PK}}(m)$. The concatenation of a and b is noted as $a \parallel b$, and the symbol $\perp$ represents an invalid key value.

2. RELATED WORK

Work on anonymous credentials spans several decades [5, 9, 10, 13]. With 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 decades. Group [14] and ring [47] 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. [49] 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. [32] 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 [6, 16, 21, 51] relating to 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 [7, 18, 20, 28] 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 [26, 39, 41, 56, 59] seek to distribute a secret among n recipients such that revoked members cannot decrypt new messages. For this the recipients are logically organized as leafs 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 $\mathrm{O}\left(r \log\left(n/r\right)\right)$ 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 [15], 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) [22, 23, 34, 37], 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 [17, 44] 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 in recent years [2, 22, 29, 35] for their improved speed in comparison to number-theoretic algorithms, allowing for faster processing.

Oblivious-RAM (ORAM) techniques [24, 27, 46, 52], 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 [25, 57] are periodically being discovered and mitigated. New protocols and services that improve on the client experience and security have also been proposed [36, 58]. 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.

3. ANONYMITY BY PIR

As stated on Section 1 TracEdge provides anonymous client authentication, and as consequence of the mechanisms in our protocol no traffic, AP usage patterns, nor identities are disclosed to the server. 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 $\mathrm{P} > \frac{1}{n} + \varepsilon$, where ε 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.

3.1 Adversary Model

TracEdgetargets protection against *covert* adversaries, see [4] for an overview. In contrast to a semi-honest adversary, a covert adversary is not bound to follow protocol execution and observe, but can arbitrarily (maliciously) deviate from the protocol to reach their goal. Yet, an important object of such an adversary is to remain covert, i.e., not be caught in performing malicious activities.

Covert adversaries are typical in scenarios where detection of malicious behavior can have significant consequences for the adversary. For example, an ISP that is reported to "cheat" on their clients' privacy will not only lose customers and revenues, but might even face legal consequences.

With TracEdge, we design an authentication mechanism that will protect clients' privacy. In case an ISP tries to cheat, clients will be able to prove this to a third party.

3.2 PIR Authentication

Algorithm 1 shows our basic authentication mechanism. We assume a Public Key encryption scheme

$$\mathcal{S} = \left(\mathrm{A}_{\mathrm{pub}}, \mathrm{A}_{\mathrm{priv}}, \mathrm{E}\left(\right), \mathrm{D}\left(\right)\right)$$

exists for every client A and for which the server and clients 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 [45], a public ledger such as the one in Namecoin [54], 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 client, the server uses a random key K, for which it builds a table with rows $\mathrm{E}_{\mathrm{A}_{\mathrm{pub}}}(K)$ for every client A in $\mathcal{U}$. An authenticating client 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 with the server. Given the table index i_0 where A's entry resides, the client constructs a query vector $\mathbf{v}$ using additive homomorphic encryption (detailed on Section 4.3). 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 4. The main insight in PIR, is that the result is obtained by performing a computation on every element of the database, thus hiding the value requested by the client to the server. This is in fact an $\Omega(n)$ operation if full anonymity in the set is desired, as any row skipped by the server leaks information about the query. In other words, the cost of computation may be traded off by being anonymous in a proportionally smaller set of the users. For instance if $\mathrm{O}(m)$ $m < n$ computation is desired, then one may reduce the cost by a factor of k at the expense of being anonymous in a smaller set.

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 for all clients 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.

To check a server commitment $\Gamma = g^K h^r \mod p$ where r is random nonce for the table, the client decrypts the value $K\{r\}$[1] in the server response with the retrieved value K', giving r'. If the client finds the commitment Γ equal to the calculated value $g^{K'} h^{r'} \mod p$ if and only if the retrieved K' matches the commitment.

3.3 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 clients to detect the server misbehavior.

[1]Note that the server could send r encrypted with the flagged key of the client, but to do so, it would still need to guess the authenticating client.

To mitigate this problem, a client who would like to increase their 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. To do so, the client retrieves $\mathrm{E}_{\mathrm{B}_{\mathrm{pub}}}(K)$ and compares it with its own encryption of K with B's public key, increasing their probability of finding a flagged key if one exists. As long as the algorithm is used to encrypt K is deterministic, e.g., Elgamal with a "fixed" random coin, clients can verify the extra entries they have obtained. 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

$$\mathrm{E}_{\mathrm{i}_{\mathrm{pub}}}(K) = (r \cdot P, r \cdot i_{pub} + P_K) = (r \cdot P, r \cdot Y_i + P_K)$$

where Y_i is client 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.

In addition to the row auditing procedure outlined above, our underlying PIR mechanism allows the query to retrieve a summation of multiple rows, thus reducing the complexity of row audit to a single query. We present the details of our Multiple Row Selection PIR scheme in Section 4.2. The process can be further streamlined by having the client retrieve a subset of the columns from the server. Since the PIR computation both on the server and client side depends on the number of columns retrieved, a faster key checking can be achieved in trade for fewer key bits audited. Because of multiple row selection, auditing becomes independent of the number of rows checked. Furthermore, because a client need not ask to retrieve all columns in the table, the key checking process has at worst the same cost as that of retrieving a the client key.

3.4 Correctness and Privacy

Given a key K, the client public key for the client A_{PUB} and a valid encryption $\mathrm{E}_{\mathrm{A}_{\mathrm{PUB}}}(K)$, 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 they learn with the chance of being exposed. Assume that the server commits to key K_1 but sets m entries in the table with key K_2. Table 1 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 client makes a query for an entry that doesn't match the

Algorithm 1: TracEdge authentication.

```
Data: $audit = \alpha$ the number of times to check for flags
$h = g^y \mod p$ the server's public signing key
$y$ the server's private signing key
$K$ the global key for this table
$p$ large prime

C → S : $t_s$ //timestamp
C ← S : $\mathrm{Sig}_{\mathrm{S_{PR}}}\left(g^K h^r \mod p \parallel t_s\right)$
//server commits to an encryption of $K$ with
  random $r$ for all clients
C → S : PIR_Q($A$)
C ← S : PIR_Resp( $\mathrm{E}_{\mathrm{A_{pub}}}(K')$ ), $K\{r\}$
$\mathrm{Sig}_{\mathrm{S_{PR}}}\left(\text{PIR\_Resp}(\mathrm{E}_{\mathrm{A_{pub}}}(K') \parallel K\{r\} \parallel t_s)\right)$

C :Check signature, compute
$K' =$ PIR_Extract(PIR_Resp( $\mathrm{E}_{\mathrm{A_{pub}}}(K')$ )
$r' \leftarrow \text{Decrypt}_{K'}(K\{r\})$

C : if $g^{K'} h^{r'} \neq g^K h^r$ then
    $K \leftarrow \perp$ and report server
end
while $audit > 0$ do
    C : $row \leftarrow \{1 \ldots n\} \setminus A$
    C → S : PIR_Q($row$)
    C ← S :
    PIR_Resp( $\mathrm{E}_{\mathrm{row_{pub}}}(K'')$ ),
    $\mathrm{Sig}_{\mathrm{S_{PR}}}\left(\text{PIR\_Resp}(\mathrm{E}_{\mathrm{row_{pub}}}(K'') \parallel t_s)\right)$

    C : Checks signature, computes
    $X =$ PIR_Extract(PIR_Resp( $\mathrm{E}_{\mathrm{row_{pub}}}(K'')$ )
    C : $audit \leftarrow audit - 1$

    C : if $\mathrm{E}_{\mathrm{row_{pub}}}(K') \neq X$ then
        $K \leftarrow \perp$ and report server
    end
end
C ↔ S : Mutual Authentication Protocol using $K$
```

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 client, the probability of not being detected decreases exponentially as a function of the number of queries.

Let DB $= \{\mathrm{E}_{\mathrm{C_i}}(K_i), 1 \leq i \leq n\}$ 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 queries to rows selected uniformly at random has a probability p_{nd} of no client detecting a flagged key:

$$p_{nd} = \prod_{i=0}^{a-1} \left(1 - \frac{m}{n-i}\right)$$

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

Table 1: Identity Leakage vs. Misbehaving Server Detection

	Commit. matches	**No matches**
Srv. learns	$\log\left(\frac{n}{n-m}\right)$ bits	$\log\left(\frac{n}{m}\right)$ bits
Srv. exposed?	No, prob. $\frac{n-m}{n}$	Yes, prob. $\frac{m}{n}$

clients, 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 clients, 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 they lose the clients' trust irrevocably, a server cannot misbehave with impunity for any reasonable amount of time. In Section 4.2, we will show how multiple queries can be aggregated in a fraction on a single PIR query.

3.5 Table Management

Because clients may enter or leave the authentication set at any point, the authentication table must be managed such that queries and row auditing present consistent results. The operations related to table management in our model are as follows:

Empty row creation The server computes the encryption of key K using its own public key. This value is taken to signify an empty row value.

Client ID Assignment At the time of registration to the system, every client is assigned an empty table row. This assignment will be maintained across table generations, and the value of the assigned row is communicated to the client. The server writes the encrypted value of K for the new user on its corresponding row.

Table creation For every non-empty row, the server writes the encryption of K with the public key of the client assigned to the row. Empty entries receive the empty row value.

Client removal The server writes an empty row value on the exiting client row, and frees the client ID assignment for the row, and the rekeying process is started.

Rekeying When the K expires, a new table is used. If no tables are available in the table pool, a new one is generated and the old table is marked as deprecated and used until a new one is available. Keys retrieved during this time expire a fixed time after retrieval, after which clients are forced to authenticate.

Note that client removal can still happen during the time a table is marked as deprecated, as the server can update the row with the empty value on the fly. Therefore any number of client removals may happen between old table deprecation and the use of the new table. Keys expire after the table is deprecated to prevent now invalid users to access the system and to prevent credential disclosure.

4. PRIVATE INFORMATION RETRIEVAL

The Query and Response functions used in Algorithm 1 rely on Private Information Retrieval primitives. We preface our optimizations with an overview of PIR and NTRU Encrypt, our underlying homomorphic encryption scheme, to offer a

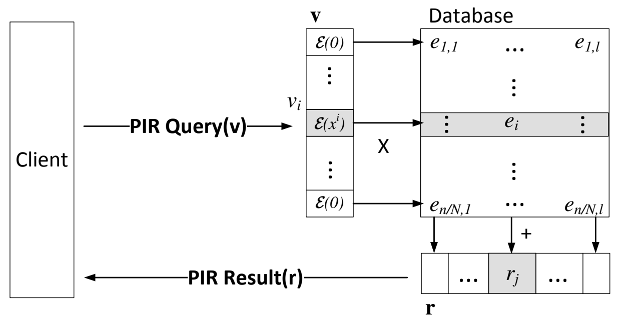


Figure 1: Private Information Retrieval overview.

better understanding of the mechanisms on which TracEdge is built.

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.

4.1 Overview

In our scheme, the database is a set of n records of length ℓ bits. Every row i contains bits $\mathrm{DB}_i = \{b_{i,j}, 1 \leq j \leq \ell\}$, and we use NTRUEncrypt [29, 50] over the Ring $\mathbb{Z}_q / \left(X^N - 1\right)$ with polynomial multiplication $\star$, 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 4.2. 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 they are not being targeted by the server (we describe this feature in Section 3.4). 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 1):

Query Generation The client computes vector $\mathbf{v}$ 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 $\lfloor i_0/N \rfloor$-th element in $\mathbf{v}$.

Response Generation To generate the reply, the server divides the database in regions of N rows, grouping bits in the same region's column as a polynomial $e_{i,j} = \sum_{i=0}^{N-1} b_{i,j} x^j$ and computes $e'_{i,j} = v_i \star e_{i,j}$. The meaning of $\star$ depends on the additive homomorphism of $\mathcal{E}(\cdot)$, which for NTRU is polynomial multiplication modulo $X^N - 1$ as described in Section 4.3).

The server then computes a result vector $\mathbf{r}$ 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 $\mathcal{E}$. For additive homomorphic encryption $\mathcal{E}(a+b) = \mathcal{E}(a) + \mathcal{E}(b)$ for instance, the elements of r can be written as

$$\begin{aligned} r_j &= \sum_{i \neq i_0} e_{i,j} \star \mathcal{E}(0) + e_{i_0,j_0} \star \mathcal{E}\left(x^{i_0}\right) \\ &= \mathcal{E}\left(e_{i_0,j_0} \star x^{i_0}\right) \end{aligned}$$

Note that the values in the databases need not be in plaintext or even encrypted with the same $\mathcal{E}(\cdot)$. 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 complexity of our scheme can be improved by using Kushilevitz's PIR [34]. This particular technique reduces communication complexity to $\mathcal{O}\left(\sqrt{n}\right)$. While it can be generalized for $\mathcal{O}(n^{1/d})$, we show in Section 6 that $d = 2$ gives enough savings for short communication time.

4.2 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 $\mathbb{Z}_q[X] / \left(X^N - 1\right)$, the product $x^{i_0} \star y \mod \left(X^N - 1\right)$ rotates the coefficients of y by i_0 positions or $x^{i_0} \star y = \sum_{i=0}^{N-1} y_{i+i_0 \mod N} x^{i+i_0 \mod N}$. 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 $\mathbf{v}$ was zero, or a rotation of the query region, where $\mathbf{v}$ 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.

In general, for any query vector $\mathbf{v}$, the resulting vector $\mathbf{r}$ contains a summation of the rows whose indexes match the non-zero coefficients of $\mathbf{v}$. This is a useful observation for it allows clients to verify the compliance of the server database with the claimed commitment with a *single* query over an arbitrary set of rows. To do so, the client constructs a query vector with more than one non-zero coefficient. To verify the result, the client simply adds the expected encryptions of K of every row corresponding to the query and tests for equality. Any flagged key included in the set deviates the result away from the calculated value, and reveals server misbehavior. Note that, it is sufficient for the client to retrieve a subset l of the columns at the cost of a reduced detection probability by probability $(1 - \frac{1}{2^l})$. Therefore, a cheating server can be mitigated in the covert adversaries model at the fraction of the cost of a single PIR query.

4.3 NTRU Encryption

In this section we describe the properties of NTRU [53] 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 $\mathbb{Z}_q[X] / \left(X^N - 1\right)$ 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 6.

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 $f \star g = \sum_{i=0}^{N-1} h_i x^i$ where

$$h_k = \sum_{i+j \equiv k \mod N} f_i \cdot g_j$$

The private key f is chosen at random with the standard [53] recommending coefficients in the range $[-1, 1]$. The public key is $h = p \star f_q \star g \mod q$ 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

$$c = r \star h + P_m \mod q,$$

where r is a polynomial chosen at random modulo q.

Decryption consists of two steps. First the value $a = f \star c \mod q$ is computed. Secondly, $P_m = f_p \star a \mod p$ 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

$$r_j = \sum_{i \neq i_0} \mathcal{E}(0) \star e_{i,j} + \mathcal{E}\left(x^{i_0}\right) \star e_{i_0, j_0}$$

so to extract the response, we note that polynomial multiplication $g \star x^{i_0} \mod (X^N - 1)$ rotates the coefficients of g cyclically by i_0 positions. In particular, coefficient k of g is

$$(g \star x^{i_0})_k = g_{k - i_0 \mod N}$$

therefore, to calculate the bit associated with column r_j the client extracts the $2 \cdot i_0 \mod N$-th bit of $\mathcal{D}(r_j)$.

4.4 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.

Performing the addition of 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. SYSTEM

We implement TracEdge as an extension to the Protected EAP [30], 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.

Access to the directory of client public keys is crucial for the key checking mechanism. Since clients are authenticating to set up a link to the ISP, access to the directory can be performed through other channels such as mobile data or through the AP providing access only to the directory.

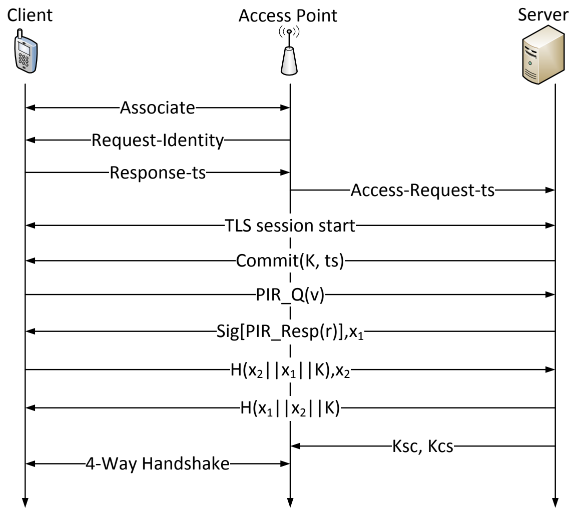


Figure 2: Architecture of the **EAP-TE** protocol.

5.1 The EAP-TE method

Figure 2 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`.

5.2 The Supplicant

Our client query generation is a wpa-supplicant [38] patch for Linux and Android systems. Query generation on the client involves computing 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 [34] 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 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.

5.3 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 6.

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 client 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 Optimizations

Several expensive operations occur in **EAP-TE**. On the client side, vectors with $\sqrt{n}$ elements with mostly $\mathcal{E}(0)$ are generated to form the PIR query. While NTRU Encrypt is a fast operation compared to other number-theoretic schemes, polynomial multiplication is intensive enough to add considerable time for large numbers of clients if done naively. We show the impact of the implementation on the speed of polynomial multiplication on Section 6. 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.

As outline in Section 3.5, tables need not be generated immediately. Delay in key revokation in this case works similarly to current response times of around one hour for mobile providers. 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 $\mathrm{H}(K) \cdot P$ and uses it to build every entry in the table, i.e., $(\mathrm{H}(K) \cdot P, \mathrm{H}(K) \cdot x \cdot P + K)$, where $x \cdot P$ is the client's public key. This halves the number of EC scalar multiplications.

Our system uses the parallel NTRU-PIR described in Section 4.4. 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 6.

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, β CUDA blocks add two polynomials at a time in shared memory, producing β polynomials. Finally a single block adds the former to get the result for the column. We find the optimum value of the parameter β on Section 6.

6. 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. The clients consist of Android-based smartphones.

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 allowing tens of Mbits/second. We justify these assumptions in Section 6. 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 [11, 12]. 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 4.

Table 2 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 needed for the PIR response generation. Conversely, a faster curve allows for quicker 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,

Table 2: Implementation choices

Parameter	Value
$\|K\|$	128-bit
$\mathrm{E}_{\mathrm{PUB}}(K)$	EC Elgamal [19]
EC Curves	`sect131r1`, `sect163k1`
$\mathrm{H}(\cdot)$	SHA-256
$\mathcal{E}(\cdot)$	NTRU-Encrypt [29]
NTRU Parameters	`APR2011_439_FAST` [53]
N	439
q	2^{21}
n	10^7 clients

Table 3: Choice of q

q	Observed Additions
2^{11}	2
2^{12}	47
2^{13}	250
2^{14}	951
2^{15}	4408
2^{16}	18087
2^{17}	73047
2^{18}	367865
2^{19}	1.12×10^6
2^{20}	7.01×10^6
2^{21}	2.21×10^7

we limit table size, generation time, and key lifetime. We discuss the implications and countermeasures of extended key lifetime in Section 7.

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 $\mathcal{E}(0)$ to $\mathcal{E}(1)$. 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 3 summarizes our results.

6.1 Database and Query Generation

Given the parameters in Table 2 for TracEdge, the maximum size of the table using EC Elgamal over 163-bit security is 41×10^7Bytes, or 410MB. Our server code builds a database table by choosing a key K uniformly at random, and using $\mathrm{H}(K)$ 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 5.

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 6 summarizes the results.

Table 4: Evaluation systems

	Hardware Setup	Graphics/WLAN
Srv. 1	Quad 3.0GHz, 8GB	9800GT (112c)
Srv. 2	Quad 2.5GHz, 8GB	GTX280 (240c)
Srv. 3	2x 8c. Xeon, 130GB	-
Srv. 4	8x Xeon, 21GB	2x GF100 (448c)
Srv. 5	8x Xeon, 21GB	1x GK104 (1536c)
Srv. 6	Quad 2.5GHz, 8GB	GTX780 (2304c)
Cli. 1	Galaxy SII 1GB RAM	802.11bgn
Cli. 2	HTC One 2GB RAM	802.11abgn

Table 5: Table creation time (s) $\pm\sigma$

Server	`sect163k1`	`sect131r1`
1	1160 ± 4	3424 ± 9
2	1190 ± 9	3211 ± 13
3	261 ± 7	751 ± 12
4	$1105s \pm 3$	2997 ± 10

Table 6: Query generation on client

Device	Avg. Encryption (μs)$\pm\sigma$	Total time
1	159.3 ± 0.12	1.86s
2	173.4 ± 0.08	1.98s

6.2 Communication Complexity, and Performance

Communication complexity determines the amount of data that needs to be sent over the air. The scale of our client database makes it a good candidate to use the technique in [34]. 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 4 to the target APs and connect to a test server at a well-provisioned site. Afterwards we make a similar measurement with Speedtest [43] as a control. We measure downlink and uplink throughput over a period of 10 seconds. Figure 3 shows how over 50% of the sampled APs can sustain throughputs higher than 10Mbps over 10 seconds.

Response extraction at the client is performed by decrypting all the polynomials received from the server. Table 7 shows average times for our clients over 1000 samples.

We also measure authentication latency as seen on the client handsets, defined as the time between query transmission and response receipt in Table 7, and it includes 350ms computation time on the single-GPU system and 0.9s network latency. We compare the measured latency value with the expected latency for a multiple-GPU system later in this section.

6.3 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 compu-

Table 7: Query extraction and latency on client

Device	Av. Decryption	Total time	Latency
1	$63.7 \pm 0.20\mu s$	45.3ms	1.41s
2	$78.6 \pm 0.13\mu s$	55.9ms	1.42s

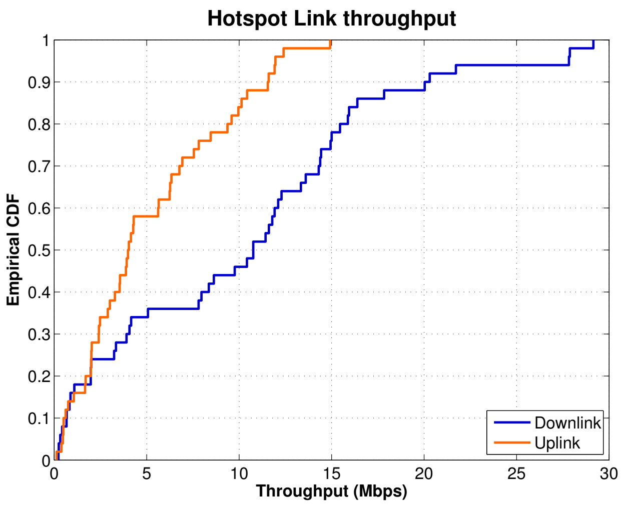


Figure 3: Public ISP hotspot throughput in urban area.

Table 8: Unoptimized Polynomial Operations $N = 439$ (μs)

Srv.	Add.	Mult.	GPU Mult.	Total/col.
1	4.28	251.6 ± 0.33	76.5	438s
2	2.67	228.2 ± 0.38	38.4	225ms
3	2.94	129.9 ± 0.05	-	-
4	2.97	130.2 ± 0.1	13.4	113.5ms
5	2.97	130.2 ± 0.1	5.9	48.8ms
6	2.67	228.2 ± 0.38	5.7	27.9ms

tation lengthy. Fortunately, the operation is suitable for parallelization, as discussed in Section 4.4. We evaluate two computation strategies: multi-threading over CPU cores and GPU computation using CUDA [42]. Table 8 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 $\mathrm{O}\left(N^2\right)$ multiplication algorithm used.

In contrast, the Fast Fourier Transform strategy outlined in Section 5.4 significantly reduces the evaluation time per column (over an order of magnitude gain). Figures 4a and 4b 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) 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. The improvement comes from increased parallelism during the point-to-point product in the frequency domain, where the result is dependent only on the two factor coefficients.

Our results above give the time required of end-to-end authentication using TracEdge by adding query transmission, PIR computation, response transmission and query extraction of 1.12 seconds. Because table and query generation can be performed offline they do not add to authentication time. Likewise, mutual authentication after the global key is retrieved is not considered for end-to-end authentication as it is fast and common to other secure Wi-Fi authentication protocols. As shown in Table 7, our single-GPU system performs closely to the expected computation time in a multi-GPU system. Moreover, because multiple row auditing has a lesser or equal cost than key retrieval, two GPUs are enough perform the operation simultaneously.

To understand how well our system performs under heavy client load, we measure the maximum number of authentications per second our server is able to sustain by saturating with with client queries. While our server is only able to maintain 2.3 authentications per second with only one GPU because of the number of memory transfers required to process the whole table, a multi-GPU with 8 devices can sustain up to 60 auth/s.

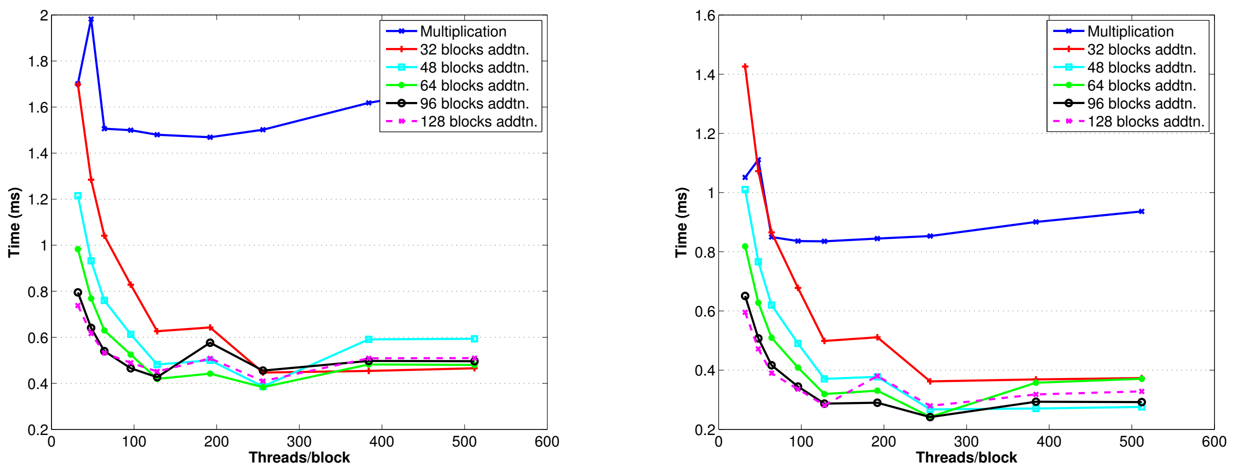
Building a multi-GPU setup assuming a \$380 retail price of the GTX780 devices plus four host computers with two GPUs installed per host, would cost in the range of \$4000-5000. This results an investment of at most \$83.33 per authentication/sec from the provider for 10 million clients.

7. CONCLUSIONS AND DISCUSSION

We have presented TracEdge, an authentication protocol that protects client identity against covert adversaries. TracEdgemitigates 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 estimation of its computational and communication costs. The key PIR component can perform response computation in 43.9 ms on a platform with 8 off-the-shelf GTX780 GPUs. With devices in the market with over 5700 cores, computation times can be reduced further by augmenting hardware capabilities.

Implementation of TracEdge implies a cost on the ISP, as multiple parallel computation devices need to be deployed for operation. As stated in Section 6, as much as \$83 per authentication per second may be required, making TracEdge more of a niche market solution for clients who are more focused on privacy. A more reduced user base however, also lowers the hardware requirements linearly.

While delaying table generation opens the possibility of credential sharing where a client may authenticate with the server and then share the retrieved key with unauthorized clients, prevention can be achieved with a combination of TracEdge with a USIM component [1] and Distance Bounding Protocols [8] running between the SIM card and AP. In this case, the AP needs to have access to the authentication key. Similar to EAP-AKA method [31] which uses the subscriber module to authenticate to a network, a TracEdge implementation would store the retrieved authentication key and use it to produce the necessary keying material, preventing human/software access to this value. The outcome of the distance bounding authentication is a time-limited shared session key between this specific AP and the mobile client.



(a) Amazon GK104GL (b) Desktop GTX 780

Figure 4: CUDA parameter impact in PIR computation time.

Sharing this session key with other cheating clients does not help them since it is tied to a single AP-client session. An unauthorized client would then need to be co-located with the original client, contending for the same AP.

Further identity leaks from traffic pattern analysis are still possible as they fall outside the scope of TracEdge. A client concerned with this risk may use anonymity networks such as TOR to hide their identity to the access point. While there is a performance penalty for using these networks, their performance has seen an upward trend, reaching 9 Mbps per relay average [55], making it comparable with our link throughput wardriving experiment in Section 6. This suggests relay anonymity networks are no longer a bottleneck.

References

[1] 3GPP Technical Specification Group Services and System Aspects. 3GPP TSG S3.26: Need for a WLAN specific UCC application. Tech. rep., Nov. 2002.

[2] Aguilar-Melchor, C., and Gaborit, P. A fast private information retrieval protocol. In *Information Theory, 2008. ISIT 2008. IEEE International Symposium on* (July 2008), pp. 1848–1852.

[3] Apple Inc. iOS Security Guide. `https://www.apple.com/privacy/docs/iOS_Security_Guide_Oct_2014.pdf`, Oct. 2014.

[4] Aumann, Y., and Lindell, Y. Security against covert adversaries: Efficient protocols for realistic adversaries. *Journal of Cryptology 23*, 2 (2010), 281–343.

[5] Belenkiy, M., Camenisch, J., Chase, M., Kohlweiss, M., Lysyanskaya, A., and Shacham, H. Randomizable proofs and delegatable anonymous credentials. In *Advances in Cryptology-CRYPTO 2009*. Springer, 2009, pp. 108–125.

[6] Beresford, A. R., and Stajano, F. Location privacy in pervasive computing. *IEEE Pervasive Computing 2*, 1 (Jan. 2003), 46–55.

[7] Boneh, D., Gentry, C., and Waters, B. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *Advances in Cryptology–CRYPTO 2005* (2005), Springer, pp. 258–275.

[8] Brands, S., and Chaum, D. Distance-bounding protocols. In *Advances in Cryptology—EUROCRYPT'93* (1994), Springer, pp. 344–359.

[9] Brands, S. A. *Rethinking public key infrastructures and digital certificates: building in privacy.* MIT Press, 2000.

[10] Camenisch, J., and Lysyanskaya, A. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *Advances in Cryptology—EUROCRYPT 2001*. Springer, 2001, pp. 93–118.

[11] Certicom Research. SEC 2: Recommended Elliptic Curve Domain Parameters Version 1.0. `http://www.secg.org/sec2_final.pdf`, 2000.

[12] Certicom Research. SEC 2: Recommended Elliptic Curve Domain Parameters Version 2.0. `http://www.secg.org/sec2_v2.pdf`, 2010.

[13] Chaum, D. Security without identification: Transaction systems to make big brother obsolete. *Commun. ACM 28*, 10 (Oct. 1985), 1030–1044.

[14] Chaum, D., and van Heyst, E. Group Signatures. In *EUROCRYPT* (1991), D. W. Davies, Ed., vol. 547 of *Lecture Notes in Computer Science*, Springer, pp. 257–265.

[15] Chor, B., Goldreich, O., Kushilevitz, E., and Sudan, M. Private Information Retrieval. In *FOCS* (1995), IEEE Computer Society, pp. 41–50.

[16] Chow, R., and Golle, P. Faking contextual data for fun, profit, and privacy. In *Proceedings of the 8th ACM Workshop on Privacy in the Electronic Society* (New York, NY, USA, 2009), WPES '09, ACM, pp. 105–108.

[17] Damgård, I., and Jurik, M. A Generalisation, a Simpli.cation and Some Applications of Paillier's Probabilistic Public-Key System. In *Public Key Cryptography*, K. Kim, Ed., vol. 1992 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2001, pp. 119–136.

[18] Dodis, Y., and Fazio, N. Public key broadcast encryption for stateless receivers. In *Digital Rights Management*. Springer, 2003, pp. 61–80.

[19] ElGamal, T. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In *Advances in Cryptology*, G. Blakley and D. Chaum, Eds., vol. 196 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 1985, pp. 10–18.

[20] Fiat, A., and Naor, M. Broadcast encryption. In *Advances in Cryptology—CRYPTO'93* (1994), Springer, pp. 480–491.

[21] Gedik, B., and Liu, L. Location privacy in mobile systems: A personalized anonymization model. In *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems* (Washington, DC, USA, 2005), ICDCS '05, IEEE Computer Society, pp. 620–629.

[22] Gentry, C. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009.

[23] Gentry, C., and Ramzan, Z. Single-Database Private Information Retrieval with Constant Communication Rate. In *ICALP*, L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, Eds., vol. 3580 of *Lecture Notes in Computer Science*, Springer, pp. 803–815.

[24] Goldreich, O., and Ostrovsky, R. Software Protection and Simulation on Oblivious RAMs. *J. ACM 43*, 3, 431–473.

[25] Gonzales, H., Bauer, K., Lindqvist, J., McCoy, D., and Sicker, D. Practical Defenses for Evil Twin Attacks in 802.11. In *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM)* (December 2010), pp. 1–6.

[26] Goodrich, M., Sun, J., and Tamassia, R. Efficient tree-based revocation in groups of low-state devices. In *Advances in Cryptology – CRYPTO 2004*, M. Franklin, Ed., vol. 3152 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2004, pp. 511–527.

[27] Goodrich, M. T., Mitzenmacher, M., Ohrimenko, O., and Tamassia, R. Practical oblivious storage. In *Proceedings of the second ACM conference on Data and Application Security and Privacy* (New York, NY, USA, 2012), CODASPY '12, ACM, pp. 13–24.

[28] Halevy, D., and Shamir, A. The lsd broadcast encryption scheme. In *Advances in Cryptology—CRYPTO 2002*. Springer, 2002, pp. 47–60.

[29] Hoffstein, J., Pipher, J., and Silverman, J. H. NTRU: A ring-based public key cryptosystem. In *Algorithmic number theory*. Springer, 1998, pp. 267–288.

[30] IEEE. Standards for Wireless, Local, and Metropolitan Area Networks. `http://standards.ieee.org/findstds/standard/802.1X-2010.html`, 2012.

[31] IETF. Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA). `http://tools.ietf.org/html/rfc4187`, 2006.

[32] Jarecki, S., Kim, J., and Tsudik, G. Authentication for Paranoids: Multi-party Secret Handshakes. In *Applied Cryptography and Network Security*, J. Zhou, M. Yung, and F. Bao, Eds., vol. 3989 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2006, pp. 325–339.

[33] Kohno, T., Broido, A., and Claffy, K. C. Remote physical device fingerprinting. *Dependable and Secure Computing, IEEE Transactions on 2*, 2 (2005), 93–108.

[34] Kushilevitz, E., and Ostrovsky, R. Replication is NOT Needed: SINGLE Database, Computationally-Private Information Retrieval. In *FOCS* (1997), IEEE Computer Society, pp. 364–373.

[35] Lauter, K., Naehrig, M., and Vaikuntanathan, V. Can Homomorphic Encryption be Practical? *IACR Cryptology ePrint Archive 2011* (2011), 405.

[36] Lee, N. Facebook's head of special projects talks wearables, WiFi and human connections. `http://www.engadget.com/2013/11/11/facebook-expand-ny/`.

[37] Lipmaa, H. First CPIR Protocol with Data-Dependent Computation. *IACR Cryptology ePrint Archive 2009* (2009), 395.

[38] Malinen, J. WPA Supplicant. `http://hostap.epitest.fi/wpa\_supplicant/`, 2012.

[39] Naor, D., Naor, M., and Lotspiech, J. Revocation and tracing schemes for stateless receivers. In *Advances in Cryptology — CRYPTO 2001*, J. Kilian, Ed., vol. 2139 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2001, pp. 41–62.

[40] Narayanan, A., and Shmatikov, V. Robust De-anonymization of Large Sparse Datasets. In *IEEE Symposium on Security and Privacy*, IEEE Computer Society, pp. 111–125.

[41] Noubir, G. Multicast Security. Tech. rep., European Space Agency, 1998.

[42] nVidia. Compute Unified Device Architecture. `http://developer.nvidia.com/object/cuda.html`, 2012.

[43] OOKLA. The global broadband speed test. `http://http://www.speedtest.net/`, 2014.

[44] Paillier, P. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *Advances in Cryptology EUROCRYPT 99*, J. Stern, Ed., vol. 1592 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 1999, pp. 223–238.

[45] PGP Corporation. PGP Global Directory. `https://keyserver.pgp.com`, 2014.

[46] Pinkas, B., and Reinman, T. Oblivious RAM Revisited. *IACR Cryptology ePrint Archive 2010* (2010), 366.

[47] Rivest, R. L., Shamir, A., and Tauman, Y. How to Leak a Secret. In *ASIACRYPT* (2001), C. Boyd, Ed., vol. 2248 of *Lecture Notes in Computer Science*, Springer, pp. 552–565.

[48] Rushe, D. Lavabit founder refused FBI order to hand over email encryption keys. `http://www.theguardian.com/world/2013/oct/03/lavabit-ladar-levison-fbi-encryption-keys-snowden`, Oct. 2013.

[49] Schechter, S., Parnell, T., and Hartemink, A. Anonymous Authentication of Membership in Dynamic Groups. In *Financial Cryptography*, M. Franklin, Ed., vol. 1648 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 1999, pp. 184–195.

[50] Security Innovation. NTRU Reference Implementation, 2014.

[51] Shokri, R., Theodorakopoulos, G., Papadimitratos, P., Kazemi, E., and Hubaux, J.-P. Hiding in the mobile crowd: Locationprivacy through collaboration. *Dependable and Secure Computing, IEEE Transactions on 11*, 3 (2014), 266–279.

[52] Stefanov, E., and Shi, E. ObliviStore: High Performance Oblivious Cloud Storage. In *IEEE Symposium on Security and Privacy* (2013), IEEE Computer Society, pp. 253–267.

[53] The IEEE P1363 Working Group. Standards for Lattice-Based public-key cryptography. `http://grouper.ieee.org/groups/1363/`, 2008.

[54] The Namecoin Team. Namecoin, 2015.

[55] The TOR Project. Tor Metrics: Servers. `https://metrics.torproject.org/network.html`, 2014.

[56] Wallner, D., Harder, E., and Agee, R. Key management for multicast: Issues and architectures, 1999.

[57] WatchGuard. Anatomy of a Wireless "Evil Twin" Attack. `http://www.watchguard.com/infocenter/editorial/27079.asp`, 2011.

[58] WiFi Alliance. Wi-Fi CERTIFIED Passpoint. `https://www.wi-fi.org/hotspot-20-technical-specification-v100`, June 2012.

[59] Wong, C. K., and Lam, S. Digital signatures for flows and multicasts. *Networking, IEEE/ACM Transactions on 7*, 4 (Aug 1999), 502–513.