REPORT DOCUMENTATION PAGE	*Form Approved* **OMB No. 0704-0188**

1. REPORT DATE *(DD-MM-YYYY)* NOV 2012	**2. REPORT TYPE** Conference Paper (Pre-Print)	**3. DATES COVERED** *(From - To)*

4. TITLE AND SUBTITLE FOR YOUR PHONE ONLY: CUSTOM PROTOCOLS FOR EFFICIENT SECURE FUNCTION EVALUATION ON MOBILE DEVICES (PRE PRINT)	**5a. CONTRACT NUMBER** FA8750-11-2-0211
	5b. GRANT NUMBER N/A
	5c. PROGRAM ELEMENT NUMBER 62303E
6. AUTHOR(S) Henry Carter, Chaitrali Amrutkar, Italo Dacosta and Patrick Traynor	**5d. PROJECT NUMBER** PROC
	5e. TASK NUMBER ED
	5f. WORK UNIT NUMBER GA

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Georgia Institute of Technology, 515 10th St. NW, Atlanta, GA 30332; University of Oregon, 677 East 12th Ave, Eugene, Oregon 97403	**8. PERFORMING ORGANIZATION REPORT NUMBER** N/A
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) DARPA, 3701 North Fairfax Drive, Arlington, VA 22203-1714; Air Force Research Laboratory, Air Force Research Site/RITA, 525 Brooks Rd, Rome, NY 13441-4505	**10. SPONSOR/MONITOR'S ACRONYM(S)** N/A
	11. SPONSORING/MONITORING AGENCY REPORT NUMBER AFRL-RI-RS-TP-2012-058

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved For Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.

13. SUPPLEMENTARY NOTES
In submission, International Conference on Information Security and Cryptology (ICISC), Nov 28 – 29, 2012. This work was funded in whole or in part by Department of the Air Force contract number FA8750-11-2-0211. The U.S. Government has for itself and others acting on its behalf an unlimited, paid-up, nonexclusive, irrevocable worldwide license to use, modify, reproduce, release, perform, display, or disclose the work by or on behalf of the Government. All other rights are reserved by the copyright owner.

14. ABSTRACT
Mobile applications increasingly require users to surrender private or context-sensitive information, such as GPS location or social networking data. To facilitate user privacy when using these applications, Secure Function Evaluation (SFE) could be used to obliviously compute functions over encrypted inputs. The dominant construction for desktop applications is the Yao garbled circuit, but this technique requires significant processing power and network overhead, making it extremely expensive on resource-constrained mobile devices. In this work, we develop Efficient Mobile Oblivious Computation (EMOC), a set of two SFE protocols customized for the mobile platform. Using partially homomorphic cryptosystems, we develop protocols to meet the needs of two popular application types: location-based and social networking. Using these applications as comparison benchmarks, we demonstrate execution time improvements of 99% and network overhead improvements of 96% over the most optimized garbled circuit techniques available. Our results show that mobile application developers should reconsider implementing garbled circuits due to their extreme resource usage, and instead rely upon our equivalently secure and significantly more efficient alternative.

15. SUBJECT TERMS
garbled circuit, cryptosystem, mobile, partially homomorphic

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** CARL R. THOMAS
a. REPORT U	**b. ABSTRACT** U	**c. THIS PAGE** U	UU	20	**19b. TELEPHONE NUMBER** (*Include area code*) N/A

For Your Phone Only: Custom Protocols for Efficient Secure Function Evaluation on Mobile Devices

Henry Carter, Chaitrali Amrutkar, Italo Dacosta, and Patrick Traynor

Converging Infrastructure Security (CISEC) Laboratory
Georgia Tech Information Security Center (GTISC)
Georgia Institute of Technology
{carterh, chaitrali}@gatech.edu,
{idacosta, traynor}@cc.gatech.edu

Abstract. Mobile applications increasingly require users to surrender private or context-sensitive information, such as GPS location or social networking data. To facilitate user privacy when using these applications, Secure Function Evaluation (SFE) could be used to obliviously compute functions over encrypted inputs. The dominant construction for desktop applications is the Yao garbled circuit, but this technique requires significant processing power and network overhead, making it extremely expensive on resource-constrained mobile devices. In this work, we develop *Efficient Mobile Oblivious Computation* (EMOC), a set of two SFE protocols customized for the mobile platform. Using partially homomorphic cryptosystems, we develop protocols to meet the needs of two popular application types: location-based and social networking. Using these applications as comparison benchmarks, we demonstrate execution time improvements of 99% and network overhead improvements of 96% over the most optimized garbled circuit techniques available. Our results show that mobile application developers should reconsider implementing garbled circuits due to their extreme resource usage, and instead rely upon our equivalently secure and significantly more efficient alternative.

1 Introduction

The confluence of high-speed connectivity and device capability has led to the recent surge in mobile application development. While software common to desktop computing (e.g., word processing, email) exists in this space, the most popular mobile applications often provide services based on a user's current context (e.g., location [5], social interconnections [1], etc.). Such applications allow users to make more informed decisions based on their surroundings. However, these applications also regularly expose sensitive data to potentially untrusted parties.

Cryptographers have long worked to develop mechanisms that allow two parties to compute shared results without exposing either individual's sensitive inputs or requiring assistance from a trusted third-party. Such techniques are referred to as *Secure Function Evaluation* (SFE), and provide a set of powerful primitives for privacy-preserving computation. While garbled circuits have been known for nearly 30 years [26], efficient realizations of such schemes have only become possible recently [19, 16, 9, 13, 20]. However, their use on mobile devices, where the nature of applications are different and the use of context sensitive information is the norm and not the exception, has just begun to be assessed [11]. In the past, special-purpose protocols using partially homomorphic encryption [7, 10, 27, 6] have been developed and optimized for specific SFE applications (e.g. cryptographically verifiable voting). This technique promises significant performance gains, but has yet to be applied to mobile applications.

In this paper, we develop specialized protocols designed to perform privacy-preserving versions of operations commonly found in applications running on mobile phones. Our *Efficient Mobile Oblivious Computation* (EMOC) techniques use partially homomorphic cryptosystems to restate secure computation as a series of simple arithmetic operations over encrypted inputs. Specifically, we design and implement two privacy-preserving protocols and demonstrate their use in two popular application classes: location-based Twitter feeds (a geographic proximity test protocol) and a social networking tool to identify nearby "friends of friends" (a private set intersection protocol). Comparing these applications with equivalent garbled circuit constructions, we demonstrate that our applications can produce the same results at computational and bandwidth costs *reduced by orders of magnitude in some cases.*

In so doing, we make the following contributions:

– **Design privacy-preserving mobile applications replacing garbled circuit constructions with homomorphic cryptographic primitives:** We design custom privacy-preserving protocols to meet the specific resource constraints of the mobile platform. We then implement these protocols in applications representative of two of the most popular mobile application classes: *location-based messaging* and *social networking*. We prove that our applications provide equivalent security guarantees to their SFE-based counterparts.
– **Propose canonical evaluation tests for mobile SFE applications:** In the desktop world, canonical tests for SFE efficiency have existed for several years. The existence of this common frame of reference for performance between varying techniques has fostered significant growth in the number of schemes available and the performance efficiencies of those schemes. However, these desktop applications are not representative of the types of privacy-preserving computation that would be most useful on the mobile platform (e.g. it is unlikely two mobile users need to securely compute AES). As no such representative test applications have been developed for the mobile platform, we propose a set of test applications to facilitate further study in developing efficient mobile SFE. We will soon open `www.foryourphoneonly.org` as a common repository for mobile SFE applications, providing the research community with a set of existing mobile SFE techniques to compare new techniques as they are developed.
– **Characterize SFE mobile performance profiles:** The relative performance capabilities of garbled circuits on the mobile platform is largely unknown up to this point. In this work, we use our proposed test applications to conduct an extensive performance analysis of five well-known SFE compilers on the Android mobile platform to determine their feasibility in practice. We demonstrate that our custom-designed SFE protocols offer improvements in execution time as high as 99% and network overhead improvements as high as 96% over the most optimized garbled

circuit techniques. Moreover, we examine several garbled circuit optimizations that have never been compared on any platform [16, 13, 20, 15], providing a set of test data to build on in future mobile SFE research.

Our proofs and performance evaluations demonstrate that the performance gains achievable through partially homomorphic constructions merit special protocols for certain functions. Moreover, our results call for the reevaluation of the recent claims made by Huang et al. [12] that general circuit compilers provide comparable efficiency to custom protocols - *we show empirically and rigorously that this claim does not hold for functions representing the most common applications on the resource-constrained mobile platform.*

2 Related Work

With the development of the “garbled circuit” SFE protocol, Yao demonstrated the possibility of two peer users computing a function without exposing their private inputs [26]. For years following, implementations of the protocol were too computationally intensive for practical use. In 2004, Malkhi et al. produced the first practical implementation of Yao garbled circuits in the program Fairplay [19]. Fairplay provided a high-level language and compiler for building the logical circuits that are used to compute functions securely (i.e., without revealing either party’s inputs to the other). Fairplay offers the same privacy guarantees as the trusted third party model without requiring an actual third party. Building upon the Fairplay compiler, Kruger et al. developed a technique for replacing garbled circuits with ordered boolean diagrams [16] to improve Fairplay’s speed for certain functions. Huang et al. developed a technique for pipelining circuit construction and evaluation, allowing for circuits to scale to any size without filling up the memory of the constructing machine [13]. More recently, Mood et al. developed a memory-saving circuit generation technique using an intermediate compiler language [20], while Kreuter et al. developed a technique for highly parallelizeable large-scale circuit evaluation [15]. These efforts have produced a practical means for performing secure computation in a desktop environment. However, garbled circuit evaluation still requires significant processor and memory overhead when producing and evaluating circuits, and exchanging encrypted inputs. Even with the improved performance of Kruger’s OBDD and Huang’s pipelining approach, and considering Kerschbaum’s assertion that communication overhead is of little importance in secure computation [14], garbled circuits are likely to be too expensive for the hardware constraints of mobile devices. Huang et al. began exploring this question in a work examining the performance of pipelined circuits on mobile phones [11]. We thoroughly evaluate this question in our work.

One possible solution to this problem lies in the relatively young area of homomorphic encryption. Henecka et al. demonstrated that homomorphic encryption can be used in conjunction with garbled circuits to provide performance improvements for some SFE problems [9]. However, many special-purpose protocols have been designed to use only partially homomorphic encryption to preserve privacy in applications such as private set intersection [7, 3, 6], voting applications [10], and distributed location privacy [27]. In addition, several protocols for private information retrieval [21] and private stream search [2, 23] leverage this partially homomorphic property of certain encryption schemes to search and compare encrypted data in a manner that prevents the machine performing the search from learning anything about the data. The benefit of the currently available partially homomorphic encryption schemes is that they are efficient, even on mobile devices [24]. Considering the extreme processing and memory constraints found on the mobile platform, a new set of custom protocols developed for the mobile platform is necessary. While Huang challenges this notion [12], our paper presents privacy-preserving protocols that demonstrate the efficiency gains of custom-designing SFE protocols over general garbled circuit compilers on the mobile platform.

3 Cryptographic Assumptions

Before we define and prove the security of our applications, we specify the requirements for the underlying primities. We also state basic assumptions that are necessary for the security of our protocols to hold.

3.1 Homomorphic Cryptography

The main tool our protocols use in guaranteeing the privacy of all inputs is the homomorphic property of certain cryptosystems. For a cryptosystem to be homomorphic, there must be some operation that, when performed on two ciphertexts, causes some predictable change to the underlying plaintexts. Our EMOC protocols capitalize on one homomorphic property: multiplicative homomorphisms. In a multiplicative homomorphic scheme, the product of two ciphertexts is equivalent to an encryption of the product of two plaintexts. Specifically, for a multiplicative homomorphic encryption scheme $E_k()$, given two plaintext messages $X, Y \in M$; ciphertexts $C, D : E_k(X) = C, E_k(Y) = D$:

$$C \times D \equiv E_k(X \times Y) \tag{1}$$

3.2 Public Key Encryption

The second requirement for cryptosystems used in our protocols is that they be public key encryption schemes. All of these protocols require that one of the participants in a two-party computation must perform homomorphic operations over encrypted data. This user must be able to encrypt his own inputs, but be *unable* to decrypt the result of the homomorphic operations. This operation is clearly only feasible in a public key cryptosystem, where the user performing homomorphic operations does not possess the decryption key. In addition, we assume any cryptosystem used will produce ciphertexts that are indistinguishable under chosen-plaintext attack (i.e. semantically secure).

3.3 Threat Model

In developing the protocols used in our applications, we make two assumptions to provide results in a fair and secure manner. The first is that finding a trusted third party is difficult or impossible. As an example, while a number of websites currently offer to

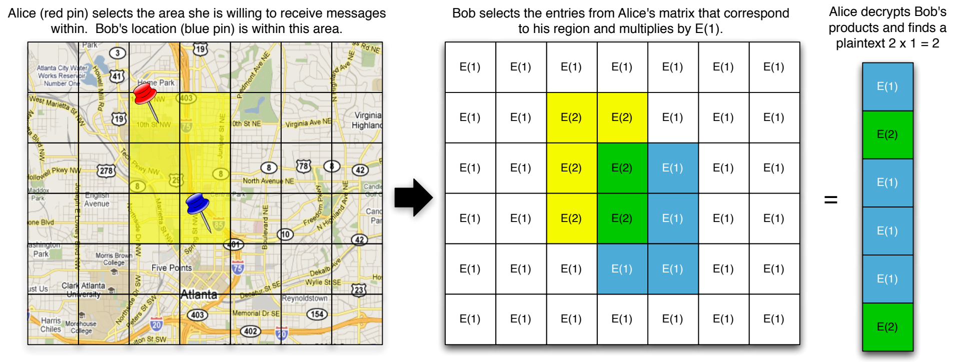

Alice (red pin) selects the area she is willing to receive messages within. Bob's location (blue pin) is within this area.
Bob selects the entries from Alice's matrix that correspond to his region and multiplies by E(1).
Alice decrypts Bob's products and finds a plaintext 2 x 1 = 2
E(1)
E(2)
=
Atlanta
Five Points

Fig. 1: Proximity Test Protocol. Alice builds a location matrix with encryptions of ‘1’ in every entry except those that correspond to the area she is willing to receive tweets within. In her travel area, she enters encryptions of ‘2’. Bob selects the entries that correspond to his travel area, multiplies each entry by an encryption of ‘1’, and returns the product to Alice. When Alice decrypts, she knows that: if any value is a ‘2’, Bob’s tweet is relevant to her. If every value is a ‘1’, Bob’s tweet is not relevant to her location.

provide the location of friends within a certain physical radius, they can not be trusted to *not* process, store or sell such data.

The second assumption is that all privacy guarantees in Section 5 hold against a semi-honest adversary. As defined in Lindell and Pinkas’ work [18], this means that an adversary will follow the protocol as written, using valid inputs, but will attempt to learn as much as possible outside the jointly computed results by studying logs of all communications. Since this protocol is meant to guarantee the privacy of inputs, we can do nothing if the user chooses false inputs designed to corrupt the protocol. Many garbled circuit implementations also makes this same assumption, *proving security based on semi-honest adversaries* [19, 16, 9, 13, 18]. Not only is our model equivalent to the current security model for two-party computation, we assert that this model is realistic and useful for certain context-sensitive mobile applications in the market. Our protocols developed under this threat model will also provide a foundation for seeking protocols that can guarantee privacy against other adversarial models.

4 EMOC Application Protocols

In this section, we describe in detail the EMOC protocols, applied in two sample applications. We first present a protocol for geographic proximity testing in a Location-Based Twitter application, which allows Alice to subscribe to Bob’s tweets without either party revealing their location. Second, we present a private set intersection protocol in our Social Graph connectivity tool, which allows Alice and Bob to determine where their social networks overlap without exposing the identities of all of their friends - an application with potential use when meeting new (and untrusted) people. As a simplified proof of concept, we develop a protocol for solving the canonical millionaire’s problem in the technical report version of this work [4].

4.1 Geographic Proximity Test

Location-based messaging, especially for advertisements, has recently received significant attention. Beyond advertising based on location, it offers the potential for useful applications such as a proximity test to alert two people if they are close enough to arrange a meeting. It could also be combined with applications like Twitter to allow for location-based tweet filtering and following. However, these applications must query the physical location of a users, which could compromise the user's privacy. To resolve this information leakage, we present a protocol for securely computing when two users are within a chosen proximity of one another. While used in a specific application here, the protocol can be used in any location-based mobile application. The ability to specify an input region of any shape or size allows the proximity test to provide a result at any desired granularity, from the same building to the same city.

Problem Definition Assume two Twitter users, a follower Alice and a tweeter Bob. Since Bob generally tweets about events in his vicinity, Alice wishes to receive tweets from Bob only when she is nearby. Alice selects as her input an area around her current location where, if Bob tweets close to this area, she wants to receive the tweet. Bob inputs an area around his current location where his tweets would be relevant. The goal is to compute whether the area where Alice wishes to receive Bob's tweets intersects with the area where Bob's tweets are relevant.

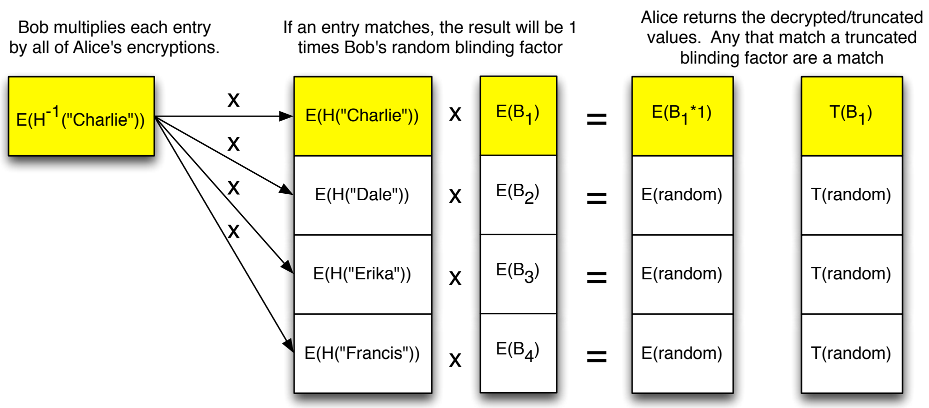
Description of the Protocol Before the protocol is initiated, both parties must define the following elements:

- A *multiplicative* homomorphic encryption scheme $E_{pk}(\cdot)$. In all figures, this will be denoted as E(·).
- A matrix of size $M \times N$ where each cell corresponds to a physical region within the city where Alice and Bob are located. Imagine the matrix as a grid laid over a city map. Each cell has a publicly known correlation to the city location beneath it.

Before receiving any of Bob's location-based tweets, Alice selects an area of her city of any shape or size that defines the area where she wants to receive tweets (Figure 1). She then generates an $M \times N$ location matrix L_A. For each cell, Alice inputs a '2' if that cell corresponds to the area where she wishes to receive tweets, and '1' if it does not. She then encrypts each cell in the matrix with her public key pk_A. When Alice checks Bob's Twitter feed, she initiates the protocol by sending L_A to Bob.

When Bob receives Alice's location matrix L_A, Bob then selects the cells in L_A that correspond to the region where his tweet is relevant, and for each of the n cells, he re-randomizes the entry by multiplying in an encryption of the value '1'. Bob then returns an array of the n results R to Alice. Upon receiving R, Alice decrypts the values using her private key to find n values, either '1' or '2'. She then sends this decrypted array of values back to Bob to complete the computation.

If any of the values returned is a '2', the area where Bob's tweet is relevant intersected with the area where Alice wished to receive tweets, and Bob can respond by delivering his latest tweet. If all the values returned are '1', the area where Bob's tweet is relevant does not intersect with the area Alice wishes to receive tweets, and Bob need not respond. In this manner, Alice never learns Bob's precise location and vice versa.



Bob multiplies each entry by all of Alice's encryptions.
If an entry matches, the result will be 1 times Bob's random blinding factor
Alice returns the decrypted/truncated values. Any that match a truncated blinding factor are a match
E(H^{-1}("Charlie"))
x
E(H("Charlie"))
E(H("Dale"))
E(H("Erika"))
E(H("Francis"))
E(B$_1$)
E(B$_2$)
E(B$_3$)
E(B$_4$)
=
E(B$_1$*1)
E(random)
T(B$_1$)
T(random)

Fig. 2: Private set intersection: Bob homomorphically multiplies each entry in his array by every entry in Alice's array. He then multiplies in a unique blinding factor for all of the resulting values. Alice receives these values, decrypts them, and truncates all but the least significant t bits, which she returns. If these t bits match the least significant t bits of the blinding factor multiplied in that entry, Bob knows there is a match.

4.2 Private Set Intersection

Social networking applications are a popular channel for communicating with a mobile device. However, they are also a potential channel to leak private information about a user's social life. If two mobile users were to meet at a party or conference, one might only want to allow the other into her social network based on the friends they already have in common. However, there is currently no application which allows this without revealing both users' entire social graphs. This application offers a means for securely revealing only the friends common to both users while maintaining the privacy of the rest of both social graphs. Again, we couch our protocol in an application that is highly relevant to mobile users. However, the protocol can be used in general to compute the intersection of any two sets without revealing any element outside of the intersection.

Problem Definition Assume two participants, Alice and Bob, who are both members of a social network. Each participant assigns a subset of the social network members as their friends. Given both Alice and Bob's lists of friends, we wish to compute which members of the social network are friends with both Alice and Bob while keeping the rest of their friend lists private.

Description of the protocol Before the protocol can be initiated, the following elements must be defined:

- A *multiplicative* homomorphic encryption scheme $E_{pk}(\cdot)$ which allows for modular multiplication over some cyclic group G. In all figures, this is denoted E($\cdot$).
- A secure keyed hash algorithm.
- A security parameter t.
- A predetermined number of friends N to be compared.

To initiate the protocol, Alice begins by generating a query array Q_A of size N. She does this by generating a hash key K_H and hashing each of her friends' names. She then stores these hashes in random order in Q_A. Alice then encrypts each entry with her public key pk_A and then sends Q_A, K_H, and pk_A to Bob.

Upon receiving Q_A, Bob hashes each member of his friend list using K_H. Then, he finds the multiplicative inverse of each hash within the group G which is the group of elements over which Alice's public key can encrypt. As we will later observe, if one of Bob's hashes matches one of Alice's, the product of her hash and his inverted [1] hash will be '1'. At this point, Bob encrypts each of his inverted hashes with Alice's public key pk_A. He then generates two arrays of length N^2. For the first array, he homomorphically multiplies each of his encrypted and inverted hash values by all of Alice's encrypted hash values, performing N^2 comparisons. In the second list, he generates N^2 random elements of the group G, which he also encrypts with Alice's public key. He then homomorphically multiplies the elements in this array of blinding factors B with the elements in his array of compared values, generating the result of homomorphic computation R (Figure 2). Finally, Bob sends R back to Alice.

When Alice receives R, she decrypts each element using her private key, which yields an array of N^2 random values from the group G due to Bob's blinding factors. She then sends back *only* the t least significant bits of each decrypted value to Bob.

Bob receives the decrypted values from Alice and for each entry in her results, he compares the bits returned with the least significant t bits of the blinding factor he multiplied into that entry. If the results match, it means that entry contained a '1', implying that a friend matched. To calculate which friend matched, Bob simply uses integer division on the index, where $i \div N$ is the original index of the matching friend. Thus, Bob can identify which members' of the social network he shares with Alice. Bob then sends the list of names in the intersection back to Alice to complete the computation.

Correctness Argument We quantify here the probability that Bob will have a false positive when matching the last t bits of the blinding factor with Alice's returned result. Since the hash function is assumed to be pseudorandom, multiplying a pseudorandom number x by the multiplicative inverse of a pseudorandom number y^{-1} yields a pseudorandom number z. After this value is multiplied by a random blinding factor b and truncated to the least significant t bits, it is apparent that the probability of the least significant t bits of b matching the least significant t bits of $z * b$, is $\frac{1}{t^2}$, or no better than randomly selecting bits. This value t can be increased to yield a higher probability of correctness or decreased to hide more information about the resulting hashes. The amount of information revealed by t bit truncation is defined in section 5.

5 Privacy Guarantees

In this section, we define our threat model and prove the privacy guarantees of both EMOC protocols. For each protocol, we show two properties: the security of the two-party computation and the amount of information revealed by the result of computation.

[1] This does not mean Bob inverts the hash to recover the preimage; rather, the mathematical inverse of the hash value mod p.

5.1 Definitions

In all of our protocols, we assume the standard definition of a semi-honest adversary, described in Lindell and Pinkas' [18]. Essentially, this states that both parties will follow the protocol as written but will attempt to learn information beyond the computed result from transcripts of the interaction. This assumption is also made by related efforts in this space [19, 16, 9, 13, 18]. To prove a protocol secure against semi-honest adversaries, we use the concept of indistinguishability between Alice's view in a real execution and a simulator's generation in an ideal execution. In the ideal world, two participants A, B send their inputs a, b to a trusted third party which performs some computation and returns the result $f(a, b)$. The proof idea is to show that a simulator S in the ideal world can simulate A's view in the real protocol.

Definition 1 *Semi-honest security: For any deterministic functionality $f(x, y)$ and semi-honest parties P_1 and P_2, we say that protocol π securely computes f in the presence of semi-honest adversaries if there exists ppt algorithms S_1 and S_2 such that:*

$$S_1(x, f(x, y))_{x,y\in\{0,1\}^*} \overset{c}{\approx} view_1^{\pi}((x, y), output^{\pi}(x, y))_{x,y\in\{0,1\}^*} \tag{2}$$

$$S_2(x, f(x, y))_{x,y\in\{0,1\}^*} \overset{c}{\approx} view_2^{\pi}((x, y), output^{\pi}(x, y))_{x,y\in\{0,1\}^*} \tag{3}$$

5.2 Location Privacy

Theorem 1 *Location Privacy: Assuming the encryption scheme used in the proximity test protocol is semantically secure, the proximity test protocol is secure in the presence of semi-honest adversaries.*

Proof. We prove the security of the protocol separately for each participant.

Alice is corrupt: For the proximity test protocol, we define $f(x, y)$ as follows: given inputs (x, y) that are the grid locations where Alice and Bob wish to send and receive messages, $f(x, y)$ is a shuffled set of size c grid locations which, if an intersection exists, contains the intersecting grid locations. The variable c is defined as the number of grid locations in Bob's input y. We now prove that a simulator S_A operating in the ideal world can simulate Alice's view of the real world. S_A is constructed as follows:

S_A is given input $(x, f(x, y))$, where x is Alice's input and $f(x, y)$ is the result as defined above. Upon receiving Alice's initial message containing her encrypted location matrix L_A, S_A encrypts each element in $f(x, y)$ with Alice's public key and returns $m_1 = E(f(x, y))$, completing Alice's view of the interaction.

We now show that S_A's message is indistinguishable from Alice's view of a real execution. First, we know that for any encrypted element in $f(x, y)$ at location i:

$$E(f(x, y)_i) \overset{c}{\approx} E(1 * 1) \overset{c}{\approx} E(1 * 2) \tag{4}$$

Based on the semantic security of our encryption scheme, each encryption of an entry in $f(x, y)$ is indistinguishable from an encryption of the result of Bob's homomorphic operations. Second, we know trivially that the decrypted values in m_1 are identical to Bob's message in a real interaction, as both simply contain the values in $f(x, y)$. Therefore, the proximity test protocol is secure when Alice is corrupt.

Bob is corrupt: We construct the simulator S_B as follows: S_B is given $(y, f(x, y))$, where y the grid locations where Bob will be sending messages, and $f(x, y)$ is the result as defined above. S_B prepares a location matrix L_A by storing the values in $f(x, y)$ in the grid locations in y. For any grid location not in y, S_B stores a '1'. S_B then generates an encryption key pair and encrypts each entry of L_A under the public key PK_{S_B}. It then sends the message $m_1 = E(L_A)||PK_{S_B}$. When S_B receives the shuffled array R containing the result of Bob's homomorphic operations, S_B simply decrypts each entry in R using SK_{S_B} and returns $m_2 = D(R)$.

We show that Bob's view in a real execution and an interaction with S_B are indistinguishable. For each element of $f(x, y)$ and for each entry in the matrix L_A:

$$E(f(x,y)_i) \overset{c}{\approx} E(1) \overset{c}{\approx} E(2) \tag{5}$$

Based on the semantic security of the encryption scheme, Bob cannot distinguish between encryptions of the resulting values and encryptions of the values Alice would encrypt in a real interaction. The public key PK_{S_B}, being randomly generated, is clearly indistinguishable from Alice's public key.

The second message is clearly indistinguishable, since in both the real view and the simulated view, the message contains a randomly ordered array containing the values of $f(x, y)$. Therefore, the proximity test protocol is secure when Bob is corrupt.

Information Revealed: We now show the probability of either participant guessing the exact location of the other participant. Given Alice and Bob's areas of willingness to send/receive messages x, y, the number of entries in the location matrix n, and the result of the protocol $f(x, y)$, both parties know the size of the intersection between their send/receive areas as well as the number of cells in Bob's sending area. In the worst case, either all of Bob's sending area is contained within Alice's receiving area or vice versa. Consider if Bob's area is contained within Alice's, without loss of generality. Alice has probability $\frac{1}{|x|}$ of guessing which cell contains Bob's actual location, and Bob has probability $\frac{1}{n}$ of guessing Alice's location since he does not know the size or shape of Alice's receiving area outside of his own sending area. Thus, given $g = max(|x|, |y|)$, the maximum probability of guessing the other party's location is $\frac{1}{g}$.

5.3 Private Set Intersection Privacy

Theorem 2 *Private Set Intersection Privacy: Assuming the encryption scheme used in the private set intersection Protocol is semantically secure and that the secure hash function used is pseudorandom and one-way, the private set intersection protocol is secure in the presence of semi-honest adversaries.*

Proof. Again, we prove separately the security of our protocol for Alice and Bob.

Alice is corrupt: We define the function $f(x, y)$ as follows: given inputs (x, y), which are the list of friends for Alice and Bob respectively, $f(x, y) = x \cap y$. We now construct a simulator S_A in the ideal world that, given x and $f(x, y)$, is capable of simulating Alice's view of a real interaction:

S_A is given $(x, f(x, y))$ as defined above. Upon receiving Alice's initial message containing an array of encrypted friend hashes, S_A generates an array of n^2 random blinding factors $B_1, ..., B_{n^2} \in G$, encrypts them with Alice's public key, and returns message $m_1 = E(B)$. Upon receiving Alice's second message, the decryption and truncation of m_1, S_A responds with message $m_2 = f(x, y)$.

We now show that both of S_A's messages are indistinguishable from their counterparts in Alice's view of a real execution. For the first message, we again make two points. First, we know that, for $i = 1...n^2$ and friend names x, y:

$$E(B_i) \overset{c}{\approx} E(H(x) * H^{-1}(y) * B_i) \tag{6}$$

Based on the semantic security of the encryption system, S_A's encryption of the blinding factor is indistinguishable from Bob's result gained by homomorphic multiplication. Second, we know for any two names x, y:

$$B_i \overset{s}{\approx} H(x) * H^{-1}(y) * B_i \tag{7}$$

Based on the definition of statistical indistinguishability, the result Alice sees in both a simulated and a real interaction appears random. Therefore, the private set intersection protocol is secure when Alice is corrupt.

Bob is corrupt: We construct simulator S_B as follows: S_B is given $(y, f(x, y))$. S_B begins by generating a random key K_h for the agreed symmetric hash, a random public key encryption pair PK_{S_B}, SK_{S_B}, and an array F of the pre-defined length n. S_B then enters the hashes of the names in $f(x, y)$ in uniformly random indices of F. For the remaining unfilled entries, S_B generates random numbers from the range of the hash. Bob then encrypts each entry with PK_{S_B} and sends $m_1 = E(F)||PK_{S_B}||K_h$. Upon receiving Bob's response R, S_B decrypts the response with SK_{S_B} and returns m_2, which is the last t bits of each entry in $D(R)$, where t is the predetermined security parameter for the length of the resulting values.

We show that Bob's view in a real execution of the protocol and an interaction with S_B are indistinguishable. Considering m_1, for each entry in F and for each hashed name x, given a uniformly random group element r:

$$E(r) \overset{c}{\approx} E(x) \tag{8}$$

Based on the semantic security of our encryption scheme, Bob cannot distinguish between an encryption of a hashed name and an encryption of a random value. Again, since S_B and Alice both generate their keys randomly, they are indistinguishable.

Considering m_2, we see that, given a random key, for any two names x, y:

$$H(x) \overset{c}{\approx} \mathcal{U} \overset{c}{\approx} H(y) \tag{9}$$

Based on the pseudorandomness of the hash function, any hashes produced will be indistinguishable from randomness. Because of this, these samples will remain indistinguishable after any polynomial time operations are performed over them. Therefore, for any non-matching entry in m_2 and any names x, y, z, where x is in Bob's friend list, y is in Alice's, z is randomly chosen by S_B, and $T()$ is the truncation function:

$$T(H(x) * H^{-1}(y)) \overset{c}{\approx} T(H(x) * H^{-1}(z)) \tag{10}$$

Therefore, the private set intersection protocol is secure when Bob is corrupt.

Information Revealed: We now show the probability of either party guessing the input of the other party. For an instance of the protocol we are given inputs x, y that are the inputs of both parties (without loss of generality), some number t which is the number of bits returned by Alice in the third message, some number n which is the security parameter of the hash function, and $f(x, y)$ which is the intersection of the inputs. Alice receives only the list of names in $f(x, y)$, so the probability of her guessing any friend in Bob's friend list outside of this intersection is random selection over all possible friends. Bob receives the last t bits of Alice's hash values multiplied by the inverse of his own hash values. To reverse this value back into the name of one of Alice's friends, Bob must examine all possible values for the $n - t$ truncated bits, and must invert the hash of Alice's friend's name. Therefore, Bob's ability to learn any of Alice's friends not in the intersection is no better than random over the possible truncated values and the possible hash pre images, $\frac{1}{2^{(n-t)+n}}$.

We do note a slight difference in the security guarantees provided by our protocol and the guarantees of garbled circuits. While garbled circuit constructions keep all data cryptographically secured by the garbling function, our scheme reveals to Bob t bits of each of Alice's hashed inputs multiplied by the multiplicative inverse of Bob's hashed inputs. However, based on the pseudorandomness and one-wayness of a secure keyed hash, we maintain that in practice it is still computationally infeasible to recover the remaining bits of the hashed value *and* reverse the hashed value to the correct preimage.

6 Performance Analysis

While developing new SFE protocols is useful, the main contribution of our work is the establishment of an efficient means for performing SFE on the mobile platform. Another contribution is our canonical test set that will facilitate future comparisons between techniques and encourage additional development of efficient mobile SFE schemes. Using these tests, our results demonstrate that through custom designed protocols, we can take advantage of optimizations that are not available to general-purpose garbled circuit compilers. This allows our protocols to execute in time that would be usable by the average mobile user. We also provide baseline statistics that compare a variety of garbled circuit techniques that have yet been untested on the mobile platform.

6.1 Mobile SFE Benchmarking Applications and Metrics

To demonstrate the efficiency of a given secure function evaluation technique, we chose two protocols that are widely applicable in mobile applications: a geographic proximity test and a private set intersection protocol. As we have already shown, these particular functions would be very useful in some of the most popular mobile applications. As such, they are a critical benchmark when examining new mobile SFE techniques.

In addition to presenting these test applications, we propose a set of metrics to compare efficiency between techniques executing the test applications. The first is average execution time, taken over 10 executions with 95% confidence in the error margin. To

Protocol	Input size	SFE scheme	Avg. exec. time (sec.)	Network use (KB)
Proximity Test	500 cells	EMOC	0.0259 (± 0.0054)	129.536
		OBDD	23.1480 (± 0.0351)	1,765.764
		Parallelized	26.2353 (± 0.0836)	1,854.049
		PAL	35.1888 (± 0.0487)	2,029.439
		Pipelined	11.1293 (± 0.0332)	603.497
		Fairplay	NA	NA
Private Set Intersection	20 friends	EMOC	3.8381 (± 0.0034)	109.120
		OBDD	124.4921 (± 0.2809)	2,879.016
		Parallelized	107.8990 (± 0.4249)	2,669.284
		PAL	130.7570 (± 0.2013)	3,025.966
		Fairplay	NA	NA
	16 friends	Pipelined	45.7061 (± 0.1254)	3,401.133

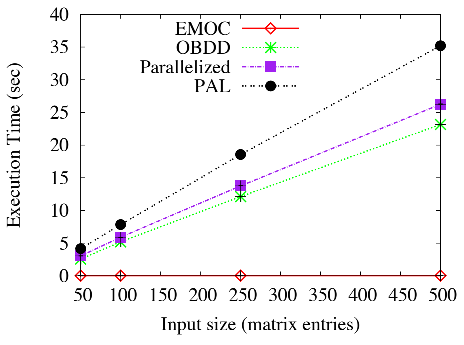
Table 1: Our experimental results. Values are present for the maximum input size measure across all applications for accurate comparison. In the private set intersection protocol, the Pipelined execution environment required input size to be a power of two.

demonstrate feasibility for practical use, we use inputs that correspond to real values a user might present to such an application. The second metric is total network usage between both parties, measured as the number of bytes exchanged during the protocol. As mobile devices are often required to function with costly network connections (both in terms of energy and billing), minimizing the amount of traffic required between two parties is critical to efficient performance.

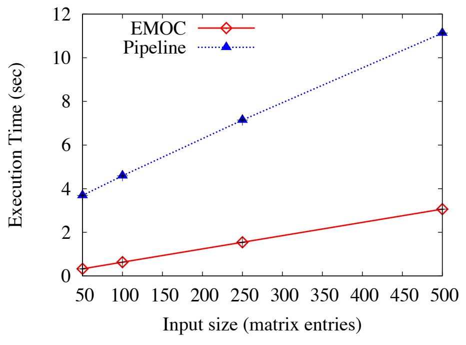
To demonstrate the practicality of our protocols in comparison to existing garbled circuit compilation techniques, we perform the experiments defined above for our scheme as well as five other garbled circuit compilation techniques. We selected these techniques because they represent the full range of general garbled circuit compilers available. We opted not to include the TASTY framework by Henecka et al. [9] in this work, as this framework requires circuits to be constructed and optimized by hand on a per-function basis. The focus of our work is to show the performance benefits of our scheme against garbled circuit compilers designed to compile *any general secure function* from a higher level language.

The first garbled circuit technique we evaluate is Fairplay [19], the standard desktop implementation of Yao's garbled circuit technique. Next, we examine Kruger's Ordered Binary Decision Diagram optimization [16], which produces smaller, more efficient garbled circuits through a modified representation. Third, the Parallelized scheme by Kreuter et al. [15] incorporates a number of optimizations for evaluating large circuits in parallel on server-class machines. The fourth scheme we evaluate is the Pseudo-Assembly Language (PAL) Compiler by Mood et al. [20], which produces circuits in a memory-efficient manner using an intermediate circuit compiler language. Finally, we examine the pipelined evaluation technique of Huang et al. [13], which splits garbled circuits into layers that can be generated and evaluated separately.

For all of the garbled circuit compilation techniques except the pipelined evaluation, we split the scheme into two phases: preprocessing and execution. For the preprocessing phase, we compiled the garbled circuits on a desktop and then examined their evaluation times, the execution phase, on the mobile device. To assure fair representation



(a) Evaluation phase



(b) Total Execution Time

Fig. 3: Proximity test execution times. Note that for the online execution, for all input sizes our application runs in a fixed amount of time while all garbled circuits show increasing execution times with increasing input size.

across techniques, we either compiled the circuits ourselves using compiler framework provided by the technique author or had the author compile the same SFDL circuit description on their own machines. In our own protocols, we consider the time Alice takes to generate her query as the preprocessing phase, while the online communication between Alice and Bob constitutes the evaluation phase. In the case of pipelined circuit evaluation, no such preprocessing phase exists, since the circuits are generated in layers during the online evaluation between the two parties. While this inability to amortize computation costs is a performance weakness when compared to other SFE techniques, we chose to perform separate experiments that compare their pipelining execution time against our combined preprocessing and online execution time. For these experiments, we acquired the same mobile framework implementation developed by Huang et al. [11]. While this does not provide a true picture of the efficiency gains in our amortized execution, it still demonstrates the significant performance gains of using partially homomorphic encryption instead of garbled circuits for mobile SFE applications.

We evaluated all precompiled garbled circuits on the standard Java Fairplay platform, which we ported to an Android application. Our own protocols were written in C and cross-compiled to run natively using the Android toolchain [8]. All performance figures were taken on the Samsung Nexus S smartphone.

6.2 Results

The results in Table 1 clearly demonstrate the advantages of our custom protocols over every garbled circuit technique tested. In both test applications, we see a significant increase in execution time and network usage for all garbled circuit implementations, even in the comparison of total execution time in Figures 3b and 4b. Because general purpose compilers cannot take advantage of optimizations inherent to specific functions, they tend to produce circuits with irregular performance profiles. This is clearly

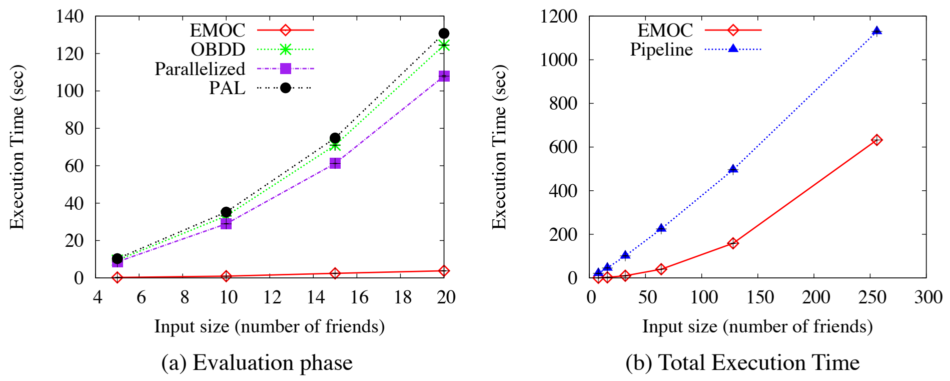


Fig. 4: Private set intersection execution times. For every garbled circuit technique except the pipelined circuits, we were only able to run experiments up to inputs of size 20 due to the large memory requirements of Fairplay.

seen in Figures 3a and 4a, where garbled circuit techniques do not consistently outperform one another between applications. For example, in the proximity test protocol, the OBDD scheme outperforms the parallelized scheme. However, this ordering is reversed for the private set intersection protocol. Ultimately, these fluctuations in performance are eclipsed by the gains achieved through our custom designed protocols, where online execution times were *at least 96% better* than the fastest garbled circuit technique. In the best case, our techniques reduced execution time by *three orders of magnitude*.

One advantage of our protocols is that for increasing input sizes, our proximity test protocol only requires an increase in preprocessing time, while the online execution remains constant across all input sizes. By contrast, *every* garbled circuit technique showed increasing execution times as input size increased, emphasizing this significant benefit of our customized protocols. In addition, the optimizations that are incorporated into garbled circuit schemes do not consistently provide any benefit on the constrained mobile platform. For example, the highly parallelized scheme performs about as well as non-parallelized garbled circuits, simply because most mobile phone hardware contains single-core processors. In the case of the pipelined evaluation circuits, we see an optimization that solves the problem of memory-intensive preprocessing (i.e., circuit generation), but does not allow for amortized execution time. EMOC provides a solution with an efficient preprocessing phase, where each precomputed ciphertext requires only 256 bytes of memory, *as well as* faster amortized execution. Contrary to Huang et al. [12], these results show that custom protocols significantly outperform the best available garbled circuit optimizations on mobile devices.

In addition to faster execution, our protocols significantly reduce the amount of network overhead compared to garbled circuit techniques. For the proximity test protocol, we observed a 78% reduction from the best garbled circuit technique, pipelined execution. In the private set intersection protocol, the improvement swelled to 96% over the best garbled circuit technique, the parallelized circuits. This improvement is due largely to the fact that our protocol does not use oblivious transfers to exchange inputs.

In theory, there exist a number of oblivious transfer schemes that perform with efficient ($O(n)$) communication overhead [22]. However, it is clear that these efficiencies do not always carry over in practice. In the case of our private set intersection protocol, we exchange theoretical "efficiency" for practical usefulness by employing techniques that use less network overhead in exchange for complexity that is theoretically less efficient ($O(n^2)$). See Appendix A for more results on the network usage of each technique.

7 Conclusion

As mobile phones become more popular, new techniques will be needed to protect the private information used in many of their applications. Garbled circuit constructions offer an increasingly realistic solution in the desktop space, but require too much processing power and network overhead to be practical on the mobile platform. By replacing garbled circuits with homomorphic encryption operations, our EMOC protocols demonstrate that certain privacy-preserving functions can be evaluated with great efficiency on the mobile platform. In addition, our canonical test applications provide a common reference point when comparing SFE techniques on the mobile platform. Using these metrics, our performance evaluation demonstrates improvements in our protocols of greater than 99% over the most efficient garbled circuit constructions, as well as an initial characterization of the performance capabilities of several garbled circuit optimizations on the mobile platform. Based on these results, we present our protocols as an efficient method for implementing SFE into some location-based and social networking applications. To foster further research into efficient mobile SFE, we make our test metrics and applications available to the research community at `www.foryourphoneonly.org` and encourage other authors working in this space to post their implementations as well.

Acknowledgments This material is based on research sponsored by DARPA under agreement number FA8750-11-2-0211. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government. We would also like to thank Kevin Butler, Peter Chapman, Yan Huang, Benjamin Kreuter, Louis Kruger, Benjamin Mood, Abhi Shelat and Chih-hao Shen for their assistance in generating garbled circuits under their respective schemes.

References

1. Banerjee, N., Agarwal, S., Bahl, P., Chandra, R.: Virtual compass: relative positioning to sense mobile social interactions. Tech. rep., Microsoft Research (2010), http://www.springerlink.com/index/K81H08U2767N2117.pdf
2. Bethencourt, J., Song, D., Waters, B.: Analysis-Resistant Malware. In: Proceedings of the ISOC Network and Distributed Systems Security (NDSS) Symposium (2008)

3. Camenisch, J., Zaverucha, G.: Private intersection of certified sets. In: Proceedings of the IFCA International Conference on Financial Cryptography and Data Security (FC) (2009)
4. Carter, H., Amrutkar, C., Dacosta, I., Traynor, P.: Efficient Oblivious Computation Techniques for Privacy-Preserving Mobile Applications. Tech. Rep. GT-CS-11-11, College of Computing, Georgia Institute of Technology (2011)
5. Constandache, I., Bao, X., Azizyan, M., Choudhury, R.: Did you see Bob?: human localization using mobile phones. In: Proceedings of the ACM International Conference on Mobile Computing and Networking (Mobicom) (2010)
6. De Cristofaro, E., Tsudik, G.: Practical private set intersection protocols with linear complexity. In: Proceedings of the IFCA International Conference on Financial Cryptography and Data Security (FC) (2010)
7. Freedman, M., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: EUROCRYPT (2004)
8. Google: Android project. http://source.android.com (2010)
9. Henecka, W., Kögl, S., Sadeghi, A.r., Schneider, T., Wehrenberg, I.: TASTY : Tool for Automating Secure Two-partY computations. In: Proceedings of the ACM conference on Computer and Communications Security (CCS) (2010)
10. Hirt, M., Sako, K.: Efficient Reciept-free voting based on homomorphic encryption. In: Proceedings of the International Conference on Theory and Application of Cryptographic Techniques (EUROCRYPT) (2000)
11. Huang, Y., Chapman, P., Evans, D.: Privacy-Preserving Applications on Smartphones. In: Proceedings of the USENIX Workshop on Hot Topics in Security (2011)
12. Huang, Y., Evans, D., Katz, J.: Private Set Intersection: Are Garbled Circuits Better than Custom Protocols? In: Proceedings of the isoc Network and Distributed Systems Security (NDSS) Symposium (2012)
13. Huang, Y., Evans, D., Katz, J., Malka, L.: Faster Secure Two-Party Computation Using Garbled Circuits. In: Proceedings of the USENIX Security Symposium (2011)
14. Kerschbaum, F., Schropfer, A., Dahlmeier, D., Biswas, D.: On the Practical Importance of Communication Complexity for Secure Multi-Party Computation Protocols. In: Proceedings of the ACM Symposium on Applied Computing (SAC) (2009)
15. Kreuter, B., Shelat, A., Shen, C.h.: Towards Billion-Gate Secure Computation with Malicious Adversaries (2012), university of Virginia Manuscript
16. Kruger, L., Jha, S., Goh, E.J., Boneh, D.: Secure Function Evaluation with Ordered Binary Decision Diagrams. In: Proceedings of the ACM conference on Computer and communications security (CCS) (2006)
17. Kustans, E.: Shark for root. https://play.google.com/store/apps/details?id=lv.n3o.shark (2012)
18. Lindell, Y., Pinkas, B.: A Proof of Yao's Protocol for Secure Two-Party Computation. In: Journal of Cryptology. vol. 22, pp. 161–188 (2009)
19. Malkhi, D., Nisan, N., Pinkas, B., Sella, Y.: Fairplay – A Secure Two-Party Computation System. In: Proceedings of the USENIX Security Symposium (SECURITY) (2004)
20. Mood, B., Letaw, L., Butler, K.: Memory-Efficient Garbled Circuit Generation for Mobile Devices. In: Proceedings of the IFCA International Conference on Financial Cryptography and Data Security (FC). Bonaire (Feb 2012)
21. Nakamura, T., Inenaga, S., Ikeda, D., Baba, K., Yasuura, H.: Anonymous Authentication Systems Based on Private Information Retrieval. In: International Comference on Networked Digital Technologies (NDT) (2009)
22. Naor, M., Pinkas, B.: Efficient oblivious transfer protocols. In: Proceedings of the ACM-SIAM symposium on Discrete algorithms (2001)
23. Ostrovsky, R., III, W.E.S.: Private Searching On Streaming Data. Journal of Cryptology 20(4), 397–430 (2007)

24. Ramachandran, A., Zhou, Z., Huang, D.: Computing Cryptographic Algorithms in Portable and Embedded Devices. In: Proceedings of the IEEE International Conference on Portable Information Devices (PORTABLE) (2007)
25. The Wireshark Foundation: Wireshark. https://www.wireshark.org/ (2012)
26. Yao, A.C.: How to generate and exchange secrets. In: Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS) (1986)
27. Zhong, G., Goldberg, I., Hengartner, U.: Louis, Lester and Pierre: Three Protocols for Location Privacy. In: Privacy Enhancing Technologies Symposium (2007)

A Network Usage Experiment Results

In addition to our primary metric of average execution time, we examined the total network traffic exchanged for each SFE technique in both test applications. To capture the amount of data exchanged, we used the "Shark for Root" Android application to capture network traffic [17], then examined the data in Wireshark [25]. The graphs below show the results for the proximity test application (Figure 5) and the private set intersection application (Figures 6, 7). For PSI, the pipelined circuit evaluation technique only accepted input sizes that were powers of two. For this reason, we show the results in a separate graph. It is important to note that for all garbled circuit techniques except the pipelined circuit evaluation, these graphs do not include the data required to initially send the circuit from the generating party to the evaluating party. Depending on the application and input size, these circuit files could be as large as 935 KB.

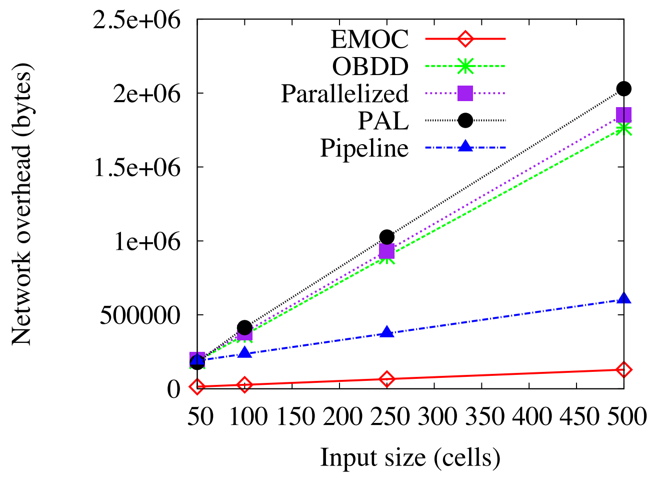


Fig. 5: Proximity test network usage. Note that even the most optimized garbled circuit evaluation technique requires over four times the amount of network traffic used by EMOC.

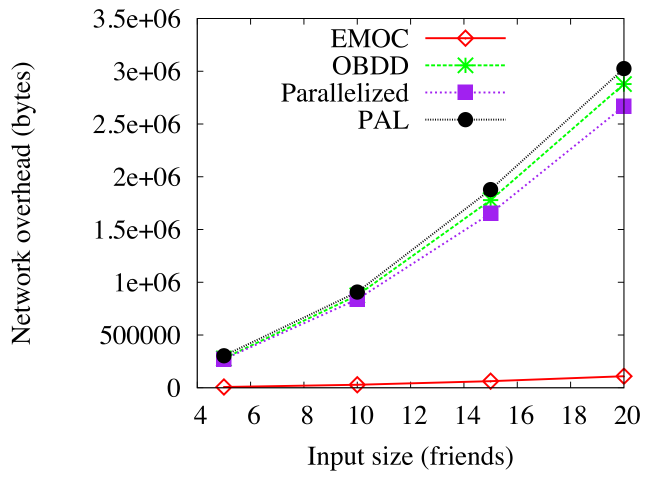


Fig. 6: Private set intersection network traffic without initial garbled circuit exchange. For every garbled circuit technique shown in this graph, we were only able to run experiments up to inputs of size 20 due to the large memory requirements of Fairplay.

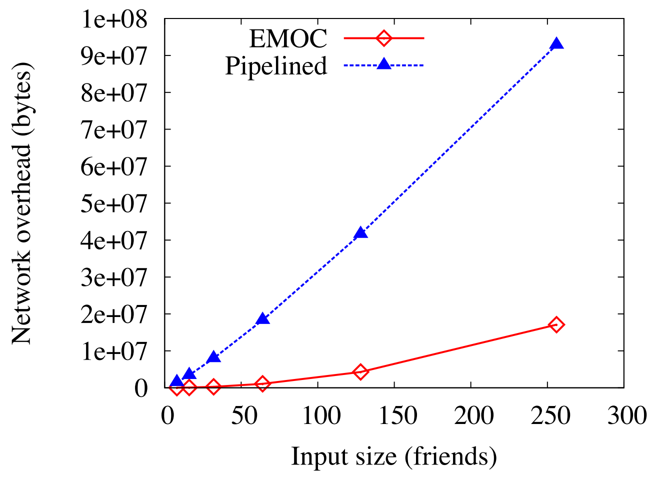


Fig. 7: Private set intersection total network traffic for large inputs. This figure shows the network usage for large inputs in our scheme and the pipelined evaluation scheme.