

Outsourcing Secure Two-Party Computation as a Black Box

Abstract. Secure multiparty computation (SMC) offers a technique to preserve functionality and data privacy in mobile applications. Current protocols that make this costly cryptographic construction feasible on mobile devices securely outsource the bulk of the computation to a cloud provider. However, these outsourcing techniques are built on specific secure computation assumptions and tools, and applying new SMC ideas to the outsourced setting requires the protocols to be completely rebuilt and proven secure. In this work, we develop a generic technique for lifting any secure two-party computation protocol into an outsourced two-party SMC protocol. By augmenting the function being evaluated with auxiliary consistency checks, we can create an outsourced protocol with low overhead cost. Our implementation and evaluation show that in the best case, our outsourcing additions execute within the confidence intervals of two servers running the same computation, and consume approximately the same bandwidth. In addition, the mobile device itself uses minimal bandwidth over a single round of communication. This work demonstrates that efficient outsourcing is possible with any underlying SMC scheme, and implements an outsourcing protocol that is efficient and directly applicable to current and future SMC techniques.

1 Introduction

As the mobile computing market continues to grow, an increasing number of mobile applications are requiring users to provide personal or context-sensitive information. However, as the recent iCloud breach demonstrates [27], these application servers cannot necessarily be trusted to maintain the security of the data they possess. To better preserve privacy and the functionality of mobile applications, secure multiparty computation (SMC) techniques offer protocols that allow application servers to process user data while it remains encrypted. Unfortunately, while a plethora of SMC techniques exist, they currently require too much processing power and device memory to be practical on the mobile platform. Furthermore, the bandwidth and power requirements for these SMC protocols will always be a limiting requirement for mobile applications even as the computational resources of mobile devices grow.

To bring SMC to the mobile platform in a more efficient way, recent work has focused on developing secure techniques for outsourcing the most expensive computation. Rather than naively trusting the Cloud to stand in for the mobile device in a standard SMC protocol, these outsourced protocols seek to use the Cloud for computation without revealing any input or output values. A number of these protocols have been specifically developed to outsource garbled circuit protocols [26, 9, 8]. These protocols attempt to optimize the outsourcing operations without increasing the complexity of the circuit being evaluated. However, because of this optimization goal, they are constructed and proven secure using specific garbled circuit evaluation techniques. As

new techniques for SMC are developed that modify the garbled circuit construction (or use completely different underlying constructions), it is unclear whether these specific outsourcing protocols will be able to take advantage of the new developments.

In this work, we develop a technique for outsourcing secure two-party computation for *any* two-party SMC technique. Rather than avoiding changes to the function being evaluated, we add a small amount of overhead to the evaluated function itself. This tradeoff allows for an outsourcing scheme that relies on the underlying two-party protocol in a black box manner, meaning the underlying protocol can be swapped for any other protocol meeting the same definition of security. This makes the task of securely incorporating newly developed SMC techniques trivial. This protocol enables mobile devices to participate in *any secure two-party SMC protocol* with minimal cost to the device and with nominal overhead to the servers running the computation. Specifically, we make the following contributions:

- **Develop a black box outsourcing protocol:** We develop and prove the security of a novel outsourcing technique for lifting any two-party SMC protocol into the two-party outsourced setting. To do this, we add a small amount of overhead to the function being evaluated to ensure that none of the inputs are modified by malicious participants. Unlike previous generic outsourcing approaches [24], this allows for *any* SMC protocol to be outsourced, not strictly reactive SMC protocols.
- **Implement and evaluate the overhead cost of the outsourcing operations:** Using the garbled circuit protocol of Shelat and Shen [43], we implement our protocol and evaluate the overhead cost of outsourcing. Rather than compare to previous outsourcing schemes, we instead measure the overhead incurred by augmenting the desired functionality. This measurement of cost better represents the value of the scheme, as a direct comparison to previous outsourcing protocols would drastically change depending on the underlying two-party SMC protocol used in our scheme. Our results show that for large circuits, black box outsourcing incurs negligible overhead (i.e., the confidence intervals for outsourced and server only execution intersect) in evaluation time and in bandwidth required when compared to evaluating the unmodified function.
- **Develop a practical mobile SMC application:** To demonstrate the practical performance of our protocol, we develop a mobile-specific facial recognition application and analyze its performance. Building on the Scifi protocol [40] for server based facial recognition, we motivate a use case for a mobile version and empirically show that our outsourcing protocol makes this application practically possible.

The rest of this work is organized as follows: Section 2 describes related research, Section 3 outlines definitions of security, Section 4 formally defines the protocol, Section 5 describes our implementation and performance evaluation, Section 6 presents a new mobile-specific application for SMC, Section 7 compares the overhead of our black box technique to previous work, and Section 8 provides concluding remarks.

2 Related Work

Since it was initially conceived in the early 1980's [44, 17], secure multiparty computation (SMC) has grown from a theoretical novelty to a potentially useful and practical

cryptographic construction. The FairPlay implementation [35] provided one of the first schemes for performing secure multiparty computation in practice. Since then, a number of other protocols and implementations have shown that privacy-preserving computation in the semi-honest threat model can be performed relatively efficiently [22, 5, 2]. However, this security model is weak in practice, and does not provide enough security for most real-world situations. To resolve this, recent study has focused on developing protocols that are secure in the malicious setting. For two-party computation, the garbled circuit construction has seen a large amount of new development [32, 33, 36, 41, 29, 42, 43] that has drastically reduced the cost of circuit checking and the associated consistency verification. Because the cut-&-choose construction that is typically applied in this setting is very costly, recent work has sought to minimize the cost of the cut-&-choose [13, 31, 20] or amortize that cost over a batch of circuit executions [34, 23]. Besides the garbled circuit technique, other techniques using somewhat homomorphic encryption [11, 10] and oblivious transfer [39] have shown promise of producing efficient protocols for secure multiparty computation in the malicious threat model. However, all of these techniques still have significant overhead cost that makes them infeasible to execute without sizable computational resources.

With smartphone applications retrieving private user data at an increasing rate, secure multiparty computation could potentially offer a way to maintain privacy and functionality in mobile computing. However, the efficiency challenges of SMC are compounded when considered in the resource-constrained mobile environment. Previous work has shown that smartphones are generally limited to simple functions in the semi-honest setting [7, 21]. Demmler et al. [12] showed how to incorporate pre-computation on hardware tokens to improve efficiency on mobile devices, but still in the semi-honest setting. In addition to the cost of evaluating these SMC protocols, Mood et al. [38] and Kreuter et al. [30] demonstrated that even with significant optimization, the task of compiling circuits on the mobile device can also be quite costly.

Given these limitations, evaluating SMC protocols directly on mobile hardware does not seem possible in the immediate future. Because of this, mobile secure computation research has recently focused on applying techniques from server-assisted cryptography [4] to move the most costly cryptographic operations off of the mobile device and onto a more capable Cloud server. To achieve this, many authors have focused on developing protocols for outsourcing secure computation of specific algorithms such as graph algorithms [6], set intersection [28], and linear algebra functions [3]. The first protocol to outsource SMC for any function was developed by Kamara et al. [25, 26]. In this work, the authors established a definition of security that assumes specific parties in the computation, while malicious, are not allowed to collude. Following on this definition, several other protocols and efficiency improvements have been developed for the outsourced setting [9, 37, 8]. Unfortunately, these protocols are built on specific secure multiparty computation assumptions and techniques. With new and varying techniques for SMC being developed at a rapid pace, it is unclear how to apply the outsourcing techniques used in these protocols to new schemes to allow them to benefit from new efficiency improvements. In this work, we seek to develop a protocol that can lift *any* two-party SMC protocol into the outsourced setting with little overhead.

In concurrent work to our own, Jakobsen et al. [24] develop a framework for outsourcing secure computation that is similar to our protocol. However, their protocol requires more specific properties in the underlying SMC protocol, where our protocol is designed to be truly generic. Furthermore, our work provides an implementation, an empirical performance analysis, and develops a mobile-specific application with a performance characterization. We examine the tradeoffs between these two protocols in Section 7.

3 Background

Outsourced two-party SMC protocols are designed to allow two parties of asymmetric computational capability to engage in a privacy-preserving computation with the assistance of an outsourcing party. We consider the situation where a mobile device possessing limited computational resources wishes to run an SMC protocol with an application server or other well-provisioned entity. To allow this, outsourcing protocols move the majority of the costly operations off of the mobile device and onto a Cloud provider *without* revealing to the Cloud either party’s input or output to the computation. These protocols aim to provide security guarantees of privacy and correctness, and also attempt to minimize the computation required at the mobile device while still maintaining efficiency between the application server and the Cloud. To meet these goals in the outsourced setting, a number of careful security assumptions must be made.

3.1 Two-party SMC security

Our black box protocol is based on the execution of a two-party SMC protocol to obliviously compute the result. We make no assumptions about the techniques used or structure of this underlying protocol except that it meets the canonical definition of security against malicious adversaries using the ideal/real world paradigm [16]. Informally, this states that for any adversary participating in the two-party SMC protocol, there exists a simulator in an ideal world with a trusted third party running the computation where the output in both worlds is computationally indistinguishable. In this definition, the simulator in the ideal world is given oracle access to the malicious adversary in the real world, and must simulate both the adversary’s view of a real execution *and* the adversary’s arbitrary input to the computation. A secure two-party SMC protocol meeting this definition of security provides two guarantees. The first is privacy, which means that a malicious adversary cannot learn anything about the other party’s input or output value beyond what is revealed by his own output value. The second guarantee is correctness. This implies that even in the presence of a malicious adversary, the output of the protocol will be the correct output of the agreed upon function except with negligible probability. For a formal definition of security and further discussion, refer to [16].

3.2 Collusion assumptions

Previous work in outsourcing SMC makes careful assumptions about who in the computation is allowed to collude. Kamara et al. [26] discuss at length the theoretical justification for these assumptions. Essentially, to achieve an n -party outsourcing protocol with

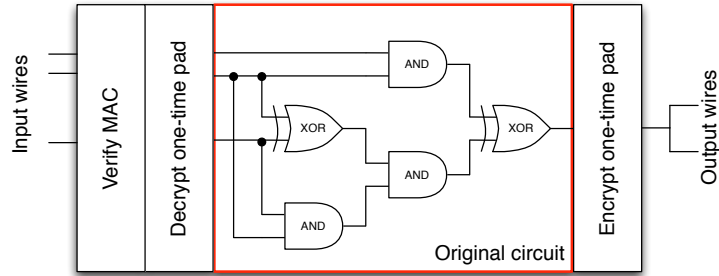


Fig. 1. The process of augmenting a circuit for outsourcing. The original circuit is boxed in red. Essentially, we require that the mobile device’s input be verified using a MAC and decrypted using a one-time pad before it is input into the function. After the result is computed, it must be re-encrypted using a one-time pad and delivered to *both* parties to guarantee that the mobile device will detect if either party tampers with the result.

better complexity than a two-party SMC protocol, it must be assumed that the Cloud (i.e., the server aiding computation but not providing input to the function) cannot collude with any other party. Other outsourcing protocols have sought ways to relax this restriction without significantly increasing the complexity of the evaluated function [9, 8]. However, these protocols still assume that the application server and the Cloud cannot collude. We follow this assumption in our construction. As stated by Kamara et al., the existence of an outsourcing protocol where this particular collusion is allowed would imply an efficient two-party SMC scheme where one party performs work that is sub-linear with respect to the size of the function being evaluated. While there are techniques for such a two-party SMC protocol, it is either unclear how they can be applied to create an outsourced protocol [18], or they are prohibitively inefficient [15].

Formally, we follow the security definition established by Kamara et al. [26] but specified for the two-party scenario as in the work of Carter et al. [9, 8]. We provide a summary of the definition in Appendix A, and refer the reader to previous work for a complete discussion.

4 Protocol

In this section, we outline our black box outsourcing protocol and discuss the security guarantees provided.

4.1 Participants

- **SERVER:** the application or web server participating in a secure computation with the mobile device. This party provides input to the function being evaluated.
- **MOBILE:** the mobile device accessing SERVER to jointly compute some result. This party also provides input to the function being evaluated.

- **CLOUD** a Cloud computation provider tasked with assisting **MOBILE** in the expensive operations of the secure computation. This party executes a two-party SMC protocol in a black box manner with **SERVER**, but does not provide an input to the function being evaluated.

4.2 Protocol

The outsourcing protocol can be informally described as follows: first, **MOBILE** prepares its input by encrypting with a one-time pad and producing a MAC tag for verifying the input is not tampered with before it is entered into the computation. Since **SERVER** and **CLOUD** are assumed not to collude, one party receives the encrypted input, and the other party receives the decryption key. Both of these values are input into the secure two-party computation, and are verified during this evaluation using the associated MAC tags (see Figure 1). If the check fails, the protocol outputs a failure message. Otherwise, the second phase of the protocol evaluates the actual SMC program. The third and final phase encrypts and outputs **MOBILE**'s result to both parties, who in turn deliver these results back to **MOBILE**. Intuitively, since our security model assumes that **SERVER** and **CLOUD** are never simultaneously malicious, at least one of these two will return the correct result to **MOBILE**. From this, **MOBILE** will detect any tampering from the malicious party by a discrepancy in these returned values, eliminating the need for an output MAC. If no tampering is detected, **MOBILE** then decrypts the output of computation. We provide a formal protocol description in Appendix B.

Security Our protocol intuitively provides both correctness and privacy for the **MOBILE** input and output based on the underlying components. Privacy is achieved based on the security of the underlying SMC protocol as well as the input and output one-time pads. Correctness is based on three main points. The MAC evaluated within the circuit ensures that the **MOBILE** input is correct. The correctness of the circuit evaluation itself is guaranteed by the underlying SMC protocol. Finally, the correctness of the output is ensured by the security model assumption that at least one of the **SERVER** and **CLOUD** are behaving semi-honestly. Thus, any tampering with the **MOBILE** output by the malicious party will be detected when compared to the output provided by the semi-honest party. For a formal simulation proof of security, refer to our technical report [1].

5 Performance Evaluation

To demonstrate the practical efficiency of our black box outsourcing protocol, we implemented the protocol and examined the actual overhead incurred by the overhead operations. We initially considered comparing our black box protocol to existing implementations of outsourcing protocols [26, 9, 8]. However, these existing protocols are built on fixed underlying SMC techniques. As new protocols for two-party SMC are developed, the plug-and-play nature of our protocol allows for these new techniques to be applied, which would provide a different comparison for each underlying protocol. Instead, we chose to compare the overhead execution costs of our black box protocol to performing the same computation in the underlying two-party protocol. Because the mobile device

Program Name	SS13 Total	BB Total	Increase	SS13 Non-XOR	BB Non-XOR	Increase
Dijkstra10	259,232	456,326	1.8x	118,357	179,641	1.5x
Dijkstra20	1,653,542	1,949,820	1.2x	757,197	849,445	1.1x
Dijkstra50	22,109,732	22,605,018	1.0x	10,170,407	10,324,317	1.0x
MatrixMult3x3	424,748	1,020,196	2.4x	161,237	345,417	2.1x
MatrixMult5x5	1,968,452	3,360,956	1.7x	746,977	1,176,981	1.6x
MatrixMult8x8	8,069,506	11,354,394	1.4x	3,060,802	4,075,082	1.3x
MatrixMult16x16	64,570,969	77,423,481	1.2x	24,494,338	28,458,635	1.2x
RSA128	116,083,727	116,463,648	1.0x	41,082,205	41,208,553	1.0x

Table 1. Comparing the original function size to the augmented outsourcing circuit. As the size of the circuit grows, the increase in gates incurred by outsourcing becomes vanishingly small.

computation requires seconds or less to execute, we focus our attention on the cost at the two executing servers. This performance analysis demonstrates two key benefits of our protocol. First, it gives a rough overhead cost for an entire class of two-party SMC protocols (in our case, garbled circuit protocols). Second, it allows us to demonstrate that our outsourcing technique allows a mobile device with restricted computational capability to participate in a privacy-preserving computation in approximately the same amount of time as the same computation performed between two servers. Essentially, we show that our protocol provides a mobile version of any two-party SMC protocol with nominal overhead cost to the servers. This is a novel evaluation methodology not used to evaluate previous black box SMC constructions, and provides a more intuitive estimate for performance when applying a new underlying SMC construction.

5.1 System Design

Our implementation of the black box outsourcing protocol uses the two-party garbled circuit protocol developed by Shelat and Shen [43] as the underlying two-party SMC protocol. We selected this protocol because it is among the most recently developed garbled circuit protocols and it has the most stable public release. We emphasize that it is possible to implement our outsourcing on *any* two-party SMC protocol, such as the recent protocols developed to reduce the cost of cut-&-choose [20, 31]. We implement our MAC within the augmented circuit using AES in cipher-block chaining mode (CBC-MAC), as the AES circuit is well-studied in the context of garbled circuit execution. This MAC implementation adds an invocation of AES per 128-bit block of input. Using the compiler developed by Kreuter et al. [29], the overhead non-XOR gate count in the augmented circuit based on input size is $(\frac{|x|15686}{128})$ for input x . We provide exact gate counts with overhead measurements for each tested application in Table 1. Our code will be made available upon publication.

Testbed Our experiments were run on a single server equipped with 64 cores and 1 TB of RAM. For each execution, the application server and cloud were run as 32 processes communicating via the Message Passing Interface (MPI) framework. The mobile device was a Samsung Galaxy Nexus with a 1.2 GHz dual-core ARM Cortex-A9 processor and 1 GB of RAM, running Android version 4.0. The mobile device communicated with the

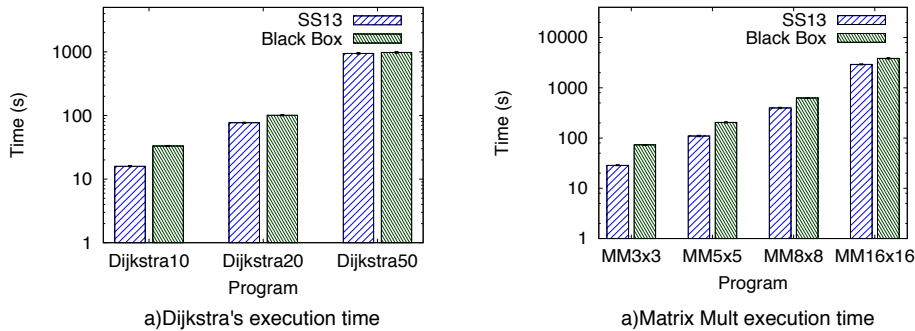


Fig. 2. Dijkstra and Matrix multiplication execution time in seconds. Note that the execution overhead diminishes even as the mobile input size increases.

test server over an 802.11n wireless connection in an isolated network environment. We ran each experiment 10 times and averaged the results with 95% confidence intervals.

Test applications We selected a representative set of test applications from previous literature [8, 29, 43, 30] to examine the performance of our protocol over varying circuit and input sizes. We use all applications as implemented by Kreuter et al. [29] except for Dijkstra's algorithm, which was implemented by Carter et al. [9].

1. Dijkstra: this application takes a weighted graph from one party and two node indices from the other party (i.e., start and end nodes), and calculates the shortest path through the graph from the start to the end node. We consider n -node graphs with 16 bit edge weights, 8 bit node identifiers, and a maximum node degree of 4. We chose this problem as a representative application for the mobile platform.
2. Matrix Multiplication: this application accepts a matrix from both parties and outputs the matrix product. We consider this application for input size n , where each matrix is an $n \times n$ matrix of 32-bit integers. This test application demonstrates protocol behavior for increasing input sizes.
3. RSA: this application accepts a modulus N and an exponent e from one party, and a message x from the other party, and computes the modular exponentiation $x^e \bmod N$. We consider input values where each value is 128 bits in length. While this is certainly too short for secure practical use, the size of the circuit provides a good benchmark for evaluating extremely large circuits.

5.2 Execution Time

With the mobile operations minimized to $O(|x| + |o_m|)$ symmetric key operations, our experiments showed a diminishing cost of server overhead as the size of the test application increased. Considering Dijkstra's algorithm in Figure 2 shows that for a graph of 10 nodes, the outsourcing operations incur a 2.1x slowdown from running the protocol between two servers. However, as the number of graph nodes increases to 50, the confidence intervals for outsourced and server-only execution overlap, indicating a

Program Name	SS13 (ex)	BB (ex)	Increase (ex)	SS13 (bw)	BB (bw)	Increase (bw)
Dijkstra10	16 ± 1%	33 ± 1%	2.1x	2.44 x 10 ⁹	3.87 x 10 ⁹	1.6x
Dijkstra20	77 ± 1%	100 ± 1%	1.3x	1.52 x 10 ¹⁰	1.73 x 10 ¹⁰	1.1x
Dijkstra50	940 ± 2%	980 ± 2%	1.0x	2.02 x 10 ¹¹	2.05 x 10 ¹¹	1.0x
MatrixMult3x3	28.6 ± 0.8%	73.2 ± 0.5%	2.6x	3.43 x 10 ⁹	7.66 x 10 ⁹	2.2x
MatrixMult5x5	110 ± 2%	200 ± 2%	1.9x	1.57 x 10 ¹⁰	2.56 x 10 ¹⁰	1.6x
MatrixMult8x8	400 ± 2%	627 ± 0.9%	1.6x	6.43 x 10 ¹⁰	8.73 x 10 ¹⁰	1.4x
MatrixMult16x16	2900 ± 1%	3800 ± 2%	1.3x	5.11 x 10 ¹¹	6.01 x 10 ¹¹	1.2x
RSA128	4700 ± 2%	4900 ± 3%	1.0x	8.69 x 10 ¹¹	8.72 x 10 ¹¹	1.0x

Table 2. Comparing SS13 and Black Box execution time (ex) and bandwidth use (bw). All times in seconds and bandwidth in bytes. Note that as the circuit size increases, the increase in execution time caused by outsourcing becomes insignificant. Likewise, as this circuit grows in size, the overhead bandwidth required for outsourcing is amortized

virtually non-existent overhead cost. When we compare these results to the gate counts shown in Table 1, we see that as the gate count for the underlying protocol increases, the additive cost of running the input MAC and output duplication amortize over the total execution time. This is to be expected from our predicted overhead of 15686 non-XOR gates for each CBC-MAC block in the input. However, since the mobile input for Dijkstra’s algorithm is of a fixed size, we observe that increasing the application server input size does not add to the outsourcing overhead, showing the black box protocol to be more efficient for large circuit sizes with small mobile input.

When we consider a growing mobile input size, we observe the overhead cost of the MAC operation performed on the mobile input. In the matrix multiplication test program, we observed a 2.6x slowdown for the smallest input size of a 3 × 3 matrix (Figure 2). As in the previous experiment, this overhead diminished to a 1.3x slowdown for the largest input size, but diminished at a slower rate when compared to the circuit size. This is a result of additional AES invocations to handle the increasing mobile input size. However, the reduction in overhead shows that even as input sizes increase, the circuit size is still the main factor in amortizing overhead.

In our final experiment, we considered a massive circuit representing one of the most complex garbled circuit programs evaluated to date. When comparing the outsourced execution to a standard two-party execution, the overhead incurred by the outsourcing operations is almost non-existent, as shown in Table 2. This experiment confirms the trends of diminishing overhead cost observed in the previous two experiments. From this and previous work, we know that evaluating large circuits from mobile devices is *not possible* without outsourcing the bulk of computation. Given that many real-world applications will require on the order of billions of gates to evaluate, this experiment shows that our black box outsourcing technique allows mobile devices to participate in secure two-party computation at roughly the same efficiency as two server-class machines executing the same computation.

5.3 Bandwidth

Because transmitting data from a mobile device is costly in terms of time and power usage, we attempted to minimize the amount of bandwidth required from the mobile de-

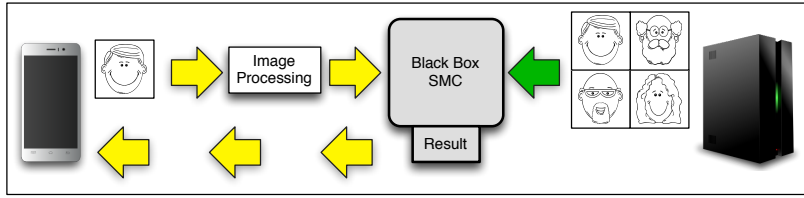


Fig. 3. An example of the facial recognition application.

vice. Our protocol requires only $2(|x| + 2k) + 4(|o_m|)$ bits to be transmitted to and from the mobile, where x is the mobile input, o_m is the mobile output, and k is the security parameter. For the RSA circuit, this would amount to 136 bytes of bandwidth used. To perform the same computation without outsourcing, Carter et al. [9] show that several gigabytes of bandwidth would be required if the mobile device possessed enough memory to perform the computation at all. Because our mobile bandwidth is nearly minimal and easily calculated for any program, we focused our evaluation on measuring the bandwidth overhead incurred between the application server and the Cloud.

As in the case of execution time, Table 2 shows an inverse relation between circuit size and overhead cost. Before running the experiment, we predicted that the bandwidth overhead would approximately match the overhead in circuit size shown in Table 1. The experiments confirmed that the actual bandwidth overhead was equal to or slightly larger than the overhead in non-XOR gates in the circuit. The reason for this correlation is twofold. First, the free-XOR technique used in the shelat-Shen protocol allows XOR gates to be represented without sending any data over the network. Thus, adding additional XOR gates does not incur bandwidth cost. Second, in cases where the actual overhead is slightly larger than the circuit size overhead, we determined that the added cost was a result of additional oblivious transfers. These operations require the transmission of large algebraic group elements, so the test circuits which incurred increased overhead from the growth of the mobile input showed a slightly larger bandwidth overhead as well. Ultimately, as in the case of execution time, our experiments demonstrate that black box outsourcing incurs minimal bandwidth usage at the mobile device with diminishing bandwidth overhead between the application server and the Cloud.

6 Application: Facial Recognition

The growing number of mobile applications available present a wealth of potential for applying privacy-preserving computation techniques to the mobile platform. Carter et al. [9] demonstrated one potential application with their privacy-preserving navigation app, and Mood et al. [37] presented a friend-finding application. We present a third mobile-specific application: facial recognition. In this setting, a secret operative or law enforcement agent carrying a mobile device needs to analyze a photo of a suspected criminal using an international crime database (see Figure 3). The database, managed by an international organization, would compare the photo to their database in a privacy-preserving manner, returning a match if the suspect appears in the database. In this

scenario, the agent must keep the query data private to prevent insiders from learning who is being tracked, and the international organization must keep the database private from agents associated with any particular nation.

To implement this application, we use the facial recognition techniques developed for the Scifi protocol of Osadchy et al. [40]. They develop a technique for two servers to perform efficient facial recognition using discrete parameters, which can more easily be manipulated in secure computation protocols. They combine machine learning techniques in a preprocessing phase with a secure online phase that compares the hamming distance of photos represented as bit strings. To demonstrate our application, we implement the online comparison phase of this protocol in our black box outsourcing protocol (the $F_{threshold}$ function in their work). The mobile device provides a 928 bit representation of a photo, while the application server provides a database of representations containing 10, 100, and 1000 faces.

Our results show that given a database of 10 faces, the outsourced protocol can run the online phase in approximately 87 seconds. For 100 and 1000 face databases, we observed execution times of 170 seconds and 1000 seconds, respectively. As the size of the facial database increases, the execution time for comparing across the entire database grows. This growing cost is a result of the large cost of representing the facial database as garbled input. Provided with a two-party SMC protocol that more efficiently computes over large data sets, our black box protocol could be used to move this application from feasible to practical. This demonstrates that an application designed and implemented to run between two servers can be feasibly executed from a mobile device. As new, more heavyweight applications are developed, our technique for outsourcing allows any of those applications to be executed from a mobile device with comparable efficiency to the server platform.

7 Comparison to Previous Techniques

While our implementation and evaluation in the previous section represents the first empirical analysis of black box outsourcing, two other protocols have been proposed in the literature, which we term KMR [26] and JNO [24]. We evaluate the tradeoffs between each technique in this section.

KMR While the main focus of their work is the fixed outsourcing protocol Salus [26], Kamara et al. sketch a black box technique for outsourcing any two-party computation protocol. Essentially, their protocol encodes each bit of the MOBILE input as a bit string of length k for some computational security parameter k . This encoded input, along with the mappings for reversing this encoding, is secret shared between SERVER and CLOUD, and then restored using only XOR gates inside the circuit. A similar encoding technique is used to maintain both privacy and integrity of the output from the circuit. This technique has the advantage of adding only XOR gates to the circuit, which can be transmitted and evaluated cheaply using many SMC techniques. However, it also requires that the mobile input and output be expanded by a factor of k . By contrast, our evaluation demonstrates that the overhead caused by adding AND gates to the computation is minimal, and the MOBILE bandwidth use is kept to $O(|x| + |o_m|)$ with small

constant multiples. This is particularly advantageous on smartphones, where data usage is often restricted by slow network speeds or provider-imposed bandwidth caps.

JNO Developed concurrently to our protocol, Jakobsen et al. [24] presented a framework for outsourcing SMC protocols across any number of “worker” servers. Their protocol follows a similar procedure to our own, but they describe a novel MAC construction that allows the MOBILE input to be checked using only inexpensive linear operations in the circuit. Essentially, their technique requires that the MAC key be committed at the start of the protocol, then opened once the rest of the input values are committed to the computation. Once the key is opened, it can be multiplied as a known constant with the MOBILE input, which is secret shared according to the underlying SMC protocol. A simple multiplicative MAC can then be verified before computation continues. The advantage of this scheme is that it does not incur the k factor expansion of KMR while still adding only linear operations to the underlying circuit (e.g., XOR for boolean circuits). The tradeoff is that the underlying SMC protocol must allow for reactive computation (i.e., private values can be opened in the middle of computation). While this property is common in secret-sharing SMC protocols, it is difficult to achieve with garbled circuits. The generic technique for making garbled circuits into a reactive SMC protocol requires additional, MAC operations inside the circuit [19]. More efficient reactive garbled circuit protocols exist [14, 37], but require special constructions that cannot be combined with all garbled circuit protocols in a generic way. Our protocol allows for true black box outsourcing of *any* SMC protocol (reactive or non-reactive), and our empirical performance evaluation demonstrates that the overhead of adding AND operations to the circuit is minimal when the circuit size is large. This setting is preferable for computation that is more efficiently evaluated using garbled circuits than arithmetic secret-sharing SMC schemes. Furthermore, Jakobsen et al. provide no empirical analysis or application study for their outsourcing technique, providing only a proof of concept with limited insight into the practicality of black box SMC outsourcing.

8 Conclusion

The growing popularity of the mobile platform is creating a strong need for privacy-preserving computation in mobile applications. However, as most SMC techniques currently require significant processing and bandwidth resources, secure outsourcing protocols have been developed to assist mobile devices in performing the most expensive cryptographic operations associated with these protocols. In this work, we develop a technique for outsourcing any two-party SMC protocol in a black box manner. Our protocol securely offloads the cost of the SMC protocol to the Cloud, providing maximal efficiency to the mobile device while maintaining strong security guarantees. Our performance evaluation shows that as the complexity of the program being evaluated increases, the cost of outsourcing diminishes. As a result, we enable execution of any SMC protocol from a mobile device at approximately the same efficiency as running the protocol between two servers.

References

1. ANONYMIZED: Outsourcing secure two-party computation as a black box
2. Asharov, G., Lindell, Y., Schneider, T., Zohner, M.: More efficient oblivious transfer and extensions for faster secure computation. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security (2013)
3. Atallah, M.J., Frikken, K.B.: Securely outsourcing linear algebra computations. In: Proceedings of the ACM Symposium on Information, Computer and Communications Security (ASIACCS) (2010)
4. Beaver, D.: Server-assisted cryptography. In: Proceedings of the workshop on New security paradigms (NSPW) (1998)
5. Bellare, M., Hoang, V.T., Rogaway, P.: Foundations of garbled circuits. In: Proceedings of the ACM Conference on Computer and Communications Security (2012)
6. Blanton, M., Steele, A., Alisagari, M.: Data-oblivious graph algorithms for secure computation and outsourcing. In: Proceedings of the ACM SIGSAC Symposium on Information, Computer and Communications Security (2013)
7. Carter, H., Amrutkar, C., Dacosta, I., Traynor, P.: For your phone only: custom protocols for efficient secure function evaluation on mobile devices. *Journal of Security and Communication Networks (SCN)* 7(7), 1165–1176 (2014)
8. Carter, H., Lever, C., Traynor, P.: Whitewash: Outsourcing Garbled Circuit Generation for Mobile Devices. In: Proceedings of the Annual Computer Security Applications Conference (ACSAC) (2014)
9. Carter, H., Mood, B., Traynor, P., Butler, K.: Secure Outsourced Garbled Circuit Evaluation for Mobile Devices. In: Proceedings of the USENIX Security Symposium (2013)
10. Damgård, I., Keller, M., Larraia, E., Pastro, V., Scholl, P., Smart, N.P.: Practical covertly secure mpc for dishonest majority or: Breaking the spdz limits. In: *Computer Security–ESORICS* (2013)
11. Damgård, I., Pastro, V., Smart, N., Zakarias, S.: Multiparty computation from somewhat homomorphic encryption. In: *Advances in Cryptology–CRYPTO* (2012)
12. Demmler, D., Schneider, T., Zohner, M.: Ad-hoc secure two-party computation on mobile devices using hardware tokens. In: Proceedings of the USENIX Security Symposium (2014)
13. Frederiksen, T.K., Jakobsen, T.P., Nielsen, J.B., Nordholt, P.S., Orlandi, C.: Minilego: Efficient secure two-party computation from general assumptions. In: *Advances in Cryptology–EUROCRYPT* (2013)
14. Frederiksen, T.K., Nielsen, J.B.: Fast and maliciously secure two-party computation using the gpu. In: *Applied Cryptography and Network Security* (2013)
15. Gentry, C.: A fully homomorphic encryption scheme. Ph.D. thesis, Stanford University (2009)
16. Goldreich, O.: *Foundations of Cryptography: Volume 2 Basic Applications*. Cambridge Univ. Press (2004)
17. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game. In: Proceedings of the Annual ACM Symposium on Theory of Computing (1987)
18. Gordon, S.D., Katz, J., Kolesnikov, V., Labs, A.I.B., Krell, F., Raykova, M.: Secure Two-Party Computation in Sublinear (Amortized) Time. In: Proceedings of the ACM conference on Computer and communications security (CCS) (2012)
19. Hazay, C., Lindell, Y.: *Efficient Secure Two-Party Protocols: Techniques and Constructions*. Springer-Verlag (2010)
20. Huang, Y., Katz, J., Evans, D.: Efficient secure two-party computation using symmetric cut-and-choose. In: *Advances in Cryptology–CRYPTO* (2013)

21. Huang, Y., Chapman, P., Evans, D.: Privacy-preserving applications on smartphones. In: Proceedings of the USENIX Workshop on Hot Topics in Security (2011)
22. Huang, Y., Evans, D., Katz, J., Malka, L.: Faster Secure Two-Party Computation Using Garbled Circuits. In: Proceedings of the USENIX Security Symposium (2011)
23. Huang, Y., Katz, J., Kolesnikov, V.: Amortizing garbled circuits. In: Advances in Cryptology–CRYPTO (2014)
24. Jakobsen, T.P., Nielsen, J.B., Orlandi, C.: A framework for outsourcing of secure computation. In: Proceedings of the ACM Workshop on Cloud Computing Security (CCSW) (2014)
25. Kamara, S., Mohassel, P., Raykova, M.: Outsourcing multi-party computation. Cryptology ePrint Archive, Report 2011/272 (2011), <http://eprint.iacr.org/>
26. Kamara, S., Mohassel, P., Riva, B.: Salus: A system for server-aided secure function evaluation. In: Proceedings of the ACM conference on Computer and communications security (CCS) (2012)
27. Kelion, L.: Apple toughens icloud security after celebrity breach. <http://www.bbc.com/news/technology-29237469> (2014)
28. Kerschbaum, F.: Collusion-resistant outsourcing of private set intersection. In: Proceedings of the ACM Symposium on Applied Computing (2012)
29. Kreuter, B., shelat, a., Shen, C.: Billion-Gate Secure Computation with Malicious Adversaries. In: Proceedings of the USENIX Security Symposium (2012)
30. Kreuter, B., shelat, a., Mood, B., Butler, K.: PCF: A portable circuit format for scalable two-party secure computation. In: Proceedings of the USENIX Security Symposium (2013)
31. Lindell, Y.: Fast cut-and-choose based protocols for malicious and covert adversaries. In: Advances in Cryptology–CRYPTO (2013)
32. Lindell, Y., Pinkas, B.: An efficient protocol for secure two-party computation in the presence of malicious adversaries. In: Proceedings of the annual international conference on Advances in Cryptology (2007)
33. Lindell, Y., Pinkas, B.: Secure two-party computation via cut-and-choose oblivious transfer. In: Proceedings of the conference on Theory of cryptography (2011)
34. Lindell, Y., Riva, B.: Cut-and-Choose Yao-Based Secure Computation in the Online/Offline and Batch Settings. In: Advances in Cryptology?CRYPTO (2014)
35. Malkhi, D., Nisan, N., Pinkas, B., Sella, Y.: Fairplay—a secure two-party computation system. In: Proceedings of the USENIX Security Symposium (2004)
36. Mohassel, P., Riva, B.: Garbled circuits checking garbled circuits: More efficient and secure two-party computation. In: Advances in Cryptology–CRYPTO (2013)
37. Mood, B., Gupta, D., Butler, K., Feigenbaum, J.: Reuse it or lose it: More efficient secure computation through reuse of encrypted values. In: Proceedings of the ACM conference on Computer and communications security (CCS) (2014)
38. 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) (2012)
39. Nielsen, J.B., Nordholt, P.S., Orlandi, C., Burra, S.S.: A new approach to practical active-secure two-party computation. In: Advances in Cryptology –CRYPTO (2012)
40. Osadchy, M., Pinkas, B., Jarrous, A., Moskovich, B.: Scifi-a system for secure face identification. In: Proceedings of the IEEE Symposium on Security & Privacy (2010)
41. Pinkas, B., Schneider, T., Smart, N.P., Williams, S.: Secure two-party computation is practical. In: Advances in Cryptology–ASIACRYPT (2009)
42. shelat, a., Shen, C.H.: Two-output secure computation with malicious adversaries. In: Proceedings of the Annual international conference on Theory and applications of cryptographic techniques (2011)
43. shelat, a., Shen, C.H.: Fast two-party secure computation with minimal assumptions. In: Proceedings of the ACM conference on Computer and communications security (CCS) (2013)

44. Yao, A.C.: Protocols for secure computations. In: Proceedings of the Annual Symposium on Foundations of Computer Science (1982)

A Outsourced Security Definition

The real world setting is made up of three parties. Two of these parties provide input to the computation, while the third party takes on computational load for one of the two input parties. All three parties provide auxiliary random inputs to the protocol. Some subset of the three parties $A = (A_1, A_2, A_3)$ can behave maliciously, but we assume that the application server and the Cloud cannot collude. For the i^{th} honest party, OUT_i is defined as its output, and for the i^{th} corrupted party, OUT_i is its view of the protocol. Then we define the i^{th} partial output as:

$$REAL^{(i)}(k, x; r) = \{OUT_j : j \in H\} \cup OUT_i$$

Here, k is the security parameter, x is all inputs to the computed function, r is the auxiliary randomness, and H is the set of all honest parties.

The ideal world setting is made up of the same parties with the same inputs as the real world with the addition of a trusted third party that receives all parties' inputs, computes the desired function, and returns the output to all parties except the outsourced party that is not providing inputs to the function. Any party may abort the computation early or refuse to send input, in which case the trusted party sends no output. As in the standard two-party definition [16], it is possible for one party, upon receiving output from the trusted third party, to terminate the protocol, preventing the other party from receiving its output. For the i^{th} honest party, OUT_i is defined as its output received from the trusted party, and for the i^{th} corrupted party, OUT_i is an arbitrary output value. Then we define the i^{th} partial output in the presence of independent malicious simulators $S = (S_1, S_2, S_3)$ as:

$$IDEAL^{(i)}(k, x; r) = \{OUT_j : j \in H\} \cup OUT_i$$

Here, k, x, r , and H are defined as above. In this real/ideal world setting, outsourced security is defined as follows:

Definition 1 *An outsourcing protocol securely computes the function f if there exists a set of probabilistic polynomial-time (PPT) simulators $\{Sim_1, Sim_2, Sim_3\}$ such that for all PPT adversaries (A_1, A_2, A_3) , inputs x , and for all $i \in \{1, 2, 3\}$:*

$$\{REAL^{(i)}(k, x; r)\}_{k \in N} \stackrel{c}{\approx} \{IDEAL^{(i)}(k, x; r)\}_{k \in N}$$

Where $S = (S_1, S_2, S_3)$, $S_i = Sim_i(A_i)$, and r is uniformly random.

We provide a complete proof of security following this definition in our technical report [1].

B Protocol Description

Common Input: All parties agree on a computational security parameter k , a message authentication code (MAC) scheme $(Gen(), Mac(), Ver())$, and a malicious secure two-party computation protocol $2PC()$. All parties agree on a two-output function $f(x, y) \rightarrow f_m, f_s$ that is to be evaluated.

Private Input: MOBILE inputs x while SERVER inputs y . We denote the bit length of a value as $|x|$ and concatenation as $x||y$.

Output: SERVER receives f_s and MOBILE receives f_m .

1. **Input preparation:** MOBILE generates a one-time pad k_{fm} where $|k_{fm}| = |f_m|$. Mobile then generates two MAC keys $v_s = Gen(k)$ and $v_c = Gen(k)$. Finally, MOBILE generates a one-time pad k_m where $|k_m| = |x| + |k_{fm}|$.
2. **Input delivery:** MOBILE encrypts its input as $a = (x||k_{fm}) \oplus k_m$. It then generates two tags $t_s = Mac(a||v_c, v_s)$ and $t_c = Mac(k_m||v_s, v_c)$. MOBILE delivers a, v_c , and t_s to SERVER and k_m, v_s , and t_c to CLOUD.
3. **Augmenting the target function:** All parties agree on the following augmented function $g(y, a, v_c, t_s; k_m, v_s, t_c)$ to be run as a two-party SMC computation:
 - (a) If $Ver(a||v_c, t_s, v_s) \neq 1$ or $Ver(k_m||v_s, t_c, v_c) \neq 1$ output \perp .
 - (b) Set $x||k_{fm} = a \oplus k_m$
 - (c) Run the desired function $f_s, f_m = f(x, y)$
 - (d) Set output values $o_s = f_s$ and $o_m = f_m \oplus k_{fm}$
 - (e) Output $o_s||o_m$ to SERVER and o_m to CLOUD
4. **Two-party computation:** SERVER and CLOUD execute a secure two-party computation protocol $2PC(g()); y, a, v_c, t_s; k_m, v_s, t_c$ evaluating the augmented function.
5. **Output verification:** CLOUD delivers its output from the two-party computation, o_m to MOBILE. SERVER also delivers the second half of its output o'_m to MOBILE. MOBILE verifies that $o_m = o'_m$.
6. **Output recovery:** SERVER receives output $f_s = o_s$ and MOBILE receives output $f_m = o_m \oplus k_{fm}$