Practical Secure Two-Party Computation: Techniques, Tools, and Applications

David Evans (PI), Aaron Mackey, abhi shelat (University of Virginia)
Michael Hicks, Jonathan Katz (University of Maryland); Steven Myers (Indiana University)

Project Summary

Many compelling applications involve computations that require sensitive data from two or more individuals. As an example, consider the myriad genetics applications soon to be within reach of individuals as the cost of personal genome sequencing rapidly plummets. An individual may wish to compare her genome with the genomes of different groups of participants in a study to determine which treatment is likely to be most effective for her. Such comparisons could have tremendous value, but are infeasible because of the privacy concerns both for the individual and study participants. What is needed is a way to compute the output of the comparison without exposing either party's private inputs.

Theoretical solutions to this problem, known as *secure multi-party computation*, have been known for several decades, including a general solution developed by Andrew Yao based on *garbled circuits*. Because of its extensive memory use and computational cost, however, the garbled circuits approach has traditionally been considered more of a theoretical curiosity than a practical mechanism for building privacy-preserving applications. Recent developments in cryptographic techniques and new implementation approaches are beginning to change this, however, and admit the possibility of scalable, practical secure computation. Building on this work, we propose to develop techniques and tools to enable practical secure two-party computations and to evaluate these tools by building several scalable privacy-preserving applications.

We will develop methods for avoiding the memory bottleneck associated with garbled circuit evaluation by aggressively pipelining circuit generation and evaluation. We will explore a variety of techniques for reducing the size of garbled circuits including minimizing the bit width of individual component instances and isolating secure computation from parts of a computation that can be done independently.

Standard garbled circuits provide strong privacy properties assuming an honest-but-curious adversary who always follows the specified protocol. Realistic adversaries, however, need not follow the protocol, and known techniques for privacy against such adversaries are too expensive. We will develop new techniques for dealing with malicious adversaries, improving the standard cut-and-choose and commit-and-prove approaches by using new cryptographic tools developed by the PIs, and considering an alternate model in which a verifiable trusted party is used to generate the circuit but not trusted with any private data.

We will integrate our approach with programming tools for defining secure computations at a high level, based on information-flow analysis and program partitioning, to enable programmers to specify secure computations without being experts. We will also explore the problem of determining how much information the result itself leaks, and how to incorporate notions of privacy leakage directly into our programming tools and circuit implementations.

Intellectual Merit. The proposed work will advance knowledge in cryptography, system security, and privacy. It builds on recent advances in cryptographic protocols, secure computation, and information-flow analysis (including several developed by the PIs) to tackle the key performance, security, and programming impediments to widespread adoption of secure computation. Our preliminary results demonstrate that our approach can produce orders of magnitude performance improvements over previous results for several important applications, and the proposal outlines ideas for extending and generalizing our approach, as well as for integrating it with new cryptographic protocols and information-flow based programming language techniques. Our team includes experts in pure and applied cryptography, security, programming languages, and genomics, and coalesces five previously successful pairwise collaborations.

Broader Impacts. Efficient secure computation is essential for preserving privacy in many critical and emerging applications. Our work aims to make privacy-preserving computation practical and accessible enough to be used routinely in applications such as personalized genetics, medical research, and privacy-preserving biometrics. Our team is strongly committed to education and public outreach, and has produced several widely-used textbooks and led the creation of two new degree programs. Our proposed work includes public outreach and distributing and supporting open-source tools for secure computation.

Practical Secure Two-Party Computation: Techniques, Tools, and Applications

Project Description

The goal of *secure two-party computation* is to enable two parties to cooperatively evaluate a function that takes both parties' private data as input without revealing any private data to the other party. At the end of the computation, the participants learn the output of the function but no other information is revealed. Secure computation has many important applications such as privacy-preserving biometric identification, secret bid auctions, and personal genetics (see Section 5). Theoretical secure computation solutions have been known since the 1980s, but real systems are scarce because of the high costs associated with traditional techniques, the mismatch between theoretical adversary models and realistic threats, and the effort required to construct a secure computation. The goal of the proposed work is to make privacy-preserving computation practical enough that it can be used routinely in important, large scale applications. Our solutions build on Yao's garbled circuit technique [138], but address the three fundamental impediments to practical secure computation:

Performance (Section 1)**.** A major problem with previous garbled circuits implementations is that they require the entire circuit to be generated and stored in memory before evaluation begins. This limits the applications that can be handled by previous implementations to small problems. We propose to develop techniques for generating and evaluating garbled circuits in a pipelined fashion, and for subdividing circuits into components that can be generated and evaluated in parallel. We will also develop techniques to minimize the size of the circuit needed to solve a given problem by developing optimizations that eliminate non-free gates, and by identifying parts of the computation that can be done without needing cooperative secure computation.

Malicious Adversaries (Section 2)**.** Secure computation research often assumes an *honest-but-curious* adversary who follows the protocol as specified but also attempts to learn additional information. Such a threat model provides a useful theoretical basis for designing and reasoning about private computations, but does not provide much assurance against realistic adversaries who may violate the protocol. There are known techniques for producing secure computations that are effective against malicious adversaries.We propose several new ideas towards practical secure computation in the presence of malicious adversaries including a new approach to cut-and-choose that reduces overall communication and memory usage, a new approach for commit-and-prove based on verifiable encryption, and an alternate model which enables low-overhead protocols by introducing a verifiable third-party trusted to produce circuits but not with any private data.

Programming (Section 3)**.** In order for private computation to be used widely, it must be relatively easy and straightforward to produce efficient, private computations. Fairplay [84] demonstrated that it is possible to compile programs written in a high-level procedural language to garbled circuits. However, circuits generated by Fairplay and its successors are too inefficient for large applications. We propose to address the programming challenges for secure computation by adapting information flow techniques (including program partitioning) and by developing targeted optimizations for producing efficient circuits.

One important issue in secure computation applications is determining whether the *result* itself leaks too much private information. We propose to develop new techniques for auditing secure computations that can be integrated into garbled circuits based on information-flow and information theory (Section 4).

The techniques we propose are general, but their effectiveness depends on the application. We plan to build several applications to measure and understand the performance properties and to target our efforts to the key challenges of important applications. We will target applications that have clear privacy requirements including biometric identification, private encryption, and personal genetics (Section 5).

Our team brings together experts in pure cryptography, applied cryptography, system security, programming languages, and genomics to enable a broad and systemic approach to the problem. We have a strong commitment to education, and a track record for impact in mentoring graduate and undergraduate students, creating new degree programs, and publishing successful textbooks. We will distribute our tools as open-source software and provide an open platform for developing private computations (Section 6).

1 Research Plan: Efficient Garbled Circuits

Yao's garbled circuits provide a general mechanism for secure computation. This section provides background on garbled circuits and identifies reasons why they have not been used widely in practical applications (Section 1.1), and describes our planned research in improving the evaluation garbled circuits (Section 1.2) and generating more efficient circuits for applications (Section 1.3). Section 1.4 reports on our preliminary results, including more than 20x speedups over previous work on several applications.

1.1 Background

Several techniques have been developed for secure computation. The most notable are *homomorphic encryption* and *garbled circuits*. Reasonably efficient techniques exist for additively homomorphic encryption (systems that enable addition to be performed on encrypted values), such as Paillier's cryptosystem [104]. Additive homomorphic encryption, however, is not sufficient to perform general-purpose computation. Recent developments have shown that fully homomorphic (able to perform both addition and multiplication, and hence arbitrary computation) encryption systems are possible [41], but the best performance results so far are too slow for any realistic use. For example, one bootstrapping operation is required for every level of the circuit and requires between 30 seconds and 30 minutes (depending on the size of the security parameter) on a modern machine [42]. Hence, this proposal focuses on garbled circuits because of their more immediate practical potential and generality.

Garbled Circuits. Yao introduced the idea of using garbled circuits (also known as *Yao circuits*) to perform secure two-party computation [138]. Garbled circuits enable two honest-but-curious parties, P_0 and P_1, to compute an arbitrary function $f(x_0, x_1)$, where P_i has input x_i, without leaking any information about their respective inputs beyond what is revealed by the outcome itself. The idea is for one party (the *circuit generator*) to represent boolean wire signals for each wire with a nonce called its *wire label*, and to replace boolean signals found in each gate's truth table with the corresponding wire labels. The output labels in the truth table are encrypted using their corresponding input labels as keys and the encrypted truth tables are sent to the other party (the *circuit evaluator*). Oblivious transfer is used to obtain the initial input wire labels, after which the evaluator can evaluate the rest of the circuit without any further communication.

Figure 1 depicts an example. The input wires, k_0^0 and k_1^0 represent both possible input values for input 0. The AND gate is a four-entry table, where each entry is the value of the appropriate output wire label encrypted with the one of the possible input values for each input. The circuit evaluator can decrypt exactly one of these entries using the two input wire labels she has. The garbled table is permuted randomly, so the selected entry provides no information, other than the corresponding output wire label. Garbled gates can be chained to compute any computable function as illustrated in Figure 2. For each binary gate, the evaluator can decrypt one and only one entry in the garbled truth table, learning one of the output wire labels which is used to evaluate the next gate. In the end, the circuit generator sends to the evaluator a pair $\langle H(\lambda_i^0), H(\lambda_i^1)\rangle$, where $1 \le i \le n$ and H is a cryptographic hash function, for each of the n final output wires, so that the evaluator can map her final output wire labels back to boolean signals.

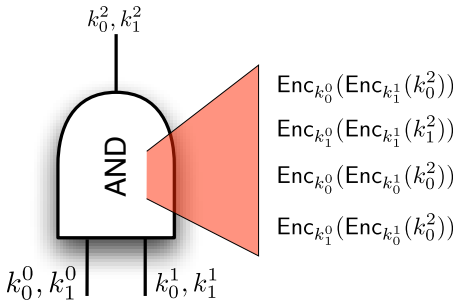

Figure 1: Yao's Garbled Circuits

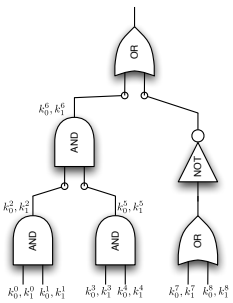

Figure 2: Encoding any function as a garbled circuit.

Oblivious Transfer. An oblivious transfer protocol involves a *sender* and a *receiver*. It allows a sender to send one of a possible set of values to a receiver; the receiver selects and learns only one of the values, and the sender cannot learn which value the receiver selected. A *1-out-of-2 oblivious transfer protocol* (denoted OT_1^2) allows the sender, who has two bits b_0 and b_1, to transfer b_σ to the receiver, where $\sigma \in \{0,1\}$ is a selection bit known only to the receiver [35]. Naor and Pinkas developed an efficient OT_1^2 protocol based on Decisional Diffie-Hellman (DDH) hardness assumption [92]. Based on the random oracle assumption, Ishai et al. devised a novel technique to reduce the cost of doing m OT_1^2 transfers to k OT_1^2, where k ($k \ll m$) serves as a configurable security parameter [66]. Our prototype uses these techniques efficiently transfer the input wire labels.

Secure Function Evaluation. Fairplay pioneered the effort to move secure function evaluation from a theoretical result to a technique that could be used in real systems [84]. Fairplay introduced a high-level language, *Secure Function Definition Language* (SFDL), for specifying a secure function, and a compiler to transform SFDL programs into low-level circuit implementations. Many subsequent projects either used Fairplay directly [10, 55, 76] or indirectly [71, 29, 120, 70, 110] to build privacy-preserving protocols. Several recent works have improved aspects of secure function evaluation. The free-XOR technique [76, 75] allows all XOR gates be executed by just XOR-ing the input wire labels, without needing any encryption operations. Security was initially proved using the random oracle model [9], but modified by Kolesnikov et al. to use the weaker correlation robustness assumption [76]. We exploit free XORs aggressively in the proposed work. Pinkas et al. [110] proposed the *garbled row reduction* technique, which we use in our prototype. It reduces the size of a garbled table to three entries (saving 25% of network bandwidth) for all non-free gates and is composable the with free-XOR technique.[1] TASTY [55] extended Fairplay's SFDL to allow the programmer to specify where in the digital circuit to integrate some arithmetic circuits (limited to addition and constant multiplication) that are realized by homomorphic encryption schemes. They also incorporated the free-XOR technique [76]. However, their approach still started from compiling enhanced SFDL programs, so the programmer does not have enough control over the circuit construction to minimize bit widths or make maximal use of free-XORs as is possible with our proposed approach.

1.2 Circuit Evaluation

A major reason for the poor performance of previous garbled circuit implementations is that the entire circuit must be generated and loaded in memory before evaluation starts. Since garbled circuits for solving problems can be huge, this poses a serious limit on the size of problem that can be handled. We propose to improve the performance of circuit evaluation by pipelining generation and evaluation to avoid this memory bottleneck, and by developing techniques for evaluating garbled circuits in parallel.

Pipelining. We can reduce the memory requirements, as well as latency, of garbled circuit protocols by performing circuit generation and evaluation in a pipelined fashion. In our proposed framework, the processing of the garbled truth tables is aggressively pipelined. At the beginning of the evaluation, both the generator and evaluator instantiate the circuit structure, which is known to both and fairly small since it can reuse components just like a non-garbled circuit. Note that the process of generating and evaluating the circuit does not (indeed, it cannot, because of privacy) depend on the inputs, so there is no overhead required to keep the two parties synchronized.

Our proposed framework automates the pipelined execution, so a user only needs to construct the desired circuit. When the protocol is executed, the generator transmits garbled truth tables over the network as they are produced, in the order defined by the circuit structure. As the client receives the garbled truth tables, it associates them with the corresponding gate. The client determines which gate to evaluate next based on the available output values and tables. Gate evaluation is triggered automatically when all the necessary

[1] Pinkas et al. [110] also presented an optimization based on secure secret sharing over finite fields that saves 50% of network bandwidth for all kinds of binary garbled circuits. Unfortunately, this optimization cannot be combined with the free-XOR technique. In most cases, free XORs provide more benefit than this table-size reduction, although there may be opportunities where it is better to use the table reduction technique.

inputs are ready, and the memory used by the table can be reclaimed after the gate is evaluated.

We have validated this approach by building a simple Java framework that supports pipelined circuit evaluation using a variation on the *Observer* design pattern [40]. As described in Section 1.4, our prototype already achieves order of magnitude improvements over previous circuit evaluation results for several applications. We will develop a more sophisticated framework to take advantage of pipelining opportunities while minimizing latency, modularize the circuit structure so even that need not be maintained in memory, and conduct experiments to evaluate different ways to control the generation and evaluation processes.

Parallelization. Circuits can be evaluated in any order that follows the topological dependencies, and subdivided into components that are evaluated separately in parallel. In addition, once the general structure of the circuit is determined, components can be generated in parallel. We will develop techniques to take advantage of multi-core processors to generate and evaluate circuits more efficiently. A larger opportunity is available using the processing power available on graphic processors (GPUs). Several projects have demonstrated substantial performance improvements by implementing AES encryption using GPUs [54, 85, 11]. Commodity GPUs today have hundreds of cores and CUDA makes these cores readily available for general-purpose processing [101], so the main challenge is finding a way to map the operations needed generate and evaluate large garbled circuits to GPU threads in a way that can effectively use many cores.

1.3 Circuit Generation

The resources needed to generate and evaluate a garbled circuit scale approximately linearly with the number of non-free binary gates in the circuit. To reduce the circuit size, we propose to explore several techniques for reducing the size of the circuit. After developing and evaluating the techniques manually, we will also automate the most general and effective optimizations (Section 3.2).

Much effort has been spent on designing logic circuits for hardware implementations. Some of the techniques developed for hardware circuits may be effective in garbled circuits also, but there are important differences between hardware circuits and garbled circuits that present new challenges and opportunities: (1) Hardware circuits are typically optimized to minimize power consumption and circuit area; with garbled circuits, our main goal is to minimize the number of non-free binary gates. (2) Garbled circuits are built and evaluated in software. This enables dynamic, data-specific optimizations that would not be cost-effective in hardware. (3) Privacy requires that a particular garbled table can only be evaluated once, whereas hardware circuit design focuses on being able to reuse physical circuits.

Component Library. We will develop a library of useful circuit components (comparator, adder, muxer, min, etc.) designed to make the best use of free XOR techniques. We will also explore automated design techniques to find optimal circuits for particular logical functions for a given cost metric.

Minimizing Bit Width. The dynamic use-specific nature of garbled circuits means we can generate circuits for each operation that are specific to the actual maximum input data size. For many examples (including all the examples in Section 1.4), a core structure is repeated many times in the circuit, but the maximum input sizes vary. We have found that designing our components to have variable bit width and using the minimum bit width necessary for each instance reduces the cost of typical applications by about 25%.

Symbolic Evaluation. In cases where some inputs to a garbled circuit are known, we can use symbolic evaluation to propagate these values through the circuit, leading to opportunities to use logical inference with symbolic values to reduce the number of non-free gates. Similarly, we can automatically propagate known wire signals when the circuit is built.

Isolating Secure Computation. Most computations involve a mix of operations that involve no private data, operations that involve private data from only one party, and operations that require private data from both parties. Only the third type requires expensive secure computation; the other two can be done locally by one of the parties without any need for evaluating garbled circuits. The challenge is that the parts of the computation that involve private data cannot be easily separated from the rest of the computation. In our preliminary experiments, we have found several opportunities for taking advantage of this to simplify and reduce the cost of the circuit such as computing the key schedule for AES locally and computing the

Application	Best Previous	Measurement	Results	Our Results	Speedup
Fingerprint Matching	Barni et al. [7]	Closest Threshold Match[a]	16s	3.5s	4.5[a]
Face Recognition	SCiFI [103]	900-bit Hamming, Online	0.310s	0.019s	16.3
		900-bit Hamming, Total	213s	0.05s	4176
Levenshtein Distance	Jha et al. [71]	100×100[b]	92.4s	4.1s	22.4
		200×200[c]	534s	18.4s	29.0
Smith-Waterman	Jha et al. [71]	60×60	*d*	447s	*d*
AES Encryption	Henecka et al. [55]	Online Time (per block)	0.4s	0.008s	50
		Total Time (per block)	3.3s	0.2s	16.5

Table 1: Performance Comparison for Several Secure Computations.
All results are for 80-bit wire labels and the security parameters for the extended OT protocol [66] are $(80, 80)$. Our results are the average of 100 trials with the client and server each running on a Intel Core Duo E8400 3GHz; the comparisons are with results reported in the cited papers, using similar, but not identical, machines. *a*. The speedup for the fingerprint matching is reported as 4.5, but our results are for an input size 1.8 times larger than used in [7] (512x640x8 vs. 320x640x7), which cannot find the best match but instead identifies all entries within some threshold. *b*. Protocol 1, a garbled-circuit only implementation that is faster than Protocol 3, but does not scale to 200×200. *c*. Protocol 3, a hybrid protocol (the flagship protocol of [71]). *d*. No meaningful comparison is possible here, although our protocol is about twice as fast, since [71] implemented a simplified Smith-Waterman protocol [63].

gap function in the Smith-Waterman application. (One emphasis of the work proposed in Section 3 is to automate the identification of such opportunities.)

Non-Binary Gates. The garbled circuit technique extends to gates with any number of inputs and outputs. Because of the *permute-and-encrypt* technique [84] gates with multiple inputs can be evaluated with a single encryption, so although the generation time scales with the number of inputs the evaluation time is nearly constant. We propose to develop techniques for using non-binary gates and automating the process of designing an efficient circuit that takes advantage of larger gates. One example where this has been useful is implementing constant lookup tables (present in many applications such as the score matrix for Smith-Waterman and the SBox for AES).

Using Wire Labels. The wire labels obtained during a garbled circuit evaluation are normally treated as a worthless by-product of the evaluation, but can be used in subsequent computations. They still contain no semantic information, but can be used in conjunction with permuted data structures to perform additional computation without leaking any information. For example, the wire exiting each comparison sub-circuit in a tree-structured circuit for determining the minimum value from a large set encodes the information about each pairwise comparison. Thus, the wire labels obtained by the evaluator can be used to evaluate a *backtracking tree* created by the generator to very efficiently obliviously retrieve profile information associated with the minimum value found. We used this technique in our fingerprint matching protocol [64] and propose to explore and develop other opportunities for exploiting obtained wire labels.

1.4 Applications and Preliminary Results

To validate our ideas we built a prototype framework for efficient circuit evaluation employing a straightforward implementation of the pipelining technique from Section 1.2 and using several of the circuit design ideas from Section 1.3 to develop target applications. Table 1 summarizes the results. The fingerprint matching application is described in more detail in a forthcoming paper [64]; the other applications are described in a paper currently under submission [63].

In *privacy-preserving fingerprint matching*, a client has a scanned candidate fingerprint and the server has a database of fingerprint images with associated profile information. The system does not reveal any information about the candidate fingerprint to the server, or about the database to the client, except the identity of the closest match if there is a match within some threshold distance (or the non-existence of any close match). We designed a bit-width minimizing circuit for finding the minimum difference of the Euclidean distances (removing the random vector added in the homomorphic encryption phase), and used a novel backtracking strategy to obliviously obtain the associated profile information. Our

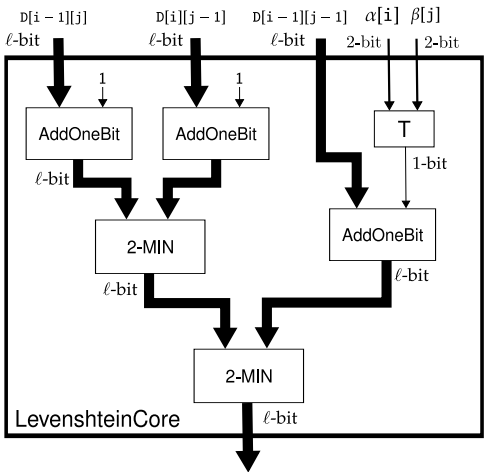

Figure 3: Straightforward Circuit.

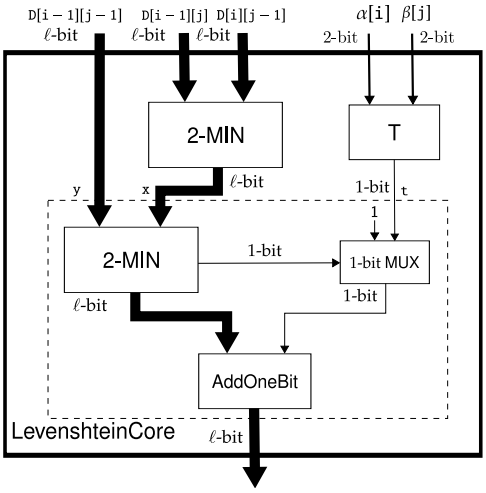

Figure 4: Optimized Circuit.

fingerprint matching protocol combines an additive homomorphic encryption phase used to compute Euclidean distances between fingerprint vectors, and a garbled circuit phase for finding the closest match within ϵ. For the other applications, we use only garbled circuit techniques.

The main operation in the *privacy-preserving face recognition* application is computing the Hamming distance between bit vectors representing face characteristics. Our circuit-only implementation is significantly faster that Osadchy et al.'s results which use a combination of homomorphic encryption and 1-out-of-n oblivious transfer. The *Levenshtein Distance* (edit distance) and *Smith-Waterman* (genome alignment) applications use a dynamic programming algorithm. In the privacy-preserving setting, each party has one of the input strings and they wish to compute the distance between the two strings without revealing anything else about the input strings. In considering these problems, Jha et al. concluded that garbled circuits could not scale to solve small (200-length strings for Levenshtein distance) problems because of the memory blowup as the circuit size increases [71]. Our pipelining approach avoids the memory problem, allowing use to solve arbitrarily large instances. For example, we used our prototype to evaluate a 2000×10,000 Levenshtein distance problem on commodity PCs, evaluating more than 1.29 billion non-free gates in 223 minutes. Figure 4 shows an example circuit from our implementation, resulting from applying the ideas from Section 1.2 starting from the straightforward circuit in Figure 3. The transformations preserve the logical functionality, but along with the bit width minimization, approximately halve the number of non-free gates required.

The *AES encryption* application performs encryption where one party knows the key and the other party knows the message. If it can be made reasonably efficient, two-party encryption has many potential applications including keyword searching and blind signatures [110]. Our implementation is distinguished from previous ones in that instead of constructing a huge circuit, we design our protocol around the structure of a traditional program. Our guiding principle is to identify the minimal subset of the computation that needs to be privacy-preserving, and only use expensive cooperative computation for those computations. We also take advantage of the circuit-minimization techniques identified above including partial evaluation, known-value propagation, and the fast table lookup primitive. We developed two versions of private AES, one focused on on-line performance (achieving a 50x speedup over the best previous result [55]) and the other focused on total execution time (achieving a 16x speedup).

Summary. Our preliminary results suggest that efficient garbled circuit implementations can outperform custom-designed hybrid solutions for a wide range of applications. Despite these performance improvements, however, the performance still needs to improve by at least another order of magnitude before problems of the scale typically needed for biometric matching or personal genetics applications can be handled. We are optimistic that such improvements can be achieved using the techniques proposed here

including parallelizing evaluation (which has not been used at all in our preliminary results) and taking further advantage of pipelining and opportunities for minimizing the number of non-free gates. Constructing each application still requires a fair bit of manual work (e.g., about 3 student-weeks for the AES encryption), so it is important to automate the construction of high-performing circuits as we propose in Section 3.

2 Research Plan: Malicious Adversaries

The previous section assumed an honest-but-curious adversary, but most application scenarios for secure computation need to also provide privacy against malicious adversaries who do not necessarily follow the protocol as specified. Two approaches with many variations on each have been used to achieve security against malicious adversaries in the traditional setting: *cut-and-choose* [80, 98] and *commit-and-prove* [45, 69, 81]. We consider both techniques in the next two subsections, and propose directions that can enhance their practicality with our approach. In Section 2.3, we propose a new model for implementing secure computation using a verifiable trusted party.

2.1 Cut-and-Choose

In the standard cut-and-choose method, the generator first prepares s copies of the garbled circuit and sends them to the evaluator. The two parties then randomly select an α-fraction of the circuits (where $\alpha \approx \frac{1}{2}$ or optimally, $\alpha \approx \frac{3}{5}^{\text{ths}}$ [17]) to be opened and verified by the evaluator. Standard implementations of these steps require the circuits to be stored and re-read several times. As noted in Section 1.2, one of our goals is to reduce the memory required for secure computation by using pipelined execution. The standard cut-and-choose method, however, requires multiple copies of the entire circuit. We propose to explore techniques to make this work efficiently with pipelined evaluation. One approach we will explore is for the generator to use a pseudo-random seed for each circuit, hash each circuit during its generation, and send the hash to the evaluator. Goyal et al. developed an approach along these lines, but did not consider pipelining issues or implement it in a practical system [49]. When the parties select which circuits to reveal, the generator sends the seed for each revealed circuit to the evaluator who reproduces each circuit and verifies the circuit and corresponding hash. After the evaluation of each circuit (which can be done in a pipelined fashion), the evaluator store the final output; after all circuits have been evaluated, the evaluator computes the majority output. We propose to investigate how to combine these ideas in a way that is efficient while provably preserving the needed security properties. In particular, we must ensure that these changes do not compromise the measures required to ensure that the generator's inputs to all αs copies of the circuits are consistent.

2.2 Commit-and-Prove

Aside from the original GMW paper [46], both Jarecki and Shmatikov [69] and Lindell and Pinkas [81] considered commit-and-prove techniques. We will investigate two new approaches to commit-and-prove, and develop a new technique that limits our reliance on the use of random-oracles.

Verifiable Encryption with Fast Zero-Knowledge Protocols. The challenge with using zero-knowledge to prove that a garbled circuit is a proper encoding stems from the complexity of the block ciphers (such as AES) typically used to garble gates. Proofs about such circuits are exceedingly large both because of the large number of gates involved (e.g., $\approx 30,000$ for AES) and the size inflation induced by general reductions from statements about such circuits to languages for which zero-knowledge protocols are known. Block-ciphers are efficient for the honest-but-curious model, but for the malicious model it makes sense to consider slower methods for garbling gates that permit more efficient proofs of correctness.

We propose to investigate a *verifiable encryption* scheme which constructs garbled gates using double-encryption or a related scheme that permits efficient zero-knowledge proofs of consistency of garbled gate encodings. Verifiable encryption is a type of encryption scheme which has the ability to prove (in a zero-knowledge fashion) specific claims about messages encrypted under it [14, 13]. These two prior works on verifiable encryption were based on the discrete log and composite residuosity hardness assumptions and consider how to verify any general property.

We propose to consider new ways to construct encryptions schemes—perhaps by considering new hardness assumptions such as pairing-based Diffie-Hellman assumptions—that encode messages in the exponents (in contrast to methods like El Gamal). The goal is to combine the exponents through the use of the pairing. The resulting ciphertext will contain information about both encoded messages in the exponent. This should make it possible to construct more efficient zero-knowledge proofs about the messages. In particular, there are known Σ-protocols (see below) for showing different relations between exponents for Diffie-Hellman and discrete log related ciphertexts. Alternately, such a scheme may also have efficient proofs using the recently developed Groth-Sahai Non-Interactive Zero-Knowledge (NIZK) proofs [52]. Such proofs are designed to be highly efficient, and can prove properties about the relations of exponents in many practical signature and encryption schemes.

Any such encryption scheme will result in the computation of a number of exponentiations as opposed to invocations of AES. Thus, it will be computationally more expensive to generate and evaluate gates. However, the goal is to reduce the total computation involved in a secure computation in the malicious setting. We will aim to develop a construction that will permit each garbled gate to be constructed in isolation to allow both pipelining and the parallel creation of garbled gates.

Σ-Protocols and Seed-Incompressible Functions. In any garbled circuit solution that uses zero-knowledge proofs, it is desirable for both efficiency and security to make the proofs non-interactive. This reduces communication latency, and provides security benefits such as making honest prover protocols secure in malicious models and providing for concurrent security. A traditional approach to removing interaction has been to apply the Fiat-Shamir transform to convert a Σ-protocol (a three-round, public-coin, and honest verifier zero-knowledge proof) into a NIZK proof [36]. However, this protocol is secure only in the random oracle model, and actual heuristic implementations of this model are known to be incorrect [15, 47, 8].

We propose to leverage prior research, conducted by PI Myers, to help transform Σ-protocols into NIZK arguments that can be used in garbled circuits. PI Myers has investigated methods to place random-oracle-based constructions on a firmer theoretical footing. The challenge is to find specific "random-oracle-like" properties, such that functions with these properties (a) can be realized in the standard model, and (b) can be securely used in some cryptographic applications in lieu of access to a truly random function. Halevi, Myers and Rackoff introduced the notion of *seed incompressibility* [53]. At a high level, this notion captures the intuition that a random function has no structure. It seems hopeless to define an efficiently computable function that has no structure, since the fact that the function is computed by a small circuit is itself some structure. However, one may still hope that this small circuit is the only interesting "small property" of the function. That is, no adversary can find a significantly smaller property that differentiates it from a random function. Unfortunately, seed incompressible functions have been proven to not exist [53]. A slightly weaker notion of *correlation-intractability* under seed-compression attacks, however, may be achievable. Further, it is possible (in the common random string model) to use such functions to efficiently transform Σ-protocols in to one-time NIZK arguments. PI shelat has developed a Σ-protocol compiler. We will modify it to implement NIZK arguments, as prescribed by the correlation-intractable seed-incompressible construction, and use it as a key component of a compiler for circuits that are secure against malicious adversaries.

Many Round Commit-and-Prove. Much cryptographic work focuses on *reducing* the number of rounds in a protocol even at the cost of increased communication. We propose a protocol that does the opposite: we increase the number of rounds with the aim of reducing the total size of all exchanged messages.

Briefly, the idea is to use a Σ-protocol to implement the "commit-and-prove" method from [45]. We can estimate the complexity of this approach as follows. The first two players will commit their inputs using a commitment scheme for which there exists efficient Σ-protocols (e.g., Pedersen commitment [108]). We organize the circuit into levels where each level consists of either only XOR gates or only AND gates. In evaluating the circuit, XOR gates can be handled locally, but the players must announce commitments of the resulting outputs and provide Σ-protocol proofs that these commitments are correct. The AND gates can be handled by oblivious transfer protocols recently proposed by Peikert et al. [109] in the common random string model that require only a small constant (8) number of exponentiations in the group Z_p. At the end of the evaluation, both parties hold commitments to the shares of the inputs to the next level. This

approach has round complexity that is related to the depth of the circuit. However, its communication complexity is highly competitive with the best two-party protocols. In comparison, the cut-and-choose method for compiling a Yao circuit requires roughly s copies of 4 symmetric encryptions to be exchanged where s is a statistical security parameter that is roughly 320 for 2^{-80} security. Our proposed approach also works well with the ideas from the previous section concerning the use of efficient proof systems such as the Groth-Sahai NIZK and the seed-incompressible functions. We propose to investigate this approach and design and implement prototypes to evaluate its performance.

2.3 A Verifiable Trusted Party Model

The role of a trusted party is to perform work on behalf of two mutually distrusting agents. If universally trusted parties existed there would be no need for secure multi-party computation—everyone could simply give their private data to the trusted party. In practice, however, it is unlikely that any party can be completely trusted with everyone's private data. The verifiable trusted party model addresses these shortcomings by requiring that the trusted party T does (1) not learn the inputs of the other parties, (2) does not learn the outputs of the other parties, and (3) is unable to force the parties to make errors in their computations. Moreover, the model makes it possible to verify that T performs all of the operations requested of it by the parties in the protocol. Izmalkov, Lepinski, and Micali [67] introduced the verifiable trusted party model in the context of a cryptographic game theory problem. Micali and PI shelat later used the model in another context related to game theory [89].

We propose to define a verifiable trusted party model for two-party computation, and to develop methods for using a verifiable trusted party T to generate garbled circuits for a given function, f. One approach has T constructing and signing the garbled circuit. The generator then asks T for the keys corresponding to the input wires of the circuit, and the evaluator asks for just the garbled circuit. From this point, the protocol proceeds as usual. The effect of signing the garbled circuit and description of f is to enforce culpability on T if the garbled circuit is faulty. We imagine that T would function like a certificate-authority, selling digitally signed garbled circuits that encode a given function. Should T ever sell an improperly encoded garbled circuit, it would lose credibility in the marketplace, as its garbled circuit could be posted along with its signature, showing that T was not trustworthy.

This proposal is simple and efficient, and we will implement it. However, the model raises several research questions: (1) Is it possible for T to stream the circuit to the evaluator following the pipelining approach from Section 1.2 such that the entire circuit does not need to be held in memory *and* so it is still possible to allow proofs that T has cheated. In other words, if T cheats, is the entire circuit necessary, or can single gates be offered as evidence of cheating. One avenue we will explore is using the notion of a *quotable signature*, recently proposed by PI shelat [3]. (2) The function f is currently revealed in the case of cheating by T. Is it possible to keep f private *even* during an audit showing that T has cheated? A generic zero-knowledge approach is feasible in theory; our goal is to find a practical solution that is as fast as a technique that proves an initial encoding of f correct. (3) Another natural question is whether one can (efficiently, and without the use of general function evaluation) design a T that makes a garbled circuit without learning the actual circuit f that the players wish to compute.

3 Research Plan: Programming Secure Computations

One of our goals is to enable widespread use of secure computation, which requires a high-level programming language for conveniently describing secure computations. Here, we describe our plans for using information flow to automatically partition programs into parts that need to be executed cooperatively, and for building a compiler that produces efficient garbled circuit protocols from high level programs.

3.1 Information Flow Analysis

Security-typed languages incorporate means to specify and check the security of computations directly in their type systems. This approach has been popularized by Jif [90, 18]. In Jif, types are annotated with *security labels* that indicate which principal *owns* the data and what principals may read or modify the data.

Principals also separately specify *acts-for* relationships, identifying principals to whom they delegate some authority. Programs may include explicit *declassification* coercions at points where data may flow to a less trusted party; these are essentially a kind of security-oriented downcast.

This approach allows programmers to focus on specifying appropriate security policies for the data being manipulated by their program, while relying on the compiler to enforce those policies. Using security-typed languages forces a programmer to explicitly annotate code to guide the compiler in establishing that the computation is secure. For sound languages, successful compilation guarantees the program adheres to the security policies indicated in the source code. Although there has been a great deal of work on information flow in programs (e.g., see the survey by Myers and Sabelfeld [119]), work in this direction has been largely separated from research in cryptographic protocol design. Two particularly promising synergies involve combining secure computation with program partitioning and declassification.

Partitioning. Secure program partitioning was introduced in Jif/Split [139, 140]. Given a Jif program annotated with security labels and a set of trust relationships, the Jif/Split compiler automatically partitions the program to run on one or more hosts so that all security requirements are satisfied. Program partitioning provides the same abstraction as secure computation: both aim to securely perform a computation involving multiple independent entities that do not fully trust each other. Subsequent work examined automatic partitioning of programs among web browser and web server [19, 24], including work by PI Hicks on enforcing security policies across the tiers of a multi-tier web application [131, 26]. These systems are limited, however, in that they ultimately require each host to trust *some* other host completely. For example, the oblivious transfer example in Jif/Split *requires* a trusted third party [139]. But these computations can be performed obliviously using secure computation. Hence, our proposed work combines the two ideas by extending Jif/Split with the idea of a *virtual host AB* that is trusted by principals *A* and *B*. When a splitting computation would require a trusted third party *C*, the compiler instead generates a secure computation protocol that can be executed cooperatively by the two principals.

Declassification. Information flow programming techniques prevent accidental information leaks, but typically need to explicitly declassify data at some points to perform useful computations. In particular, writing some desired functionality *without* any explicit declassifications and then compiling it will generate errors at exactly the points when a declassification is needed. These are the information release points where private data is needed for the computation to proceed. Recent work has explored ways to infer the least number of declassification points [73], but the need to declassify presents a fundamental risk of these techniques. With secure computation, we can replace the need to declassify with secure computation. At the end of the computation, an explicit declassification step may still be needed to reveal the result. We propose to integrate the declassification policy into the generated program (including the information auditing techniques we propose in Section 4).

3.2 Generating Efficient Garbled Circuits

The information flow analysis enables the compiler to determine parts of the computation that must be computed securely; our goal is to automatically generate efficient protocols to perform those computations. Fairplay already demonstrated the feasibility of automatically compiling a procedural program into a secure computation protocol [84], but as noted in Section 1.1, Fairplay and its successors are not able to take advantage of many important optimization opportunities. We propose to develop compilation techniques to take maximal advantage of the opportunities identified in Section 1.3. For some, like *symbolic evaluation*, adapting standard compiler techniques may be sufficient. For others, like *bit width minimization*, new techniques will be needed that go beyond standard optimizations. We acknowledge that it may not always be feasible to fully automate these optimizations. For these cases, we will design mechanisms for programmers to provide additional information to the compiler to enable the optimizations. We will also develop techniques to automatically select library components to minimize the size of the resulting circuit.

We will use the information flow analysis to separate the computation into those parts that can be performed locally by the parties and those that must be carried by the virtual host or hosts. This may involve automatically restructuring the computation to isolate the parts that must be computed cooperatively. For

example, if we have some function $f(x, y)$ that is large and complex, we may decompose it into $f'(g_1(x), g_2(y))$ where g_1 and g_2 can be computed locally and only f' needs to be cooperatively evaluated. Garbled circuits provide a direct way to compose secure and local computations by obliviously transferring the input labels corresponding to the intermediate results. As an example, consider the problem of computing the intersection of two sets of size n. The trivial $O(n^2)$-sized circuit for this computation can be improved using sorting networks with each party locally sorting their inputs before starting the secure computation.[2]

4 Research Plan: Auditing

Secure computation research typically assumes that the result can be revealed to one or both parties at the end of the computation. Any useful result, however, must reveal some information about the other party's private data. To enable secure computation for sensitive applications it is necessary that participants can have assurances limiting the amount of information leaking in the revealed results. These issues have been studied widely in the database community, but considering how to incorporate them into secure computation is largely unexplored.[3] We propose to explore two approaches for auditing leakage in secure computations. One uses an information flow analysis of the source program connected to a differential privacy model; the second takes an information-theoretic approach, estimating the amount of entropy remaining to a curious participant.

4.1 Information-Flow Auditing of Secure Protocols

There are several ways that security-typed languages can add assurance to secure computations. Declassification points in programs are obvious candidates for logging: these are the only points at which information is actually being released. Audits could then be conducted on logged data to examine potential breaches, assess the rate of information leakage [83, 88, 6, 77], or even assess what may constitute a remote party's current belief [20]. Recent work by Reed and Pierce [117] considered programming language foundations for *differential privacy* [27, 28] with functions written in a special-purpose language that bounds the amount of information an adversary can learn. In ongoing work, we have found that this language can be encoded in a dependently typed language with support for affine types [130] (extending PI Hicks' prior systems Fable and SELinks [131, 26]). As these languages are also able to include information flow tracking (as in Jif) we can potentially augment the entire system to perform a secure computation as described in the previous section while accounting more quantitatively for information release.

4.2 Measuring Leakage

In prior work, PI Evans developed an information-theoretic metric for measuring privacy loss in RFID systems based on probability distributions [99, 100]. The secure computing setting presents many new challenges and opportunities, however. One opportunity garbled circuits provide is measuring the information leakage as a computation is executed and designing the circuit to suppress outputs that would leak too much information, thus preventing either party from learning the result.

We will first consider a simpler scenario where a client is sending queries to a server, for example to find the Hamming distance between a query string and the server's private string, both of length N. The client and server execute a secure computation protocol that only reveals the result to the server, which then has a choice whether or not to reveal the output to the client. For a single query, the amount of information leaking depends only on the result (e.g., if the output is 0 the client would learn the exact value of the server's secret; if it is near $\frac{N}{2}$ much less information leaks) and can be easily computed. For a sequence of queries, however, this computation quickly becomes intractable, and the server must instead use techniques to bound the entropy remaining. Attackers may be able to design sequences of queries that reveal the private information as a group, while each individual query appears to leak minimal information (e.g., [96, 48]). For example, Figure 5 shows the number of possible 5-nucleotide sequences consistent with all responses

[2] Even in a malicious setting this can be a net win, since it is possible for each party to independently verify that the other party correctly sorted their inputs (without leaking any other information about those inputs) more efficiently than it is to sort the inputs.

[3] One notable exception is Wang et al. [136], which incorporated constraint-based query auditing into a specialized privacy-preserving computation, but did not consider auditing in general secure computation.

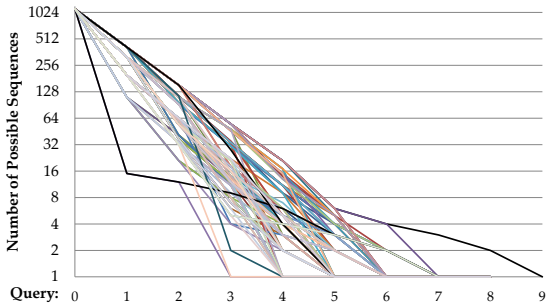

Figure 5: Uncertainty Reduction.
Each line shows the number of 5-nucleotide sequences that are consistent with the Levenshtein distance responses over a sequence of queries chosen using the smallest entropy strategy. Results for 100 experiments, each with a different randomly-select secret sequence.

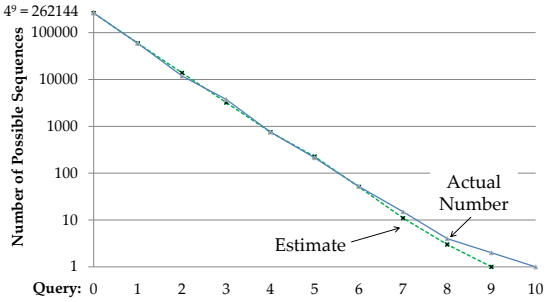

Figure 6: Estimating Information Leakage.
Accuracy of leakage estimate for Hamming distance for 9-nucleotide sequence. The actual number is computed exhaustively; the estimate line is computed using a closed equation.

over a series of queries. This type of exhaustive analysis cannot scale to large sequences so it is necessary to develop heuristics to estimate and formula to bound the information leaked. In our preliminary work, we have developed formulas for computing both upper bounds and tight estimates of the information leaking over a series of queries, and tested our estimates on small inputs by running exhaustive simulations (Figure 6). We propose to further develop techniques based on Monte Carlo sampling that take advantage of the structure of the problem to obtain an estimate of the entropy, as well as function-specific techniques for computing an upper bound.

5 Applications

We introduced several applications in Section 1.4 including privacy-preserving biometrics, sequence comparisons, and private encryption, and we intend to use those applications to evaluate our work. In addition, we will develop several practical applications focusing on important privacy-sensitive applications in genetics, which we describe in detail below. We will use the applications to evaluate the proposed techniques from the previous sections. We will also develop manually optimized versions of many of the applications, and compare them to versions produced automatically by our high-level programming tools.

Opportunity. The completion of the human genome sequence over ten years ago has been followed by an ever-expanding search for the inheritable differences (or genetic *variants*) between human genomes that effect disease risk. The exhilarating pace of biotechnology development has simultaneously provided the means to now resequence thousands of new human genomes (or at least significant, variant-containing portions) at great economies of scale, allowing scientists to further identify progressively rarer mutations that might affect predisposition to complex diseases, such as type 1 diabetes, heart disease, and psychological disorders [137]. Using panels of common variants at millions of sites across the human genome, scientists can screen even larger patient populations to discover genes responsible for complex human traits. This deluge of genetic information gathering is tightly balanced by ethical, legal (e.g., the Genetic Information Nondiscrimination Act of 2008 [134]) and social ramifications to ensure patient confidentiality while allowing patients to benefit from the research as they choose. Data from such studies are also tightly controlled, safeguarded by the NCBI in a centralized database (dbGaP [82, 37]), to which individual scientific access requires IRB approval and monitoring.

In parallel with these scientific advances has been a continuing increase in public awareness and interest in learning one's own genome, or at least the variants that relate to human health and well-being [23]. Direct-to-consumer genetic testing companies are responding to this interest, providing kits to simultaneously test for many thousands of variants related to disease-variant carrier status (e.g., the cystic fibrosis gene, BRCA cancer mutations), increased disease risk (e.g., type 2 diabetes, Parkinson's disease), abnormal drug responses (e.g., Warfarin toxicity, caffeine metabolism), and other heritable traits [78]. Other variants can

be used to report ancestral mixing (e.g., an individual might learn they are 60% European, 33% African, and 7% Asian by descent) [118]. One testing company provides a service to discover and (re)connect with unknown or lost relatives who have also been tested [1].

Privacy-Preserving Personalized Genetic Research. With such an abundance of personal genetic information becoming available, appropriate privacy assurances will provide an opportunity to fruitfully compare one's own genetic data with that of someone else's, either in bulk or individually. For bulk studies, individuals may wish to compare their own variants to those observed in all patients participating in a clinical drug trial to assess their own likelihood of responding to the drug. Similarly, individuals could assess their relative genetic proximity to the cases vs. controls in a disease association study. Without preexisting-knowledge of what the specific marker variants are for such phenotypes, the required computation amounts to a classification problem, clustering the individual with one or the other patient population based on raw correlation with variants in the study seen associated with disease status. For example, the individual could be clustered using a Hamming distance metric across 10,000 common variants, weighted by the relative ability of each variant to predict disease status (e.g., using mutual information content [5]). Such a computation appears to be feasible using our secure computation techniques.

People might also wish to contribute their own genetic information to further scientific understanding of a particular rare disease without directly enrolling in the study or allowing their genetic information be centrally stored in dbGaP. Here, all the scientific association study requires is a summation of the raw counts in a 2×2 contingency table, without regard to which individuals in the study harbored particular variants [16]. In either of these scenarios, performing association of the individual to the group, while maintaining privacy for both the individual and the groups in dbGaP, requires scalable secure computation.

Privacy-Preserving Screens for Genetic Risk. For those variant markers that have already been discovered and validated, someone may want to compare markers to those of a prospective partner, motivated by concerns regarding procreative compatibility as well as simple personal preferences. This comparison can be done with a simple Hamming distance computation across thousands of genetic markers of interest, which we have already demonstrated as a secure computation. One possible, although arguably dystopian, application is *genetic dating*, a compatibility screening to privately compare genetic information, identifying the presence of potential issues [74]. Information leakage can be controlled by limiting the result of the comparison to boolean answers, indicating the risk of any shared recessive genetic conditions.

6 Education, Outreach, and Transfer Plan

Our team has a strong commitment to education, outreach, and to making useful tools available to the research community, government, and industry.

Course Development. All the PIs are heavily involved in curriculum development. PI Evans is Founding Director of the Interdisciplinary Major in Computer Science [32] (the first cross-school program of its kind at UVa) and authored an introductory computer science textbook [31]. PI Katz is co-author of a textbook on modern cryptography that has been used at more than 40 schools [72].PI Myers was one of two architects of the Security Informatics MS Program at Indiana University and co-authored a textbook on phishing [68]. PI shelat is co-authoring a new cryptography textbook [105]. We anticipate opportunities to include the ideas in this proposal in future courses at both the introductory and graduate levels as well as in future textbooks. PIs Katz, Myers and shelat regularly teach undergraduate and graduate cryptography courses, and will find opportunities to incorporate ideas from the proposed work into these courses. PI Mackey co-developed a module on bioinformations for a new course on computation targeted to graduate students in other fields.

Graduate and Undergraduate Research. Graduate students made essential contributions to developing this proposal, and all of the PIs view mentoring graduate students as among the most important things we do. PI Hicks recently co-authored a paper in *Communications of the ACM* on a new approach to managing a research group [56]. All the PIs have strong records and serious commitments to producing PhD students who go on to successful careers in academia, government labs, and industry. We also have successful records

for developing undergraduate researchers in our research groups. Two former undergraduate researchers in PI Evans' group are now faculty members (Jon McCune, now at CMU and GJ Halfond, now at USC) and two are recent CRA Outstanding Undergraduate Research Award finalists (Salvatore Guarnieri, now a PhD student at the U. Washington, and Adrienne Felt, now a PhD student at UC Berkeley). Two of PI shelat's undergraduate research advisees have been recognized with CRA Outstanding Undergraduate Research Awards (Rachel Miller, finalist in 2009, now a graduate student at MIT; and David Noble, honorable mention in 2010). PI Myers is key member of IU's site-wide REU proposal for undergraduate research in Security Informatics with its INSPIRE-Information Security Program in Informatics Research Experience.

Public Outreach. Our team is deeply involved in, and committed to, outreach to children, other research communities, industry, and the general public. PI Evans and PI shelat contribute to the NSF-funded Tapestry workshop at Virginia [22] which aims to assist high school teachers in attracting and motivating a diverse group of students in their computing classes. PI Evans has developed a two-day course that introduces cryptography to 7th and 8th graders and taught the course to over 200 students at a nearby low-income middle school as well as to summer students in the GEAR-UP program [135] and 2nd graders. PI Myers provides advice to a local high school Science Olympiad coach on areas relating to computing or mathematics. Our experience with these courses indicates that cryptography can be used to make mathematics exciting and appealing to a wide range of students. PI Katz has given several tutorials and seminars exposing a wider research and industry computing to cryptographic research, including lectures on modern cryptography for networking researchers at SIGCOMM 2007 and mathematicians at the *AMS Joint Mathematics Meeting*.

There will always be some cost associated with secure computation, and it will only be widely used as a result of either regulatory or consumer pressures. Hence, we believe it is important to also make efforts to inform the general public. As an example, PI Evans' previous work on privacy for RFID systems and social networks has been covered in hundreds of press articles (e.g., [97, 133, 12, 21] and influenced influenced industry (e.g., NXP releasing a new version of the Mifare tag with improved security [102]) and policy (e.g., the Dutch government deciding not to deploy a insecure nationwide RFID system).

Tool Distribution. We will release the tools we develop as supported open-source software that includes accessible and documented interfaces to our private programming system at several levels, including the low-level Java framework, an annotated circuit-level description language, and a high-level programming language. In addition, we will make a library of circuits designed to provide efficient components for private computation available. The PIs have strong records for releasing and supporting open source tools. PI Evans' lightweight static analysis tool Splint is incorporated in most Linux distributions [30] and his MCL simulator is used by many other research groups [33]; PI Hicks has released a program analysis tool for detecting data races (Locksmith [113, 111]), a safe dialect of C (Cyclone [51, 50, 58], and tools for enhancing the security of web applications (SELinks [25, 131] and DRuby [38, 39]); PI Mackey is an original member of the Open Bioinformatics Foundation and has released the GLEAN and Evigan genome analysis tools and developed public databases for genomic Malaria research.

7 Results from Prior NSF Support

PI Evans has been the PI for 5 completed, and 1 ongoing NSF grant, focusing on system security, program analysis, and applications of cryptography. His CAREER award (*CAREER: Programming the Swarm* ($285K, 3/1/01-2/28/06, 0092945) and *ITR: A Framework for Environment-Aware Massively Distributed Computing* ($400K, 9/15/02-8/31/05, 0205327) explored programming and security issues involved in building dependable software systems. Works from these projects have been cited over 1800 times, including work on biological programming models [2, 43, 44], protocols for secure wireless networks [60, 62, 61], virtual machine security [106, 107], and lightweight static analysis [79, 34]. Software developed through these grants include Splint [30] (which is included in most Linux distributions, three commercial products, and received over 175,000 pageviews in the past year), MCL [33], Perracotta, CellSim, and the N-Variant Framework.

PI Hicks has been the PI or co-PI of 3 completed and 4 ongoing NSF grants. With NSF support he has published 40 papers in peer-reviewed conferences and journals (garnering more than 1000 total citations), released 10 substantial software packages (mostly analysis tools), and graduated six PhDs (with three more

expected next year). A closely related grant is *CAREER: Programming Languages for Reliable and Secure Low-level Systems* ($550K, 6/04–6/10, CNS-0346989) which explored security and reliability in low-level and concurrent software (often written in C) [58, 116, 132, 111, 112, 57, 125, 126, 94, 115, 87, 114], and means to achieve safe high-availability using dynamic software updating [127, 95, 128, 129, 93].

PI Katz received an NSF CAREER award in 2004 and is currently supported by two other NSF awards. Katz's CAREER award deals with security in large-scale decentralized systems like peer-to-peer networks. Katz's prior NSF support has resulted in more than 45 publications in addition to two published books, and has been used to support several graduate students (4 of whom have graduated already) and two post-docs.

PI Mackey has no prior NSF support. **PI Myers** and **PI Shelat** are new faculty members and have no completed NSF grants. They are co-PIs on a recent NSF award *Implications of Fully Homomorphic Encryption* ($500k, 09/10–08/13, 1018543) to study the cryptographic implications of fully homomorphic encryption. Shelat also received an NSF CAREER award in 2009 for the period 2009–2014, focusing on collusion-free protocols, optimistic methods for designing protocols, and building a compiler for Σ-protocols.

8 Impact Summary

The proposed work will develop new programming approaches and cryptographic techniques for secure computation and use those techniques to develop frameworks, compilers, and demonstration applications for secure computation. Selected research milestones for the project are summarized below.

Year 1

- Develop, evaluate, and release prototype framework for efficient garbled circuit evaluation.
- Design and prove security of minimal communication cut-and-choose protocol.
- Implement seed-incompressible modifications to Σ-protocol compiler.
- Develop and evaluate manual information-theoretic auditing techniques for simple applications.

Year 2

- Produce parallelized garbled circuit framework and optimize critical circuits for parallel execution.
- Develop verifiable encryption algorithm and multi-round commit-and-prove protocol.
- Integrate secure computation techniques into secure program partitioning tool.
- Produce and evaluate several full-scale applications, including a personalized genetics application.

Year 3

- Automate circuit optimization techniques including symbolic evaluation and bit width minimization.
- Analyze performance of cut-and-choose, commit-and-prove, and verifiable third party protocols.
- Incorporate secure computation into declassification points in secure program.
- Release high-level programming system for efficient honest-but-curious secure computations.

Year 4

- Automate advanced circuit optimization techniques including use of non-binary gates and restructuring to isolate secure computations.
- Implement the two most promising malicious protocols and integrate malicious adversary protocols into programming system.
- Develop general auditing mechanisms and incorporate them into declassification points.
- Evaluate versions of target applications implemented using the high-level programming tools.

Year 5

- Evaluate security and performance of different malicious-adversary circuit implementations.
- Incorporate general auditing techniques into circuits and programming tools, and evaluate their precision.
- Release high-level programming system for efficient malicious-adversary secure computations.

9 Collaboration Plan

Our team coalesces several longstanding collaborations towards a unified goal. PIs Myers and shelat previously collaborated to prove a long-standing open question, showing that single bit CCA2-secure encryption implies the existence of many-bit CCA2 security [91] and have joint NSF-funded project on fully homomorphic encryption. PIs Evans and Katz crossed paths as undergraduates at MIT and have been working together on the garbled circuits aspects of this proposal since last spring [64]. PIs Evans and shelat are colleagues at the University of Virginia and are co-advising students as PIs on an NSF-funded project on RFID privacy. PIs Katz and shelat have previously collaborated on collusion-free multi-party computation [4]. PIs Katz and Hicks are colleagues at the University of Maryland who have previously collaborated on secure sharing among distrusting parties [86] and expect to co-advise students working on this project. Thus, our team conjoins five prior pair-wise collaborations involving the PIs. We are excited about the opportunities the proposed work will provide to develop these collaborations as well as enable new collaborations.

Although modern telecommunications makes long distance collaboration feasible, we firmly believe that there is still no substitute for the depth and immediacy of face-to-face interactions. Our team is conveniently located to enable regular in-person meetings, as shown in Figure 7. In particular, Fredericksburg, VA is within a 90 minute drive of both the University of Virginia and the University of Maryland, enabling easy part-day meetings at a coffee shop there which some of the PIs and our students have taken advantage of in developing the work that forms the basis of parts of this proposal.

In addition to the fairly spontaneous sub-team meetings enabled by our geographic proximity, we will have longer full team meetings approximately every 12 months to ensure we maximize opportunities for collaboration across the entire team. We will start with a kick-off meeting in Charlottesville, and then rotate among the PI sites for subsequent meetings. In addition, we also believe there is a lot of value in student circulation and cross-pollination. We anticipate having our students having both short and extended visits at each other's sites to work closely with the PIs as well as other students, as well as some PIs being external dissertation committee members for students of other PIs at different institutions.

Table 2 summarizes the roles of each of the PIs. As PI, Evans will coordinate the overall team effort, ensuring that each sub-project maximally leverages the ideas and tools being developed by others. Each of the major components of the proposed work each involve both theoretical advances and experimental systems development. We believe much of the most exciting work in security has resulted from close

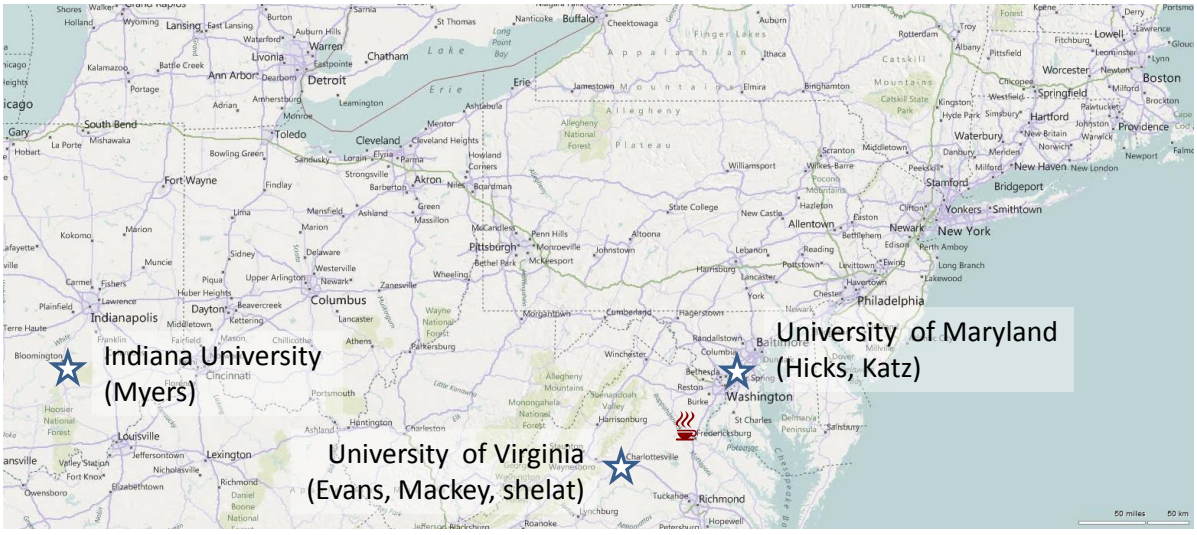

Figure 7: Locations of the PIs.

Area	Primary PIs
Efficient Garbled Circuits (Section 1)	Evans, Katz
Malicious Adversaries (Section 2)	Myers, shelat, Katz, Evans
High-Level Programming (Section 3)	Hicks, Evans, Katz
Auditing (Section 4)	Evans, Katz, Hicks
Applications (Section 5)	Mackey, Evans, Katz, Myers, shelat

Table 2: Division of Research Areas.

collaborations between theorists and experimentalists, and have designed our team around this belief. At both the University of Virginia and University of Maryland, our team includes PIs who are more theory-focused (shelat at UVa, Katz at UMd) and systems-focused (Evans at UVa, Hicks at UMd), although all PIs have interests in both theoretical and practical aspects of the proposed work, as well as experience working from both perspectives. At Indiana University, PI Myers is more theory-focused but also worked on a number of practical systems and problems (e.g., phishing [68], ubiquitous security and malware [124, 59, 65, 121], and measurement of implemented cryptographic systems [122, 123]). This arrangement enables close collaboration on both theory and systems at each site, as well as ensuring a substantial amount of cross-site collaboration.

We will target real applications of secure computation. Our team includes PI Mackey, from the Public Health Sciences department at UVa, who works on genomics. The other PIs, especially Evans and shelat at UVa, but others on visits, will work closely with Mackey (and one of his students, partly funded by this proposal) to identify opportunities for secure computation in personal genomics and statistical medical research.

In addition to collaboration across our team, we all actively collaborate with other researchers and are deeply involved in our respective research communities. PI Evans is active in the security research community; he was Program Co-Chair for the past two *IEEE Symposia on Security and Privacy* (Oakland 2009 and Oakland 2010) and the lead organizer of an NSF co-sponsored workshop on the *Science of Security*. He is also involved in the programming languages and software engineering research communities (e.g., PC member and panels chair for ICSE and external review committee for PLDI). PI Hicks is primarily active in the programming language community; he will be Program Chair for the *39th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages* (POPL 2012) and was General and Program Chair for the 2007 *ACM SIGPLAN Workshop on Programming Languages and Analysis for Security*. PI Katz is active in the cryptography research community; he is on the editorial boards for the *International Journal of Applied Cryptography*, *Fundamenta Informaticae*, and *IEE Proceedings — Information Security*, was Program Chair for *Applied Cryptography and Network Security* 2007, and is a program committee member for many theory, cryptography, and security conferences. PI shelat is active in the theory and cryptography research communities. He was on the program committees for CRYPTO 2010, Oakland 2009, and TCC 2008. PI Myers is active in the theory, cryptography, and security communities. He was on the program committees for ACM CCS 2008 and 2010, Financial Crypto 2007 and 2009 and Eurocrypt 2007. PI Mackey is active in the genomics research community and an original participating member of the Open Bioinformatics Foundation. The involvement of our team in multiple academic communities will ensure maximum exposure of our work to a broad range of researchers, as well as enabling us to leverage work across the cryptography, security, programming languages, theory, and genomics research communities.

References Cited

[1] 23andMe, Inc. 23andMe: Genetic Testing for Health, Disease, and Ancestry. `https://www.23andme.com/`.

[2] Tarek Abdelzaher, Brian Blum, Qiuhua Cao, Y. Chen, David Evans, J. George, Selvin George, Lin Gu, Tian He, Sudha Krishnamurthy, Liqian Luo, Sang Son, John Stankovic, Radu Stoleru, and Anthony Wood. EnviroTrack: Towards an Environmental Computing Paradigm for Distributed Sensor Networks. In *24th International Conference on Distributed Computing Systems*, Mar 2004.

[3] Jae Hyun Ahn, Dan Boneh, Jan Camenisch, Susan Hohenberger, abhi shelat, and Brent Waters. Computing on Authenticated Data. Manuscript, 2010.

[4] Joël Alwen, Jonathan Katz, Yehuda Lindell, Giuseppe Persiano, abhi shelat, and Ivan Visconti. Collusion-Free Multiparty Computation in the Mediated Model. In *Advances in Cryptology — Crypto 2009*, volume 5677 of *LNCS*, pages 524–540. Springer, 2009.

[5] Dimitris Anastassiou. Computational Analysis of the Synergy among Multiple Interacting Genes. *Molecular Systems Biology*, 3:83, Jan 2007.

[6] Michael Backes, Boris Köpf, and Andrey Rybalchenko. Automatic Discovery and Quantification of Information Leaks. In *30th IEEE Symposium on Security and Privacy (Oakland)*, pages 141–153. IEEE, 2009.

[7] Mauro Barni, Tiziano Bianchi, Dario Catalano, Mario Di Raimondo, Ruggero Donida Labati, Pierluigi Faillia, D. Fiore, R. Lazzeretti, V. Piuri, F. Scotti, and A. Piva. Privacy-Preserving Fingercode Authentication. In *12th ACM Multimedia and Security Workshop*, 2010.

[8] M. Bellare, A. Boldyreva, and A. Palacio. An Uninstantiable Random-Oracle-Model Scheme for a Hybrid-Encryption Problem. In C. Canchin and J. Camenisch, editors, *Advances in Cryptology-EUROCRYPT'04*. Springer, 2004.

[9] Mihir Bellare and Phillip Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *ACM Conference on Computer and Communications Security*, 1993.

[10] Assaf Ben-David, Noam Nisan, and Benny Pinkas. FairplayMP: A System for Secure Multi-Party Computation. In *ACM Conference on Computer and Communications Security*, 2008.

[11] Joppe W. Bos, Dag Arne Osvik, and Deian Stefan. Fast Implementations of AES on Various Platforms. In *Software Performance Enhancement for Encryption and Decryption and Cryptographic Compilers*, 2009.

[12] Boston Globe. T Card Has Security Flaw, 6 March 2008.

[13] Jan Camenisch and Ivan Damgård. Verifiable Encryption, Group Encryption, and Their Applications to Separable Group Signatures and Signature Sharing Schemes. In Tatsuaki Okamoto, editor, *ASIACRYPT*, volume 1976 of *Lecture Notes in Computer Science*, pages 331–345. Springer, 2000.

[14] Jan Camenisch and Victor Shoup. Practical Verifiable Encryption and Decryption of Discrete Logarithms. In Dan Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 126–144. Springer, 2003.

[15] Ran Canetti, Oded Goldreich, and Shai Halevi. The Random Oracle Methodology, Revisited. *J. ACM*, 51(4):557–594, 2004.

[16] Rita M. Cantor, Kenneth Lange, and Janet S. Sinsheimer. Prioritizing GWAS Results: A Review of Statistical Methods and Recommendations for their Application. *American Journal of Human Genetics*, 86(1):6–22, Jan 2010.

[17] Chih-hao Shen and Abhi Shelat. Two-Output Secure Computation With Malicious Adversaries. Manuscript, 2010.

[18] Stephen Chong, Andrew Myers, K. Vikram, Nate Nystrom, Lantian Zheng, and Steve Zdancewic. Jif 3.0: Java Information Flow. `http://www.cs.cornell.edu/jif`.

[19] Stephen Chong, K. Vikram, and Andrew C. Myers. SIF: Enforcing Confidentiality and Integrity in Web Applications. In *USENIX Security Symposium*, 2007.

[20] Michael R. Clarkson, Andrew C. Myers, and Fred B. Schneider. Quantifying Information Flow with Beliefs. *Journal of Computer Security*, 17(5):655–701, 2009.

[21] CNet.com. Exclusive: The Next Facebook Privacy Scandal, 23 January 2008.

[22] Jim Cohoon and Joanne Cohoon. Tapestry Workshop on Attraction and Engagement of Students to Computing. `http://www.cs.virginia.edu/tapestry/`.

[23] Celeste M Condit. Public Attitudes and Beliefs about Genetics. *Annual Review of Genomics and Human Genetics*, 11:339–59, Sep 2010.

[24] Ezra Cooper, Sam Lindley, Philip Wadler, and Jeremy Yallop. Links: Web Programming Without Tiers. In *5th International Symposium on Formal Methods for Components and Objects*, September 2006.

[25] Brian J. Corcoran, Michael Hicks, Nikhil Swamy, and Simon Tsang. SELinks: End-to-end Security for Web Applications. `http://www.cs.umd.edu/projects/PL/selinks/`.

[26] Brian J. Corcoran, Nikhil Swamy, and Michael Hicks. Cross-Tier, Label-Based Security Enforcement for Web Applications. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 269–282, June 2009.

[27] Cynthia Dwork. Differential Privacy. In *33rd International Colloquium on Automata, Languages and Programming (ICALP)*, 2006.

[28] Cynthia Dwork. Differential Privacy: A Survey of Results. In *Theory and Applications of Models of Computation (TAMC)*, 2008.

[29] Zekeriya Erkin, Martin Franz, Jorge Guajardo, Stefan Katzenbeisser, Inald Lagendijk, and Tomas Toft. Privacy-preserving face recognition. In *9th International Symposium on Privacy Enhancing Technologies*, 2009.

[30] David Evans. Splint: Annotation-Assisted Lightweight Static Checking. `http://www.splint.org`, 1994–2011.

[31] David Evans. Introduction to Computing: Explorations in Language, Logic, and Machines. `http://www.computingbook.org/`, 2010.

[32] David Evans and Joanne Cohoon. Creating a Computer Science Major for Arts and Sciences Students. *CRA Computing Research News*, January 2008.

[33] David Evans and Lingxuan Hu. MCL Simulator. `http://www.cs.virginia.edu/mcl/`, 2004–2011.

[34] David Evans and David Larochelle. Improving Security Using Extensible Lightweight Static Analysis. *IEEE Software*, Jan 2002.

[35] Shimon Even, Oded Goldreich, and Abraham Lempel. A Randomized Protocol for Signing Contracts. *Communications of the ACM*, 1985.

[36] Amos Fiat and Adi Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *Advances in Cryptology—CRYPTO '86*, pages 186–194. Springer-Verlag, 1987.

[37] National Center for Biotechnology Information. Database of Genotypes and Phenotypes (dbGaP). `http://www.ncbi.nlm.nih.gov/gap`.

[38] Michael Furr, Jong hoon (David) An, Mark Daly, Benjamin Kirzhner, Avik Chaudhuri, Jeffrey Foster, and Michael Hicks. Diamondback Ruby. `http://www.cs.umd.edu/projects/PL/druby/`.

[39] Michael Furr, Jong hoon (David) An, Jeffrey S. Foster, and Michael Hicks. Static Type Inference for Ruby. In *24th Annual ACM Symposium on Applied Computing*, March 2009.

[40] Erich Gamma, Richard Helm, Ralph E. Johnson, and John Vlissides. *Design Patterns — Elements of Reusable Object-Oriented Software*. Addison-Wesley, March 1995.

[41] Craig Gentry. Fully Homomorphic Encryption using Ideal Lattices. In *41st ACM Symposium on Theory of Computing*, 2009.

[42] Craig Gentry and Shai Halevi. Implementing Gentry's Fully-Homomorphic Encryption Scheme. Technical report, IBM Research, October 8, 2010.

[43] Selvin George, David Evans, and Lance Davidson. A Biologically Inspired Programming Model for Self-Healing Systems. In *Workshop on Self-Healing Systems*, Nov 2002.

[44] Selvin George, David Evans, and Steven Marchette. A Biological Programming Model for Self-Healing. In *First ACM Workshop on Survivable and Self-Regenerative Systems*, Oct 2003.

[45] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority. In *ACM Symposium on the Theory of Computing*, pages 218–229. ACM, 1987.

[46] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to Solve any Multi-Party Protocol Problem. In *19th Symposium on Theory of Computing (STOD)*, 1987.

[47] Goldwasser and Taumann-Kilai. On the (In)security of the Fiat-Shamir Paradigm. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 102–115, 2003.

[48] Michael T. Goodrich. The Mastermind Attack on Genomic Data. In *30th IEEE Symposium on Security and Privacy (Oakland)*, 2009.

[49] Dan Grossman, Michael Hicks, Trevor Jim, and Greg Morrisett. Cyclone: A Type-safe Dialect of C. *C/C++ Users Journal*, January 2005.

[50] Dan Grossman, Mike Hicks, Trevor Jim, Greg Morrisett, Nikhil Swamy, and Yanling Wang. Cyclone: A Safe Dialect of C. `http://cyclone.thelanguage.org`.

[51] Jens Groth and Amit Sahai. Efficient Non-Interactive Proof Systems for Bilinear Groups. In Nigel P. Smart, editor, *EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432. Springer, 2008.

[52] Shai Halevi, Steven Myers, and Charles Rackoff. On Seed-Incompressible Functions. In *TCC*, pages 19–36, 2008.

[53] Owen Harrison and John Waldron. Practical Symmetric Key Cryptography on Modern Graphics Hardware. In *USENIX Security Symposium*, 2008.

[54] Wilko Henecka, Stefan Kögl, Ahmad-Reza Sadeghi, Thomas Schneider, and Immo Wehrenberg. TASTY: Tool for Automating Secure Two-partY computations. In *ACM Conference on Computer and Communications Security*, 2010.

[55] Michael Hicks and Jeffrey S. Foster. Score: Agile Research Group Management. *Communications of the ACM*, October 2010.

[56] Michael Hicks, Jeffrey S. Foster, and Polyvios Pratikakis. Inferring Locking for Atomic Sections. In *ACM SIGPLAN Workshop on Languages, Compilers, and Hardware Support for Transactional Computing (TRANSACT)*, June 2006.

[57] Michael Hicks, Greg Morrisett, Dan Grossman, and Trevor Jim. Experience with Safe Manual Memory Management in Cyclone. In *ACM International Symposium on Memory Management (ISMM)*, 2004.

[58] Hao Hu, Steven Myers, Vittoria Colizza, and Alessandro Vespignani. WiFi Networks and Malware Epidemiology. *Proceedings of the National Academy of Sciences*, January 2009.

[59] Lingxuan Hu and David Evans. Secure Aggregation for Wireless Networks. In *Workshop on Security and Assurance in Ad Hoc Networks*, Jan 2003.

[60] Lingxuan Hu and David Evans. Localization for Mobile Sensor Networks. In *10th Annual International Conference on Mobile Computing and Networking (MobiCom)*, Sep 2004.

[61] Lingxuan Hu and David Evans. Using Directional Antennas to Prevent Wormhole Attacks. In *Network and Distributed System Security Symposium (NDSS)*, Feb 2004.

[62] Yan Huang, David Evans, Jonathan Katz, and Lior Malka. Efficient Privacy-Preserving Computing Using Garbled Circuits. Currently under anonymous submission, preliminary version available at `http://www.cs.virginia.edu/evans/pubs/eppcgc.pdf`, 2011.

[63] Yan Huang, Lior Malka, David Evans, and Jonathan Katz. Efficient Privacy-Preserving Biometric Identification. In *Network and Distributed System Security Symposium*, 2011.

[64] Nathaniel Husted and Steven Myers. Mobile Location Tracking in Metro Areas: Malnets and Others. In *17th ACM Conference on Computer and Communications Security*, CCS '10, pages 85–96, New York, NY, USA, 2010. ACM.

[65] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending Oblivious Transfers Efficiently. In *CRYPTO*, 2003.

[66] Sergei Izmalkov, Matt Lepinski, and Silvio Micali. Verifiably Secure Devices. In *Theory of Cryptography Conference*, 2008.

[67] Markus Jakobsson and Steven Myers. *Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft*. Wiley-Interscience, 2006.

[68] S. Jarecki and V. Shmatikov. Efficient Two-Party Secure Computation on Committed Inputs. In *Eurocrypt*, pages 97–114, 2007.

[69] Ayman Jarrous and Benny Pinkas. Secure Hamming Distance Based Computation and Its Applications. In *International Conference on Applied Cryptography and Network Security*, 2009.

[70] Somesh Jha, Louis Kruger, and Vitaly Shmatikov. Towards Practical Privacy for Genomic Computation. In *IEEE Symposium on Security and Privacy*, 2008.

[71] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. Chapman and Hall/CRC Press, 2007.

[72] Dave King, Susmit Jha, Divya Muthukumaran, Trent Jaeger, Somesh Jha, and Sanjit A. Seshia. Automating Security Mediation Placement. In *European Symposium on Programming*, 2010.

[73] Robert L. Klitzman and Meghan M. Sweeney. "In Sickness and in Health"? Disclosures of Genetic Risks in Dating. *Journal of Genetic Counseling*, Oct 2010.

[74] Vladimir Kolesnikov, Ahmad-Reza Sadeghi, and Thomas Schneider. Improved Garbled Circuit Building Blocks and Applications to Auctions and Computing Minima. In *International Conference on Cryptology and Network Security (CANS)*, 2009.

[75] Vladimir Kolesnikov and Thomas Schneider. Improved Garbled Circuit: Free XOR Gates and Applications. In *International Colloquium on Automata, Languages and Programming*, 2008.

[76] Boris Köpf and Andrey Rybalchenko. Approximation and Randomization for Quantitative Information-Flow Analysis. In *23nd IEEE Computer Security Foundations Symposium (CSF)*. IEEE, 2010.

[77] Chee Seng Ku, En Yun Loy, Yudi Pawitan, and Kee Seng Chia. The pursuit of genome-wide association studies: where are we now? *J Hum Genet*, 55(4):195–206, Apr 2010.

[78] David Larochelle and David Evans. Statically Detecting Likely Buffer Overflow Vulnerabilities. In *10th USENIX Security Symposium*, 2001.

[79] Y. Lindell and B. Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In *Eurocrypt*, pages 52–78, 2007.

[80] Yehuda Lindell and Benny Pinkas. Secure Two-Party Computation Via Cut-and-Choose Oblivious Transfer. Cryptology ePrint Archive, 2010. `http://eprint.iacr.org/2010/284`.

[81] Matthew D Mailman, Michael Feolo, Yumi Jin, Masato Kimura, Kimberly Tryka, Rinat Bagoutdinov, Luning Hao, Anne Kiang, Justin Paschall, Lon Phan, Natalia Popova, Stephanie Pretel, Lora Ziyabari, Moira Lee, Yu Shao, Zhen Y Wang, Karl Sirotkin, Minghong Ward, Michael Kholodov, Kerry Zbicz, Jeffrey Beck, Michael Kimelman, Sergey Shevelev, Don Preuss, Eugene Yaschenko, Alan Graeff, James Ostell, and Stephen T Sherry. The NCBI dbGaP Database of Genotypes and Phenotypes. *Nature Genetics*, 39(10):1181–6, Oct 2007.

[82] Pasquale Malacaria. Assessing Security Threats of Looping Constructs. In *International Symposium on Principles of Programming Languages (POPL)*, pages 225–235, 2007.

[83] Dahlia Malkhi, Noam Nisan, Benny Pinkas, and Yaron Sella. Fairplay—A Secure Two-Party Computation System. In *USENIX Security Symposium*, 2004.

[84] Svetlin A. Manavski. CUDA Compatible GPU as an Efficient Hardware Accelerator for AES Cryptography. In *IEEE International Conference on Signal Processing and Communications*, 2007.

[85] Piotr Mardziel, Adam Bender, Michael Hicks, Dave Levin, Mudhakar Srivatsa, and Jonathan Katz. Secure Sharing in Distributed Information Management Applications: Problems and Directions. In *Annual Conference of the International Technology Alliance*, September 2010.

[86] Jean-Philippe Martin, Michael Hicks, Manuel Costa, Periklis Akritidis, and Miguel Castro. Dynamically Checking Ownership Policies in Concurrent C/C++ Programs. In *Proceedings of the ACM Conference on Principles of Programming Languages (POPL)*, January 2010. Full version.

[87] Stephen McCamant and Michael D. Ernst. Quantitative Information Flow as Network Flow Capacity. In *ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 193–205, Tucson, AZ, USA, June 9–11, 2008.

[88] Silvio Micali and Abhi Shelat. Purely Rational Secret Sharing (Extended Abstract). In Omer Reingold, editor, *TCC*, volume 5444 of *Lecture Notes in Computer Science*, pages 54–71. Springer, 2009.

[89] Andrew Myers. JFlow: Practical Mostly-Static Information Flow Control. In *26th ACM Symposium on Principles of Programming Languages*, 1999.

[90] Steven Myers and abhi shelat. Bit Encryption is Complete for CCA2 Security. In *Annual IEEE Symposium on Foundations of Computer Science*, October 2009.

[91] Moni Naor and Benny Pinkas. Efficient Oblivious Transfer Protocols. In *ACM-SIAM Symposium on Discrete Algorithms*, 2001.

[92] Iulian Neamtiu and Michael Hicks. Safe and Timely Dynamic Updates for Multi-threaded Programs. In *Proceedings of the ACM Conference on Programming Language Design and Implementation (PLDI)*, pages 13–24, June 2009.

[93] Iulian Neamtiu, Michael Hicks, Jeffrey S. Foster, and Polyvios Pratikakis. Contextual effects for version-consistent dynamic software updating and safe concurrent programming. In *Proceedings of the ACM Conference on Principles of Programming Languages (POPL)*, pages 37–50, January 2008.

[94] Iulian Neamtiu, Michael Hicks, Gareth Stoyle, and Manuel Oriol. Practical Dynamic Software Updating for C. In *Proceedings of the ACM Conference on Programming Language Design and Implementation (PLDI)*, pages 72–83, June 2006.

[95] Erich Neuwirth. Some Strategies for Mastermind. *Zeitschrift für Operations Research*, 1982.

[96] New York Times. When Everyone's a Friend, Is Anything Private?, 7 March 2009.

[97] Jesper Buus Nielsen and Claudio Orlandi. LEGO for Two-Party Secure Computation. In *6th Conference on Theory of Cryptography*, pages 368–386, Berlin, Heidelberg, 2009. Springer-Verlag.

[98] Karsten Nohl and David Evans. Quantifying Information Leakage in Tree-Based Hash Protocols. In *Eighth International Conference on Information and Communications Security (ICICS)*, 2006.

[99] Karsten Nohl and David Evans. Hiding in Groups: On the Expressiveness of Privacy Distributions. In *23rd International Information Security Conference*, 2008.

[100] Nvidia Corporation. *NVIDIA CUDA: Compute Unified Device Architecture: Reference Manual*. Nvidia, 2008.

[101] Mary O'Connor. NXP Announces New, More Secure Chip for Transport, Access Cards. *RFID Journal*, Mar 2008.

[102] Margarita Osadchy, Benny Pinkas, Ayman Jarrous, and Boaz Moskovich. SCiFI: A System for Secure Face Identification. In *IEEE Symposium on Security and Privacy*, 2010.

[103] Pascal Paillier. Public-key Cryptosystems Based on Composite Degree Residuosity Classes. In *Eurocrypt*, 1999.

[104] Rafael Pass and abhi shelat. *A Course in Cryptography*. 2007–2010.

[105] Nathanael Paul and David Evans. .NET Security: Lessons Learned and Missed from Java. In *Twentieth Annual Computer Security Applications Conference*, Dec 2004.

[106] Nathanael Paul and David Evans. Comparing Java and .NET security: Lessons Learned and Missed. *Computers & Security*, 25(5), Jul 2006.

[107] Torben P. Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In Joan Feigenbaum, editor, *CRYPTO*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer, 1991.

[108] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A Framework for Efficient and Composable Oblivious Transfer. In David Wagner, editor, *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 554–571. Springer, 2008.

[109] Benny Pinkas, Thomas Schneider, Nigel P. Smart, and Stephen C. Williams. Secure Two-Party Computation Is Practical. In *15th Annual International Conference on the Theory and Application of Cryptology and Information Security (Asiacrypt)*, 2009.

[110] Polyvios Pratikakis, Jeffrey S. Foster, and Michael Hicks. Context-Sensitive Correlation Analysis for Detecting Races. In *ACM Conference on Programming Language Design and Implementation (PLDI)*, 2006.

[111] Polyvios Pratikakis, Jeffrey S. Foster, and Michael Hicks. Existential Label Flow Inference via CFL Reachability. In Kwangkeun Yi, editor, *Proceedings of the Static Analysis Symposium (SAS)*, volume 4134 of *Lecture Notes in Computer Science*, pages 88–106. Springer-Verlag, August 2006.

[112] Polyvios Pratikakis, Jeffrey S. Foster, and Michael Hicks. Locksmith. `http://www.cs.umd.edu/projects/PL/locksmith`, 2007.

[113] Polyvios Pratikakis, Jeffrey S. Foster, and Michael Hicks. Locksmith: Practical Static Race Detection for C. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, March 2010. Accepted for publication.

[114] Polyvios Pratikakis, Jeffrey S. Foster, Michael Hicks, and Iulian Neamtiu. Formalizing Soundness of Contextual Effects. In Otmane Aït Mohamed, César Muñoz, and Sofiène Tahar, editors, *Proceedings of the International Conference on Theorem Proving in Higher Order Logics (TPHOLs)*, volume 5170 of *Lecture Notes in Computer Science*, pages 262–277. Springer, August 2008.

[115] Polyvios Pratikakis, Jaime Spacco, and Michael Hicks. Transparent Proxies for Java Futures. In *Proceedings of the ACM Conference on Object-Oriented Programming Languages, Systems, and Appilcations (OOPSLA)*, pages 206–223, October 2004.

[116] Jason Reed and Benjamin Pierce. Distance Makes the Types Grow Stronger: A Calculus for Differential Privacy. In *International Conference on Functional Programming*, 2010.

[117] Charmaine D. Royal, John Novembre, Stephanie M. Fullerton, David B. Goldstein, Jeffrey C. Long, Michael J. Bamshad, and Andrew G. Clark. Inferring Genetic Ancestry: Opportunities, Challenges, and Implications. *American Journal of Human Genetics*, 86(5):661–73, May 2010.

[118] Andrei Sabelfeld and Andrew C. Myers. Language-Based Information-Flow Security. *IEEE Journal on Selected Areas in Communications*, 21(1):5–19, January 2003.

[119] Ahmad-Reza Sadeghi, Thomas Schneider, and Immo Wehrenberg. Efficient Privacy-Preserving Face Recognition. In *International Conference on Information Security and Cryptology*, 2009.

[120] Youngsang Shin, Minaxi Gupta, and Steven Myers. Prevalence and Mitigation of Forum Spamming. In *INFOCOM*, 2011 (To appear).

[121] Youngsang Shin, Minaxi Gupta, and Steven A. Myers. A Study of the Performance of SSL on PDAs. Rio de Janeiro, Brasil, 04/2009 2009.

[122] Youngsang Shin, Steven Myers, and Minaxi Gupta. Saving Energy on WiFi With Required IPsec. In *ICST Conference on Security and Privacy in Communication Networks (SecureComm)*, volume 50, September 2010.

[123] Youngsang Shin, Steven A. Myers, and Minaxi Gupta. A Case Study on Asprox Infection Dynamics. In Ulrich Flegel and Danilo Bruschi, editors, *DIMVA*, volume 5587 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2009.

[124] Saurabh Srivastava, Michael Hicks, and Jeffrey S. Foster. Modular Information Hiding and Type Safety for C. In *Proceedings of the ACM Workshop on Types in Language Design and Implementation (TLDI)*, pages 3–14, January 2007.

[125] Saurabh Srivastava, Michael Hicks, Jeffrey S. Foster, and Patrick Jenkins. Modular Information Hiding and Type Safe Linking for C. *IEEE Transactions on Software Engineering*, 34(3):1–20, May 2008. Full version of TLDI 07 paper.

[126] Gareth Stoyle, Michael Hicks, Gavin Bierman, Peter Sewell, and Iulian Neamtiu. *Mutatis Mutandis*: Safe and Flexible Dynamic Software Updating. In *Proceedings of the ACM Conference on Principles of Programming Languages (POPL)*, pages 183–194, January 2005.

[127] Gareth Stoyle, Michael Hicks, Gavin Bierman, Peter Sewell, and Iulian Neamtiu. *Mutatis Mutandis*: Safe and Flexible Dynamic Software Updating (full version). *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 29(4), August 2007. Full version of POPL 05 paper.

[128] Suriya Subramanian, Michael Hicks, and Kathryn S. McKinley. Dynamic Software Updates: A VM-Centric Approach. In *Proceedings of the ACM Conference on Programming Language Design and Implementation (PLDI)*, pages 1–12, June 2009.

[129] Nikhil Swamy, Juan Chen, and Ravi Chugh. Enforcing Stateful Authorization and Information Flow Policies in Fine. In *European Symposium on Programming (ESOP)*, March 2010.

[130] Nikhil Swamy, Brian J. Corcoran, and Michael Hicks. Fable: A Language for Enforcing User-defined Security Policies. In *IEEE Symposium on Security and Privacy (Oakland)*, 2008.

[131] Nikhil Swamy, Michael Hicks, Greg Morrisett, Dan Grossman, and Trevor Jim. Safe Manual Memory Management in Cyclone. *Science of Computer Programming (SCP)*, 62(2):122–144, October 2006. Special issue on memory management. Expands ISMM conference paper of the same name.

[132] Technology Review. RFID's Security Problem, January/February 2009.

[133] United States Congress. Genetic Information Nondiscrimination Act of 2008. Public Law No: 110-233.

[134] UVa Today. Math Academy Prepares Middle-Schoolers for College, July 2007.

[135] Rui Wang, XiaoFeng Wang, Zhou Li, Haixu Tang, Michael K. Reiter, and Zheng Dong. Privacy-Preserving Genomic Computation through Program Specialization. In *16th ACM Conference on Computer and Communications Security*, 2009.

[136] Wellcome Trust Case Control Consortium. Genome-Wide Association Study of 14,000 Cases of Seven Common Diseases and 3,000 Shared Controls. *Nature*, 447(7145):661–78, Jun 2007.

[137] Andrew C. Yao. How to Generate and Exchange Secrets. In *Symposium on Foundations of Computer Science*, 1986.

[138] Steve Zdancewic, Lantian Zheng, Nathaniel Nystrom, and Andrew C. Myers. Secure Program Partitioning. *ACM Transactions on Computing Systems (TOCS)*, 20(3):283–328, 2002.

[139] Lantian Zheng, Stephen Chong, Andrew C. Myers, and Steve Zdancewic. Using Replication and Partitioning to Build Secure Distributed Systems. In *IEEE Symposium on Security and Privacy*, 2003.