♦ Advances and Impact of Secure Function Evaluation

Vladimir Kolesnikov

Secure function evaluation (SFE) is an area of cryptography concerned with the design of algorithms that allow mutually distrustful partners to evaluate desired functions on their inputs, while maximally preserving input privacy. With faster computers and networks, and continuous algorithm improvements, SFE has become within reach of practicality. The first large-scale industrial application of SFE—the result of over 20 years of relevant research—has been successfully completed in Denmark, where SFE was used to run a country-wide secure sugar beets auction. In this paper, we will provide a brief overview of SFE, the problem it solves, and its uses. We will describe several recent research advancements and argue that SFE is "the next big thing." © 2009 Alcatel-Lucent.

Introduction

Cryptography (from the Greek for "secret writing") has emerged as a tool for secret communication. Primitive encipherment approaches (for example, the Caesar's code) have existed for thousands of years. However, only recently, with the development of fast computing devices, has cryptography grown into a structured and mathematical science. The science of secret communications became more formal and rigorous, and, simultaneously, new directions of cryptography appeared and developed. Modern cryptography encompasses much more than the original intent. Examples of new directions include the ability to prove knowledge of secrets and a means for electronic identification, secure financial transactions, and much more.

The state of modern communications allows easy access to almost any imaginable resource or person. At the same time, the underlying connectivity layer provides weak guarantees, if any. For example, if Alice sends a message to Bob, the message not only may be lost, it may also be read and, more importantly, modified by an adversary while in transit. While most Internet traffic is of little or no interest to attackers, a portion of it serves transactions of value and requires strong security. Protection against eavesdropping and interference with legitimate communication is relatively well understood and remains perhaps the most commonly used fruit of cryptography.

However, even a perfectly secure communication system is only part of the solution. Imagine a situation where Alice participates in a transaction with Bob but does not completely trust him. This occurs in many settings where the participants may have conflicting interests, including contract signing or buy/sell transactions. Securing the communication channel cannot provide any assurance that Bob does not cheat. Can we protect Alice's (and everyone else's) interests in this setting? The study of secure function evaluation

(SFE), which began in the 1980s, emerged from the need not only to communicate, but also to *compute* securely. It addresses the problem of providing computational security in the presence of potentially untrustworthy partners.

In this exposition, we will concentrate on methods for preventing cheating and ensuring privacy in multi-party transactions, such as selling, voting, polling, and auctioning. We will provide examples, clarify the goals of secure computing, and present some recent and relevant research and performance improvements. We conclude with a discussion of research directions and argue the value of pursuing the study and development of efficient SFE techniques.

Existing and Envisioned Applications

Consider the following setting. There are n mutually distrustful parties, $P_1, \ldots, P_n$, with respective inputs $x_1, \ldots, x_n$, who had agreed on a function F, and wish to "securely compute" $F(x_1, \ldots, x_n)$, This means that participants learn $F(x_1, \ldots, x_n)$, but no additional information about $x_1, \ldots, x_n$ is revealed to anyone. (As with encryption, more efficient varieties of SFE can be broken with massive hundred-years-long computation. If desired, security can be made stronger, or even information-theoretic, i.e., provably unbreakable). Today, we know how to securely evaluate any such function F, albeit with a significant multiplicative overhead beyond the usual cost of its computation. Some functions can be evaluated more efficiently than others, and a significant portion of SFE research focuses on efficiency improvements of current solutions.

In this section, we focus on the question *why*, rather than *how*, to implement SFE. We show that SFE is a powerful primitive, which can be fruitfully used in a variety of scenarios. Many natural applications of SFE are in transactions where privacy is essential. Below we briefly discuss several such applications. We believe this demonstrates the role of SFE as an *enabler* of new paradigms and applications.

Auctions

Guaranteeing the privacy of bids is often critical in successful auction execution. First of all, revealing the bid is often detrimental to the bidder. For example, knowledge of the item's worth to a particular person allows for easy manipulation of the auction, making the winning bid higher that it otherwise might be. Without privacy guarantees, bidders would try to craft their votes, leading to a sub-optimal auction.

Panel 1. Abbreviations, Acronyms, and Terms

AES—Advanced Encryption Standard
CPU—Central processing unit
GC—Garbled circuit
PC—Personal computer
SFE—Secure function evaluation
SIM—Subscriber identity module

Further, in the economic field of mechanism design, the concept of a trusted third party has been a central assumption since the 1970s (see, e.g., [3]). However, in many cases, and especially online, it is difficult or expensive to obtain the service of a perfectly honest auctioneer. Indeed, it is all too easy to abuse the entrusted information and never be caught at it. SFE allows execution of the auction while providing strong guarantees of privacy and security, without any lingering doubts.

An SFE-based auction would run much as a normal auction. In one implementation, parties encrypt their bids and send them to the auctioneer, who, without being able to decrypt the bids, blindly sends encrypted confirmations back to the bidders. The winner receives and decrypts a signed winning contract, and all others receive and decrypt an empty string. Note that the auctioneer can be prevented from knowing the outcome of the auction until the winner presents the contract for fulfillment. Furthermore, if desired, auction participants can execute the auction without an auctioneer.

We discuss a recent real-world large-scale secure auction later in the paper.

Buy/Sell Transactions

An interesting special case of auctions is the purchase transactions that take place if there is a middle ground between the lowest seller's and highest buyer's prices. If no acceptable price exists, parties are notified, but no information is revealed on the parties' inputs.

Distributed Database Mining

Consider the case where organizations have private databases but wish to compute on a *joint* database.

This need may arise for credit card companies who wish to mine user information for fraud, or for government agencies who wish to compare suspect lists, but business interests or regulations prevent them from sharing the data. Using SFE, players are able to achieve their goals, while satisfying the business requirements and/or applicable regulations.

Data Storage Outsourcing

While storage hardware is cheap, running and maintaining a reliable and efficient enterprise storage solution may not be. Often it makes business sense to outsource the storage to an external provider. While sensitive data can be protected by storing it in an encrypted form, this simple solution is inefficient when selective access to data is desired. A recent direction in cryptography, searchable encryption, addresses this scenario by allowing the data owner to search on outsourced data and retrieve specific records, without having to download the entire encrypted database.

Distributed Trusted Party

As discussed above, it is hard to find a perfectly honest trusted intermediary to assist with the computation. At the same time, for computations with many participants, such as voting or polling, it is difficult to arrange for all the players to be online at the same time to execute an SFE protocol. In this case, SFE can be used to create a virtual trusted party. Namely, the job of the trusted authority is performed by SFE executed by a small number of independent authorities who are trusted not to collude. Each party splits its input into shares and sends each share to a corresponding authority. Upon collection of the inputs of all players, the authorities securely compute the desired function and announce the output. Because an incomplete set of input shares does not reveal the encoded input, privacy is guaranteed unless *all* of the designated authorities collude to cheat. The likelihood of such collusion can be made negligible, if the authorities are chosen carefully. Collusion is further deterred by the much higher possibility of being caught while attempting to obtain or misuse private data, compared to the likelihood of discovery in a single-trusted-authority solution. The application discussed in the section immediately following is an example of this approach.

A Commercial Large-Scale Secure Auction Execution

A recent exciting development in the area of SFE is its first large-scale commercial execution of a sugar beet auction, which was carried out by Danish farmers and Danisco, the Danish sugar processor, with the assistance of cryptography researchers. The full report is available [2], and here we briefly present the business case and the solution.

In Denmark, several thousand farmers produce sugar beets, which are sold to Danisco, according to a pricing and volume structure specified by contract. These contracts can be traded between farmers, but trading previously had been very limited and was primarily carried out through bilateral negotiations. In recent years, among other factors, the European Union (EU) reduced support for sugar beet production, which made it imperative that contracts to farmers be reallocated to optimize payoff. It was realized that the best approach was a nationwide double auction. At the auction, for each potential price, the sellers specify the amount of beets they would sell at that price, and buyers specify the amount they would buy. The auctioneer then determines the *market clearing price* (MCP), the price at which total supply equals total demand, and transactions are performed at that price.

Bid privacy was important to farmers, and they would be reluctant to accept Danisco acting as auctioneer. Even if Danisco would never misuse their private information, the mere fear of that happening would affect farmers' bids, and thus the optimality of the auction. At the same time, Danisco needed to have control over how its contracts were traded. Hiring a consultancy house to conduct the auction in confidence was considered, but rejected as a very expensive solution. Finally, it was decided to employ a distributed trusted party, as described above. The authorities (Danisco, the farmers' union, and the research center) collected shares of bids and computed the auction using SFE.

The auction had a total of 1,200 participating bidders. The actual computation took place on January 14, 2008, and lasted about 30 minutes. The result involved around 25,000 tons of production rights changing ownership. Secure computation of the auction achieved significant cost savings, and SFE is likely to be employed again in the future.

SFE Research: Performance Improvements

We now switch gears and briefly discuss the question of *how* to evaluate functions securely. We concentrate on the two-party setting. Here, there are two players, P_1 and P_2, who both know the evaluated function F and whose inputs are x_1 and x_2, respectively. We first describe a known approach, and then discuss recent research that improves on it.

Garbled Circuit Protocol

One of the most efficient SFE protocols is Yao's Garbled Circuit (GC). There, the computed function F is represented as a Boolean circuit C, which is blindly evaluates gate-by-gate. At a high level, this is achieved by P_1 encrypting, or *garbling*, C and giving the result to P_2. P_2, in turn, evaluates the garbled circuit (under encryption) and obtains and sends the encrypted output to P_1. Finally, P_1 decrypts the output and sends it back to P_2.

More specifically, the protocol proceeds as follows: Player P_1 first garbles circuit C. For each wire W_i, he randomly chooses two secrets, $w_i^0$and w_i^1 where w_i^j is a *garbling* of the W_i's value j. (Note: w_i^j does not reveal j). Further, for each gate G_i, P_1 creates and sends to P_2 a *garbled table* T_i, with the following property: given a set of garblings of G_i's inputs, T_i allows recovery of the garbling of the corresponding G_i's output, and nothing else. The tables are usually implemented as a set of encryptions, where each pair of input garblings serves as a key to the encryption of the corresponding output garbling.

The above is best illustrated by an example. Consider a gate G_0 with incoming wires W_1, W_2, outgoing wire W_0, and wire secrets w_i^j, as shown in **Figure 1**. As noted above, a gate table T_0, associated with G_0 should allow computation of a secret on the output wire corresponding to two given input secrets. For example, if G_0 is an AND gate, w_0^0 should be computed, e.g., from w_1^0 and w_2^0. Similarly, w_0^0 should be computed also given w_1^0 and w_2^1, and given w_1^1 and w_2^0; w_0^1 should be computed given w_1^1 and w_2^1. This is achieved by using the gate table T_0 consisting of four entries, each an encryption of either w_0^0 or w_0^1 under two keys. For an AND gate, T_0 may look as follows (in the real table, the rows are also randomly permuted):

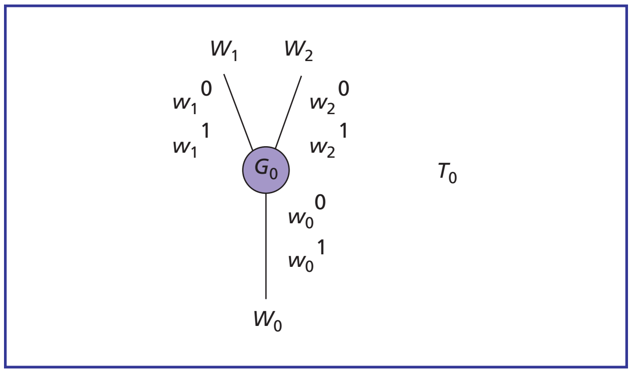


Figure 1.
Garbled gate and associated secrets.

$$T_0 = \begin{cases} t_{00} = Enc_{w_1^0}(Enc_{w_2^0}(w_0^0)) \\ t_{01} = Enc_{w_1^0}(Enc_{w_2^1}(w_0^0)) \\ t_{10} = Enc_{w_1^1}(Enc_{w_2^0}(w_0^0)) \\ t_{11} = Enc_{w_1^1}(Enc_{w_2^1}(w_0^1)) \end{cases}$$

Now, gate G_0 can be evaluated under encryption (given two input wire secrets) simply by trying to decrypt each of the four entries of T_0 using the secrets as keys. It is easy to see that only one of the decryptions will succeed, and that it will yield the encryption of the output wire secret.

In the next step, the garblings of players' inputs are securely transferred to P_2, using cryptographic techniques. If also given the gate tables T_i for the entire circuit, P_2 can obtain the garbled output simply by evaluating the garbled circuit gate-by-gate as described above. Notice that for each wire, P_2 can obtain only one garbling, the one that corresponds to the value on that wire. The garblings of the output wires of the circuit are published, and P_2 learns only the output of the circuit. He does not learn any of the internal wire values, which remain garbled. P_1 learns the output from P_2. The correctness of GC follows from the method of construction of tables T_i. Thus, neither party learns any additional information from the protocol execution.

We now briefly describe the complexity of GC. In practice, encryptions may be implemented with the (very fast) Advanced Encryption Standard (AES) function. (AES is faster than memory access, and its throughput is in the tens of gigabits per second.) The evaluation of the garbled circuit is linear in the size of the circuit and requires on average four AES calls per gate. Circuit construction requires up to eight AES calls per gate. Communication consists primarily of sending tables T_i (with four AES encryptions per each gate). Finally, transferring input keys requires a small number (two or three) of public-key operations per input transferred.

Eliminating XOR Gate Costs in GC

In many practical scenarios, the main cost of GC evaluation is the generation, transmission, and use of the garbled tables T_i. One such table, consisting of four encryptions, is required for each gate of the circuit. For large circuits, this may present a significant overhead.

Our recent work [5] eliminates the tables for XOR gates, without increasing costs for other gates. That is, in this construction, tables are not generated, transmitted, or used for XOR gates, which become "free" to evaluate. This drives various improvements, depending on the number of XOR gates in the evaluated circuits. We note that in many useful circuits, such as the addition, multiplication, or universal circuits, the number of XOR gates can be large, which results in SFE improvement up to a factor of 4 due to this technique. The factor of 4 improvement is achieved, in particular, for universal circuits.

The intuition behind the construction is as follows. We combine GC with the simple information-theoretic SFE implementation of XOR gates (e.g., construction 4 of [4]). The XOR gates of [4] do not use tables and are free of the associated costs. However, the construction of [4] imposes a restrictive global relationship on the wire secrets, which prevents its use in previous GC schemes. This restriction is overcome in [5], and we describe it here at a high level.

First, recall a gate-table-free implementation of the XOR gate G, derived from one of [4]. Let G have two input wires W_a,$\mathrm{W_b}$ and output wire W_c. Garble the wire values as follows. Randomly choose w_a^0, w_b^0, R. Set $w_c^0 = w_a^0 \oplus w_b^0$, and $\forall i \in \{a, b, c\}: w_i^1 = w_1^0 \oplus R$. It is easy to see that the gate table is not necessary in this case, and the garbled gate output is simply obtained by XORing garbled gate inputs:

$$w_c^0 = w_a^0 \oplus w_b^0 = (w_a^0 \oplus R) \oplus (w_b^0 \oplus R) = w_a^1 \oplus w_b^1$$
$$w_c^1 = w_c^0 \oplus R = w_a^0 \oplus (w_b^0 \oplus R)$$
$$= w_a^0 \oplus w_b^1 = (w_b^0 \oplus R) \oplus w_b^0 = w_a^1 \oplus w_b^0.$$

Further, as before, garblings w_i^j, do not reveal the wire values they correspond to.

We can now pinpoint the restriction that the above XOR construction imposes on the garbled values—the garblings of the two values of each wire in the circuit must differ by the same value, i.e., $\forall i : w_i^1 = w_1^0 \oplus R$, for some global R. In contrast, in previous GC constructions, *all* garblings w_i^j, were chosen independently at random, and proofs of security relied on that property.

The main observation is that it is not necessary to select all garblings independently. In [5], random R is chosen once, and garble wire values are set so that $\forall i : w_i^1 = w_i^0 \oplus R$. The security of the resulting construction is formally proven. We direct the interested reader to [5] for further details.

As mentioned above, this optimization drives savings up to a factor of 4 for both computation and communication in SFE using the GC method. The overhead of SFE is borderline affordable in many cases, and constant factor improvements such as the above are important in helping SFE gain acceptance for use in real-life applications.

Conclusions and Research Directions

In the previous sections on applications, and research and performance improvement, we attempted to provide a flavor of SFE use cases and constructions. While this account merely scratches the surface of the research area, we hope to have introduced the reader to the possibilities allowed by the technology.

We believe that the question of wide deployment of products and services based on or simply using SFE techniques is not *if* but w*hen*. With faster computers and networks, and continuous algorithm improvements, the already affordable overhead of SFE is quickly becoming insignificant. At the same time, the strong guarantees provided by SFE improve security of old applications and enable new ones.

Indeed, in most situations that arise in practice, the critical functions that require secure processing can be naturally isolated and are small in size. For example, auctions could be implemented by a browser plug-in. Most of the source code would be "ordinary," executing locally and collecting input, or displaying information. A small secure subroutine whose only task is to compute the auction function would be called once during the execution. This subroutine performs the simplest, but most critical computation, and can be represented by a small Boolean circuit. As discussed in the section on performance, this subroutine can be implemented efficiently, with minimal impact on the total solution.

While some of the cryptographic research focuses on demanding settings where the adversary is extremely powerful, solutions are much simpler and more efficient in the settings where additional reasonable assumptions are made. We advocate such settings as the earliest adopters of SFE technology. For example, use of even the simplest and most computationally weak tamper-proof devices helps significantly. Today, many personal computers (PCs) are equipped with trusted platform modules (and cellular phones with subscriber identity module (SIM) cards), which are small central processing units (CPUs) with internal functions and state that are inaccessible to the user. It is possible to take advantage of the physically protected execution on the opponent's computer and achieve further performance improvement by an order of magnitude. Another simplifying assumption is that many players (e.g., banks) would not attempt to cheat if there were significant risk of being caught. Protocols which merely check randomly for misbehavior, as opposed to those that prevent it, are much more efficient, while achieving similar goals (e.g., [1]). In our opinion, the search for such "shortcuts" and research on their proper use is the area of cryptography/security that holds most short- to mid-term promise. Studying generic SFE techniques, which would bring improvement to a variety of settings simultaneously, is an important mid-term research objective.

To summarize, we reiterate the thesis of this exposition. SFE is an active area of research. There are very efficient and simple SFE techniques, which are suitable for implementation and already in commercial use, even today. The potential for the development of new products and services based on current and expected SFE research is beginning to be recognized by the industry, and is indeed great.

References

[1] Y. Aumann and Y. Lindell, "Security Against Covert Adversaries: Efficient Protocols for Realistic Adversaries," Proc. 4th Theory of Cryptography Conf. (TCC '07) (Amsterdam, Neth., 2007), published in Theory of Cryptography, Lecture Notes in Comput. Sci. (LNCS 4392) (S. P. Vadhan, ed.), Springer, Berlin, Heidelberg, New York, 2007, pp. 137–156.

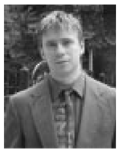
[2] P. Bogetoft, D. L. Christensen, I. Damgård, M. Geisler, T. Jakobsen, M. Krøigaard, J. D. Nielsen, J. B. Nielsen, K. Nielsen, J. Pagter, M. Schwartzbach, and T. Toft, "Secure Multiparty Computation Goes Live," Cryptology ePrint Archive, Report 2008/068, 2008, <http://eprint.iacr.org/2008/068.pdf>.

[3] A. Gibbard, "Manipulation of Voting Schemes: A General Result," Econometrica, 41:4 (1973), 587–601.

[4] V. Kolesnikov, "Gate Evaluation Secret Sharing and Secure One-Round Two-Party Computation," Proc. 11th Internat. Conf. on the Theory and Application of Cryptology and Inform. Security (ASIACRYPT '05) (Chennai, India, 2005), published in Advances in Cryptology, Lecture Notes in Comput. Sci. (LNCS 3788) (B. Roy, ed.), Springer, Berlin, Heidelberg, New York, 2005, pp. 136–155.

[5] V. Kolesnikov and T. Schneider, "Improved Garbled Circuit: Free XOR Gates and Applications," Proc. 35th Internat. Colloquium on Automata, Languages and Programming (ICALP '08) (Reykjavik, Ice., 2008), published in Automata, Languages and Programming, Lecture Notes in Comput. Sci. (LNCS 5126) (L. Aceto, I. Damgård, L. A. Goldberg, M. M. Halldórsson, A. Ingólfsdóttir, and I. Walukiewicz, eds.), Springer, Berlin, Heidelberg, New York, 2008, pp. 486–498.

(Manuscript approved June 2009)



VLADIMIR KOLESNIKOV is a member of technical staff in Bell Labs' Enabling Computing Technologies research domain in Murray Hill, New Jersey. He received his Ph.D. in computer science from the University of Toronto, in Ontario, Canada. His interests include key exchange, secure multiparty computation, network security, and foundations of cryptography. ◆