# Secure Computation of Surveys *

Joan Feigenbaum †
Yale University
*feigenbaum@cs.yale.edu*

Benny Pinkas
HP Labs
*benny.pinkas@hp.com*

Raphael S. Ryger ‡
Yale University
*ryger@cs.yale.edu*

Felipe Saint Jean §
Yale University
*felipe.saint-jean@yale.edu*

**Abstract**

We describe the design and implementation of a system for conducting surveys while hiding the information provided by the respondents. We use the CRA Taulbee Survey of faculty salaries in computer science departments as a concrete example in which there are real privacy concerns but in which participation is too large and uncoordinated for direct application of known secure multiparty function evaluation protocols. Our system extends earlier work considering privacy in auctions. We adopt the approach of designating a small number of parties to do the main secure computation, but we go farther in addressing the reality of haphazard input arrival, and possible non-arrival, so that "the function," in the usual sense, is not known until it is decided at some point to cease collecting inputs, at which point the participants at large—humans and machines—cannot be expected to be available for any interaction.

A major impediment to acceptance of secure-function-evaluation technology in practice is the fundamental incompatibility of privacy preservation without trusted parties with "sanity checking" of inputs. For the CRA Taulbee Survey, we show that a reasonable partial remedy is possible.

## 1 Surveys and Privacy

The term *survey* can reasonably and usefully be taken to be quite general, subsuming referenda, elections ("Whom would you prefer in office?"), and sealed-bid auctions ("How much would you be willing to pay?").

A survey solicits of participants information to which they have privileged access so that this information may be fed into some computation whose result is of interest to the conductor of the survey and, possibly also, at least in some indirect fashion and to some degree, to the participants. The participants' privileged access to the information asked of them may occasion concern regarding loss of privacy in revelation or may not at all; the perceived penalty for participation is as apt to be the required investment of time and effort. Surveys vary no less in their payoff aspect. The participant may gain no more than the satisfaction of manifesting a spirit of cooperation; or, as in a referendum or an election or an auction, the participant may have a clear stake in the result of the "survey" computation, which he would wish to modulate through participation.

*This work was supported by the DoD University Research Initiative (URI) administered by the Office of Naval Research.
†Supported in part by ONR grant N00014-01-1-0795 and NSF grant ITR-0331548.
‡Supported by ONR grant N00014-01-1-0795 and US-Israel BSF grant 2002065.
§Supported by US-Israel BSF grant 2002065 and ONR grant N00014-04-1-0725.

If the stakes for the participants are high, a chief concern *of the participants* is that their inputs be transmitted faithfully and processed properly, especially if privacy concerns force the processing out of public scrutiny. (In addition, when the stakes are high, all parties may be concerned that participants not "game" the system.) On the other hand, if the stakes for the participants are relatively low, a chief concern *of the conductor of the survey* (and of anyone who does have an interest in the faithfulness of its results) is that participants' responses be correct. If, further, the participants have significant privacy concerns while having little to gain from participation, the conductor of the survey and those interested in its results have an additional major concern, namely, non-participation. This is the configuration of issues raised by a currently contemplated reworking of the Taulbee Survey, occasioning the present research effort.

### 1.1 The Taulbee Survey

The annual Taulbee Survey has been conducted since 1970 by the Computing Research Association (CRA), which is an organization of more than 200 North American organizations active in computing research, mostly academic departments of computer science. The Taulbee Survey is the primary source of information on North American faculty and Ph.D. production in computer science and computer engineering. (The survey is available at [1].)

While the survey provides a large pool of data, our main focus is the salary-analysis portion of the survey. Each department is asked to report the minimum, median, mean, and maximum salaries for each rank of academic faculty (full, associate, and assistant professors and non-tenure-track teaching faculty) and the number of persons at each rank. The survey divides the departments into four tiers, three containing twelve departments each and the fourth containing all the remaining departments. For each combination of tier and academic rank, the survey publishes the following data: (1) the minimum, maximum, and mean of the reported salary minima and maxima, (2) the mean of all salaries, and (3) the median of all salaries. Note that the exact median of the salaries cannot be computed given the data reported by the departments, and therefore the survey publishes an approximation that is computed as the median of the mean salaries reported by the departments.

Recently, the CRA started running a prototype of a new version of the survey in which departments are required to provide complete, anonymized lists of the salaries of their faculty. These data are kept secret at the CRA (which essentially serves as a trusted third party) and enable the computation of more accurate statistical data, such as the exact values of the median, other percentiles, and statistical variances. Not surprisingly, some departments have presented strong objections to the new form of the survey, citing legal issues preventing them from disclosing salary information or other privacy reservations (for example, in many departments it is rather easy to identify the professor receiving the highest salary).

It appears that, unless the privacy concerns raised can be mitigated, the new version of the survey, intended to be improved, may instead be rendered invalid by a significant incidence of non-participation.

## 2 The Suggested System Architecture

Since surveys compute functions whose inputs are provided by the participants, a natural first approach to implementing them with privacy guarantees is to use generic constructions for secure multiparty function evaluation (SFE). This section describes the drawbacks of straightforward implementation along these lines and suggests a different architecture.

## 2.1 The Disadvantages of Straightforward Implementation by SFE

The minimum/maximum/mean functions computed by the survey can be computed by relatively small arithmetic or Boolean circuits, and therefore the overhead of generic secure protocols for computing these functions is reasonable. Some generic protocols are only secure against collusions of less than one half or one third of the participants, but this does not seem to be a problem for the Taulbee Survey application, because it is hard to conceive of a group of, say, six department heads that collude to find salaries in a different department in their tier.

The major, and crucial, drawback of the generic protocols is their interactive nature, the fact that the protocols require every party to be involved in more than a single round of communication, each round depending on the messages that the other parties sent in the previous round. In the case of the Taulbee Survey, it seems impossible to require twelve busy computer science department heads to be online at the same time. For the fourth-tier computations, some 160 department heads would have to interact!

While the picture of 160, or even just twelve, human department heads interacting online at a designated time is particularly absurd, the problem cannot be solved satisfactorily simply by automation. Serious practical problems include (1) the need to develop the SFE software for multiple computing platforms, so as not to impose a platform requirement on the participants; (2) the need for the participants to agree to the introduction of the SFE software into their machines; (3) the need to deploy the software on a very large number of machines and keep it version-synchronized; and, above all, (4) the need that all participating machines be available and functioning properly simultaneously during the computation. The latter issue alone, just the machine analogue of the problem of depending on many humans being online simultaneously, even if less absurd, makes this approach completely impractical.

## 2.2 The Suggested Architecture

We suggest a separation between parties that provide input to the survey/computation, which we designate as *input providers*, and parties that perform the computation, which we designate as *computation servers*. This is a natural taxonomy, because input providers have no motivation, or need, to participate in the computation. They need only to be given the opportunity to provide their inputs and to be assured that their privacy will be preserved. On the other hand, the computation servers can run a fine-tuned implementation of a generic SFE protocol, even if it requires much computation in each server and much interaction among the servers. These servers can be maintained specially to guarantee a high level of quality of service.

Ideally, the architecture comprises $N$ input providers and $M$ computation servers. An input provider is only required to send a message to all computation servers (possibly sending encrypted versions of these messages to a single server that forwards them to the corresponding servers). The protocol could involve many rounds of communication among the servers and be secure against a large collusion of them (ideally, with respect to preservation of privacy, against any collusion of fewer than $M$ computation servers). Furthermore, in the Taulbee Survey application, a department head that wants to have a better assurance of privacy could, in consultation with CRA, volunteer some local party in her own department to run a computation server. The only requirement of this server is that it provide a high quality of service, integrity (avoiding intentional collusion with other servers), and data security in the conventional sense (avoiding compromise by outside parties).

This architecture is similar to the architecture used by Naor *et al.* in [3] for auction protocols, which used only two servers. Indeed, as noted, the survey scenario is similar to the sealed-bid auction scenario, which involves many parties that provide inputs (bids) but have no need to participate in the protocol otherwise.

# 3 The Implemented System

## 3.1 Overview

We have implemented a privacy-preserving system for computing the Taulbee Survey. The current implementation derives from the protocol of Naor *et al.* for computing auctions [3], altered to simplify the computation required at data-entry time and to make the core $M$-party, generic-SFE component completely modular. The system supports an arbitrary number of input providers and, currently, as in the protocol of [3], two computation servers ($M = 2$). It is secure against any coalition of semi-honest adversaries that does not control both—when generalized, all $M$—computation servers. The system is facilitated by a control server, to be run at CRA, presumably, for the Taulbee Survey, providing administrative functions including registration of input providers, supervision of the submission of data, and launch of the core SFE computation. Input providers (parties to the survey multiparty computation to be implemented) and survey administrators (not parties, in the original sense) interact with the control and computation servers via Web interfaces.

We compute our survey results securely, ultimately, using Yao's two-party SFE protocol, which applies to functions represented as circuits with binary gates [4]. In preparation for the launch of the Yao SFE, our system must collect inputs; and it must generate the circuit that will accept the submitted inputs and compute the appropriate function. Whatever the privacy guarantees of the core SFE itself, there is ample opportunity in the surrounding protocol to compromise privacy, especially when inputs submitted at odd times must be stored for an extended period until the computation proper is run. We address this problem by splitting the inputs into shares upon submission and reassembling the shares as the initial part of the circuit to be evaluated.

Consider a circuit $C$ with $L$ input wires that computes the desired results of the survey, privacy concerns aside. The input of each input provider corresponds to a subset of the input wires of the circuit. Define a circuit $C'$ with $2L$ input wires, where $L$ of these input wires are *red*, and $L$ of them are *blue*. The circuit $C'$ is generated by taking every input wire of $C$ and defining it to be the exclusive-or (XOR) of one red input wire and one blue input wire.

Collection of input from the participants proceeds as follows. Each input provider defines, for each of its input bits, two random bits (red and blue) whose XOR is equal to the input bit. It then sends all its red bits, together with the indices of the corresponding wires, to one of the servers, and similarly sends all its blue bits to the other server. This is the only operation done by the input provider, and it can be done independently of other input providers.

A database on the control server tracks the submission of input data by the participants. At some point, at the discretion of an administrator interacting with the control server, the control server instructs the two computation servers to engage in Yao's two-party SFE to compute the output of $C'$, which is equal to the output of $C$.

## 3.2 Some Implementation Details

**User experience:** The user registers as a participant using a Web form. Once registered, she can invoke another Web form into which she may enter her survey input (the faculty-salary list in the case of the Taulbee Survey). At this stage the data are still local to her machine and are not submitted to the servers. When she clicks the "Submit" button, JavaScript code running in her browser generates "red" and "blue" shares for every input bit. The set of red bits is sent to one computation server over an encrypted SSL channel, and the blue bits are sent to the other server. After receiving the bits, the computation servers send acknowledgements to the control server, which records the data submission and sends an acknowledgement to the user.

**The computed function:** The availability of the complete list of salaries enables the computation of more statistical data than is currently produced by the survey. In order to demonstrate the feasibility of the circuit-based solution, we use it to compute a sorted list of all the salaries submitted in each tier at each rank. It is then a simple matter to change this circuit to output values such as the maximum and minimum, the median, and other quintiles. Alternatively, the sorted list for the entire tier-rank may be deemed acceptable itself as output, considering that the very purpose of the survey in publishing various aggregate statistics is to convey just this distribution in anonymized fashion. The latter approach admits arbitrary postprocessing of the tier-rank sorted lists, no longer requiring SFE, to produce the statistics to be published.

The first level of the circuit contains XOR gates that reconstruct the input from the red/blue shares. The circuit then uses an odd-even sorting network to sort the inputs. The size of this network is $O(n \ln^2 n)$ comparisons.[1]

**The code:** The two-party protocol is based on the Fairplay system [2]. Fairplay receives a program written in a special high-level programming language, compiles it in a first stage into a circuit, and then, in a second stage, generates two Java programs that implement Yao's protocol for this circuit.

Because our system computes a very specific family of circuits, we do not use the Fairplay compiler but rather a special program whose input is the number of inputs and their length in bits and whose output is a circuit implementing an odd-even sorting network for sorting these inputs. We feed this circuit to the second stage of the Fairplay system, which generates the programs implementing secure computation for this circuit. We then run these programs to compute the sorted list of the inputs.

**Performance measurements:** The size of the current implementation of the circuit is dominated by an expression of about $3N \ln^2 N \cdot \ell$ gates, where $N$ is the number of inputs, and $\ell$ is the number of bits in a single input. A simple PC can handle about 250 inputs of 10-bit numbers, which result in slightly more than 200,000 gates. We estimate that a state-of-the-art server can handle about 800 inputs of the same length. The major bottleneck is the memory consumption of the program, which is linear in the number of gates. The computation itself takes several minutes.

### 3.3 Input Checking

In a system in which, by design, the inputs are to remain private, how can erroneous inputs be caught? This question is particulary acute in a survey in which the stakes are low for the participants, so that they have little incentive to exercise great caution in their data entry. In the Taulbee Survey, even when participants have considerable interest in the overall statistics that will emerge through the computation and be published, these same participants may not especially care that their own data be entered flawlessly. Here, the interest of the community as a whole and of its "agent," CRA, does not translate directly into a powerful incentive for the individual department head to get her own salary-data entry exactly right. (Note that this unhelpful configuration of incentives does not obtain in referenda or auctions.)

Many solutions that quickly come to mind must be dismissed on various grounds, including, of course, erosion of the proposed system's privacy guarantees, but also the difficulty or impossibility of calibrating adequately useful thresholds and the impracticality of administering the propagation of one year's input data to the participants' machines or the computation servers of the next year.

Fortunately, in the case of the Taulbee Survey, the *published* data of one year, the output of the secure computation, can be brought to bear in assisting the well-meaning participant in catching her own input errors. The distribution, per faculty rank, of the newly entered department data can be displayed

[1] The sorting can also be implemented using a mix network. However, the advantage of the circuit-based construction is that it can be easily adapted to output the values in specific locations in the sorted list, *e.g.*, the items in the 10th, 50th, and 90th percentiles, without revealing any other information about the other inputs.

in juxtaposition to the published distributions of the totality of the previous year's data for the respective faculty ranks in the tier of the participant's department. The department head herself is in the best position to decide whether the comparison is well explained or suggests faulty data entry on her part. This error checking is entirely local to the participant and her machine. The distributions are juxtaposed by JavaScript code running in the participant's browser, and the checking is done by the participant alone; so the privacy properties of the system are maintained.

# 4 Open Problems and Directions

This extended abstract describes work in progress. There are different directions in which the current system can be improved. First, it could use a cryptographic protocol that provides security against malicious parties (*e.g.*, a variant of the protocol given in [3]) or, more importantly, a protocol with more than two computation servers. In particular, it would be interesting to find a multiparty protocol for the computation servers that is more efficient than the generic multiparty protocols.

Another direction is improving the implementation. This could include improving the circuit implementation of the sorting network and, in particular, using different kinds of sorting networks and optimizing the network size for inputs whose length is not a power of two (this type of inputs is rarely treated in the literature). Another issue is improving the memory consumption of the implementation, because this is its major bottleneck. In particular, because we know that gates in the circuit are used in a certain order, we might be able to force the memory allocation dynamically to correspond to the set of gates that are about to be evaluated, rather than allocating memory to all gates. Last but not least, it is important to find applications that require the privacy that is offered by secure protocols.

# References


[1] CRA Taulbee Survey, http://www.cra.org/statistics/.

[2] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella, *Fairplay—A Secure Two-Party Computation System*, Proc. of the 13th Symposium on Security, Usenix, 2004, pp. 287–302.

[3] M. Naor, B. Pinkas and R. Sumner, *Privacy Preserving Auctions and Mechanism Design*, Proc. of the 1st Conference on Electronic Commerce (EC), ACM, 1999, pp. 129–139.

[4] A. C. Yao, *How to generate and exchange secrets*, Proc. of the 27th Symposium on Foundations of Computer Science (FOCS), IEEE, 1986, pp. 162–167.