Highly Efficient Secrecy-Preserving Proofs of Correctness of Computations and Applications

Michael O. Rabin[*]
Harvard University SEAS
Cambridge, MA 02138
U.S.A.
rabin@seas.harvard.edu

Rocco A. Servedio[†]
Columbia University
New York, NY 10025
U.S.A.
rocco@cs.columbia.edu

Christopher Thorpe[‡]
Harvard University SEAS
Cambridge, MA 02138
U.S.A.
cat@seas.harvard.edu

Abstract

We present a highly efficient method for proving correctness of computations while preserving secrecy of the input values. This is done in an Evaluator-Prover model which can also be realized by a secure processor. We describe an application to secure auctions.

1 Background and Motivation

Zero Knowledge Proofs come in a number of flavors. One is direct ZKPs for membership in a NP language, for example proofs that a graph is 3-colorable. These proofs are usually phrased in terms of the particular problem they address, for example talking about graphs and their representations. Another approach deals with circuits and the bit-inputs resulting in certain outputs. This approach is of course very comprehensive since other problem representations are directly translatable into problems about circuits.

There is an extensive literature dealing with ZKPs via encryptions, especially homomorphic encryptions. Verification of processes such as electronic elections or auctions is done via encrypting the relevant numbers such as vote counts or bids, and performing operations such as additions or comparisons on these numbers in their encrypted form. In [9] for example (see the literature quoted there), a protocol is proposed for conducting secure and secrecy preserving auctions. Bidders submit bids to an Auctioneer in an encrypted and committed manner. The Auctioneer posts the encrypted bids on a bulletin board. He then opens the bids and computes, according to the posted auction rules, who the winner(s) is (are) and their payments. The Auctioneer then posts a publicly verifiable Zero Knowledge proof for the correctness of the results. This can be done in a manner revealing the identities of the winners and their payments or, if so desired, concealing that information. But in any case, the bids of all other bidders except for those of the winners remain secret. The only trust assumption made is that the Auctioneer, who knows the bid values, will not reveal that information. The protocol of [9] employs Paillier's homomorphic encryption and proofs of order relations between bids, and correctness of other operations on bids are presented by and verified on encrypted values.

It was shown in [9] that the protocols given there are practical and that currently available computing power suffices to implement auctions with thousands of bidders within reasonably practical time. Still, that solution employs special encryption functions and the basic Paillier encryption is a relatively heavy computation.

In the present paper we present a model of an Evaluator-Prover (EP) who receives input values $x_1, \ldots, x_n$ which are elements of a finite field F_p where p is, say, a 128-bit prime. The Evaluator-Prover computes a function value $y = f(x_1, \ldots, x_n)$ by a publicly announced and agreed upon straight line computation (program) SLC. The EP then publishes the value y and supplies a proof of the correctness of the computation. The proof of correctness can be verified by anybody and this verification method ensures that the probability that an incorrect published result will not be detected is smaller than 2^{-k}, where k is a security parameter. Furthermore, the proof does not reveal anything about the input values or any intermediate results of the computation, except for what is implied by the published outcome of the computation. The generality and efficiency of this model allows numerous applications.

The main idea of the secrecy preserving verification is to represent every number $x \in F_p$ involved in the SLC by a randomly constructed representing pair $X = (u_1, u_2)$ such that $x = u_1 + u_2$. For the verification of correctness the

[*] Supported in part by NSF award CCR-0205423.

[†] Supported in part by NSF award CCF-0347282, by NSF award CCF-0523664, and by a Sloan Foundation Fellowship.

[‡] Supported in part by NSF award CCR-0205423.

EP prepares translations of the SLC where for example $x, y, x+y$ (an addition step) is translated into $X = (u_1, u_2)$, $Y = (v_1, v_2)$, $W = (w_1, w_2) = X + Y$. The EP posts commitments to all numbers in the translations. The Verifier will randomly choose, say, the first coordinate, ask the EP to reveal (de-commit) u_1, v_1 and w_1, and check that $u_1 + v_1 = w_1$. A careful arrangement of the translation process ensures that in the verification only truly independently random numbers $x, y, u, v, \cdots \in F_p$ and their sums or products $u + v$ or $u \times v$ are revealed and checked.

The advantages of this method are manifold. Working with single or double precision integers and their usual arithmetic operations rather than with bits at the circuit level is considerably more efficient. Also, the translation of high-level operations into circuits raises the question of the correctness of the translation itself. Finally, expressing the computation to be verified directly in terms of the numbers and operations involved is more understandable and convincing to general users.

When it comes to verification via encrypted values, in previous approaches such as [9] there is the need to employ special encryptions such as Paillier's encryption, requiring special intractability assumptions. Also, the operations on encrypted values involve computations with numbers with thousands of bits and are quite slow. Experimental comparison between conducting a secure verifiable auction using the method proposed here, and doing the same using [9], shows a hundredfold efficiency improvement.

The applications of ZKP methodology to the conduct of secure secrecy preserving auctions in particular pose stringent requirements of efficiency on the one hand and of understandability and acceptability by the financial and business communities on the other hand. We feel that in this context the present method has clear advantages over other solutions involving homomorphic encryptions, multi-party computations, or reduction to obfuscated circuit computations, important as these approaches are on the theoretical level.

2 Model and Definitions

Our computations are performed with elements of a finite field F_p, where p is a moderately large (say 128 bits) prime. Elements of F_p will be denoted by lower case Roman letters x, y, z, u, v, w, etc. and will be referred to as *numbers*. Computations with numbers are, of course, performed mod p.

Let $x_1, \ldots, x_n$ be elements of F_p, sometimes referred to as *inputs*. A straight line computation (SLC) on these inputs is a sequence of numbers

$$x_1, \ldots, x_n, x_{n+1}, \ldots, x_L \tag{1}$$

where for every $n < m \leq L$, there are two indices $i, j < m$ such that $x_m = x_i \circ x_j$ where $\circ$ is one of $+$, $-$, or $\times$.

The number x_L is called the output or result of the straight line computation. Clearly x_L is the value of a polynomial function $f(x_1, \ldots, x_n)$ of the input values.

We can also generalize our notion of a SLC to involve addition and multiplication by publicly known constants from F_p, and to include the inverse operation $x_m = {x_i}^{-1}$, allowed when $x_i \neq 0$. Our results readily extend to this general case as well, though we do not treat it here for the sake of simplicity.

We assume n parties $P_1, \ldots, P_n$, respectively holding the input values $x_1, \ldots, x_n$. The parties wish to perform the straight line computation (1) on the input values and obtain the result $x_L = f(x_1, \ldots, x_n)$. They want to accomplish this by a secrecy preserving method, revealing nothing about the input values or the intermediate values in the computation, beyond what is implied by the value of the result x_L.[1] At the same times the parties, and perhaps others, want to be certain that the revealed value x_L is the correct result of the straight line computation (1). Thus the protocol must provide a secrecy preserving proof of correctness.

These requirements give rise to the following definitions.

Definition 1 *An Evaluator-Prover (EP) for the SLC (1) is an entity which, upon receiving input values $x_1, \ldots, x_n$, outputs the value $x_L = f(x_1, \ldots, x_n)$ and engages in a proof of correctness to certify correctness of the result value.*

Definition 2 *An Evaluator-Prover is* secrecy preserving *if the proof of correctness does not reveal anything about the input values or the intermediate values in the SLC (1) except for the information implied by the output value $x_L = f(x_1, \ldots, x_n)$. An EP is* trusted *if it outputs or reveals only x_L and the proof of correctness.*

In the real world, an example of a trusted Evaluator-Prover would be an auctioneer AU. The input values to the computation are the values of bids submitted by parties participating in the auction.[2]

There are protocols that ensure that the auctioneer cannot reveal any bid before the closing of the auction or change or suppress bids after the closing of the auction. The extent of trust we place in AU is that he will not reveal any information about the bids except for the outcome of the auction and what is implied by announcement of the outcome. For example, in a Vickrey auction where the item goes to the highest bidder at the price bid by the second highest bidder, the announcement will reveal the identity of the winner.

[1] For example, if $x_L = x_7 - x_{11}$ and the value of x_L is revealed to be 0, then it follows that $x_7 = x_{11}$.

[2] For the application to auctions we also require comparisons such as "$x_i \leq x_j$." We show in Section 9 how our secrecy preserving proofs of correctness can be extended to deal with comparisons.

Whether the winner's payment will be revealed depends on the announced rules of the auction. Our protocols can also enforce secrecy of that payment, if so desired.

The rationale for this partial trust model is that illegally and selectively leaking out bid values *before* the closing of the auction, or announcing a false auction result, can lead to collusions greatly benefiting some bidders and the auctioneer. Our protocols completely prevent such malfeasance. On the other hand, leaking out bid values *after* the end of an auction helps bidders who received such information in strategizing for future similar auctions. The value of this information advantage is, however, relatively limited. Consequently the auctioneer, who has his business reputation to guard, has a substantial incentive not to leak out information after the conclusion of auctions.

Another model is to implement the trusted Evaluator-Prover by a secure co-processor. The secure processor is a closed device for which all outputs are publicly observable. The processor is trusted not to output any information beyond that specified by the protocols. The published proof of correctness assures the participants that the output result is really the correct result of the SLC. The implementation of this model, dealing with some of the subtleties it entails, will be discussed in Section 10.

3 Overview of the method: Representations, Translations and Aspects

Representations. In order to enable secrecy preserving proofs of correctness, the parties $P_1, \ldots, P_n$ and the Evaluator-Prover represent the inputs and the intermediate values in the SLC by pairs of numbers from F_p. In the following we shall use capital letters X, Y, Z, U, V, etc. to denote elements of $F_p \times F_p$, i.e. pairs of numbers from F_p.

Definition 3 *We say that* $U = (u_1, u_2)$ represents $u \in F_p$ *if* $u = u_1 + u_2$. *We shall denote* $u_1 + u_2$ *by* $\mathrm{val}(U)$. *A participant in the protocol will create a random representation* U *of a number* u *by randomly choosing* u_1 *from* F_p *and setting* U *to* $(u_1, u - u_1)$. *Clearly* $\mathrm{val}(U) = u$.

In particular, a random representation Z of zero is obtained by randomly choosing z from F_p and setting Z to $(z, -z)$. We note that at the bit level, Kilian in [6] (inspired by unpublished work of Bennett and Rudich) used a similar representation scheme with "pair blobs" to represent binary values (see also [1]).

The high-level idea of our protocols is that a verification of an operation in the SLC will be implemented by randomly selecting and revealing either the first or the second coordinates of the pairs representing the numbers in question. The idea is that revealing just one coordinate of a pair reveals nothing about the value of the pair. We give details and proofs in the following sections.

Translations. The secrecy preserving proof of correctness of the published result of the SLC is achieved by a process of "translation" of $x_1, \ldots, x_L$ into a sequence $TR(SLC)$ of at most $O(L)$ pairs.The first n pairs in the translation, denoted $X_1, \ldots, X_n$, represent the input values $x_1, \ldots, x_n$. The pairs $X_{n+1}, \ldots, X_{L-1}$ represent intermediate values used in the SLC, and play an important role in verifying the correctness of the SLC. The final pair in the translation represents the output x_L of the computation, i.e. the value of this final pair is $x_L = f(x_1, \ldots, x_n)$.

The computations $x_m = x_i \circ x_j$, where $\circ$ is one of $+, -, \times$, will be translated in a natural way into operations on pairs $U = (u_1, u_2)$, $V = (v_1, v_2)$, $W = (w_1, w_2)$ representing x_m, x_i, x_j. For example $x_m = x_i + x_j$ is translated into $W = U + V$, i.e. ordinary vector addition. Subtraction is entirely similar to addition, but the translation of $x_m = x_i \times x_j$ is slightly more complicated and is described in Section 5.

Aspects. Ultimately, to achieve a probability less than 2^{-k} of accepting a false result of the SLC, we shall require $K = O(k)$ randomly created translations of the SLC. (We shall see below that $K = \gamma k$ is sufficient where the constant $\gamma \overset{\text{def}}{=} 90$.) As described in Section 7, in the verification procedure the Verifier randomly samples some of these K translations and verifies various "aspects" of the EP's computation in the selected translations. As described in Section 6, these different "aspects" capture different elements that are required for the overall computation to be correct: one aspect deals with consistent representation of the n input values, one deals with correctness of the random representations of zero mentioned above, one deals with correctness of addition steps, and so on.

We now turn to the detailed description of the creation of translations and of the proof of correctness.

4 Inputting and verifying the values $x_1, \ldots, x_n$

We require a commitment function $COM(\cdot)$ and digital signatures for the parties $P_1, \ldots, P_n$. (We give details about the properties we assume for our commitments in Section 8.)

Each party P_m creates K random representations $X_m^{(1)} = (a_1, b_1), \ldots, X_m^{(K)} = (a_K, b_K)$ of his input value x_m. He privately sends x_m, $SIGN_m(COM(x_m))$, and all K quadruples a_j, b_j, $SIGN_m(COM(a_j))$, $SIGN_m(COM(b_j))$ to the Evaluator-Prover EP.

The EP verifies that $x_m = \mathrm{val}(X_m^{(j)}) = a_j + b_j$ for $1 \leq j \leq K$, verifies all the $4K + 1$ commitments, and verifies all digital signatures. If any verification fails, then according to the protocol, the EP rejects P_m's input value.

After all inputs were accepted by the EP, he posts, for every party P_m, all the $2K$ signed commitments $SIGN_m(COM(a_j))$, $SIGN_m(COM(b_j))$, $1 \leq j \leq K$, to the representations of the value x_m.

Every Verifier can check and verify all the digital signatures and thereby verify that the respective commitments were made by the parties $P_1, \ldots, P_n$. Henceforth we shall assume that the signature verifications were successful and that all commitments to pairs representing values are assumed to have originated with the parties.

Next, we present a method of secrecy preserving proofs for the claim by the EP that for every P_m all committed-to pairs $X_m^{(i)}$ represent the same value. As we shall see, the method will establish a useful approximation to the validity of the claim.

Consider two pairs $U = (u_1, u_2)$ and $V = (v_1, v_2)$, where commitments $COM(u_1)$, $COM(u_2)$, $COM(v_1)$, $COM(v_2)$ are posted. We have $\mathrm{val}(U) = \mathrm{val}(V)$ if and only if $(u_1 - v_1) + (u_2 - v_2) = 0$. To prove equality of values of U and V, the EP posts d_1 and d_2, which are claimed to be respectively the differences $(u_1 - v_1)$ and $(u_2 - v_2)$. The Verifier randomly chooses an index $c \in \{1, 2\}$ and requests that EP reveal the values committed to by the posted $COM(u_c)$ and $COM(v_c)$. If $d_1 + d_2 \neq 0$ or $u_c - v_c \neq d_c$, then the Verifier rejects the claim that $\mathrm{val}(U) = \mathrm{val}(V)$. It is clear that if actually $\mathrm{val}(U) \neq \mathrm{val}(V)$, then the probability of the Verifier accepting the claim of equality of values is at most $1/2$.

Consider now two arrays of pairs $T_1 = U_1, \ldots, U_n$ and $T_2 = V_1, \ldots, V_n$ where all commitments to components of all pairs are posted, and the claim is being made that

$$\mathrm{val}(U_m) = \mathrm{val}(V_m) \quad \text{for } 1 \leq m \leq n. \tag{2}$$

The Verifier uses the above verification procedure simultaneously for all couples U_m, V_m of pairs, employing the same randomly chosen c for all couples. If the claim is not true, then the probability of acceptance by the Verifier is at most $1/2$.

We shall say that arrays T_1 and T_2 are *value-consistent* if (2) holds true.

Let $T^{(i)} = X_1^{(i)}, \ldots, X_n^{(i)}$, $1 \leq i \leq K$, be the K arrays of pairs of elements from F_p, where $X_m^{(i)}$ is the i-th pair submitted to EP by P_m. Denote by $COM(T^{(i)})$ all the $2n$ commitments to the components of the pairs in the array $T^{(i)}$. According to the procedure of submitting input values, all those commitments were posted by the EP. The EP claims that these are commitments to K pair-wise value-consistent arrays. Denoting by $T^{(i)}[m]$ the m-th pair in the array $T^{(i)}$, this means that for every m, all values $\mathrm{val}(T^{(i)}[m])$ are equal.

Fix $\alpha \overset{\text{def}}{=} 5.5$. To validate the EP's claim, the Verifier chooses a sequence of $2\alpha k$ different superscripts[3] (i_1, j_1), $\ldots$, $(i_{\alpha k}, j_{\alpha k})$ uniformly at random from $\{1, \ldots, K\}$. For each value $1 \leq s \leq \alpha k$, the Verifier obtains from the EP a proof, as detailed above, that the arrays $T^{(i_s)}$ and $T^{(j_s)}$ are value-consistent. If all proofs succeed then the Verifier accepts.

Theorem 4 *If the EP's claim that all pairs of arrays (given by their posted commitments) are value-consistent is true, then EP can obviously pass the verification.*

Fix $\beta \overset{\text{def}}{=} 2/3$. To see that this is an effective verification strategy, let us suppose that for every superscript $i \in \{1, \ldots, K\}$, fewer than $\beta K = \beta\gamma k = 60k$ of the arrays are value-consistent with the array $T^{(i)}$. We may view the choices of the pairs of superscripts as being done sequentially, i.e. in the $(s+1)$-st round the pair (i_{s+1}, j_{s+1}) is chosen from the remaining $K - 2s$ superscripts.

Now for $0 \leq s < \alpha k$, in the $(s+1)$-st round, regardless of the outcomes of previous rounds and of the value chosen for i_{s+1}, there are at most $\beta\gamma k = 60k$ superscripts that are value-consistent with i_{s+1} out of the remaining pool of $\gamma k - 2s \geq \gamma k - 2\alpha k = 79k$ possibilities for j_{s+1}. So the $(s+1)$-st pair chosen is value-consistent with probability at most $\frac{\beta\gamma k}{\gamma k - 2\alpha k} = \frac{60}{79}$, and thus is *not* value-consistent with probability at least $\frac{\gamma k - 2\alpha k - \beta\gamma k}{\gamma k - 2\alpha k} = \frac{\gamma - 2\alpha - \beta\gamma}{\gamma - 2\alpha} = \frac{19}{79}$. If the $(s+1)$-st pair chosen is not value-consistent, then the verification survives the $(s+1)$-st round with probability at most $1/2$. So in each of the αk rounds, regardless of what has happened before, the probability that the verification survives that round is at most $1 - \frac{\gamma - 2\alpha - \beta\gamma}{2\gamma - 4\alpha} = (1 - \frac{19}{158})$. Consequently, the overall probability that the Verifier accepts is at most $(1 - \frac{\gamma - 2\alpha - \beta\gamma}{2\gamma - 4\alpha})^{\alpha k} = (1 - \frac{19}{158})^{5.5k}$. Since $0.4942 \approx (1 - \frac{19}{158})^{5.5} < 1/2$, we have proved:

Theorem 5 *Suppose that for the sequence of arrays $T^{(1)}, \ldots, T^{(K)}$, where each array comprises n pairs of numbers from F_p, there is no subset S with $|S| \geq \beta K$ such that every two arrays in S are value-consistent. Then the probability that the Verifier will accept the proof of value-consistency of all couples of arrays in the sequence is at most $1/2^k$.*

5 The Translation Process

Once the input values were submitted in pair representations and accepted by the EP as above, the EP prepares K translations of the SLC (1) as follows. To avoid cumbersome superscript/subscript notation, below we consider one array $T = X_1, \ldots, X_n$ of representations of the n submitted input values.

[3] That is, αk pairs of superscripts (i, j) used to identify the arrays $T^{(i)}, T^{(j)}$ to be compared.

In the computation (1), an input or intermediate result x_i will in general be involved in several subsequent operations $x_i \circ x_j = x_m$. To enable our secrecy preserving proof of correctness, we prepare in the translation, once X_i (a representation of x_i) was inputted or computed, as many new random representations of $\mathrm{val}(X_i)$ as there are involvements of x_i in subsequent computations in the SLC (1).

Definition 6 *Let X be a pair. A* new random representation *X' of $x = \mathrm{val}(X)$ is obtained by randomly choosing z from F_p and setting X' to $X + (z, -z)$, i.e. $X' = X + Z$, where Z is a random representation of* 0.

The EP starts by extending the array $X_1, \ldots, X_n$ by $Z_1, \ldots, Z_s$ each of which is an independent random representation of 0, where $s = O(L)$ is the total number of new representations that will be created in the translation process. Next, if say x_1 occurs in s_1 subsequent computations in (1) (where we count a computation $x_1 \circ x_1$ as having two occurrences of x_1), then the EP extends the translation array by $Y_1, \ldots, Y_{s_1}$. Here $Y_j = X_1 + Z_j$, $1 \le j \le s_1$. The other inputs $X_2, \ldots, X_n$ give rise to additional new representations $Y_{s_1+1}, \ldots, Y_t$ in a similar way. Each new representation employs the next unused Z_j.

Consider now the first operation $x_{n+1} = x_i \circ x_j$ of (1), where on the right hand side we have input values. We first consider the case that the operation $\circ$ is the $+$ operation. To translate this operation, EP chooses from the sequence $Y_1, \ldots, Y_t$ the first new representations of x_i and x_j. Call these, to avoid double indices, $Y' = (u_1, v_1)$ and $Y'' = (u_2, v_2)$. The translation of $x_{n+1} = x_i + x_j$ is

$$X_{n+1} = Y' + Y'' = (u_1 + u_2, v_1 + v_2). \qquad (3)$$

Next EP creates a first new representation NX_{n+1} of $\mathrm{val}(X_{n+1})$, which of course equals $x_i + x_j$, by employing Z_{t+1}, the next unused representation of 0 :

$$NX_{n+1} = X_{n+1} + Z_{t+1} \qquad (4)$$

Now, if x_{n+1} is used s_{n+1} times in the SLC (1), the EP creates s_{n+1} new random representations of x_{n+1} by:

$$Y_{t+1} = NX_{n+1} + Z_{t+2}, \ldots, Y_{t+s_{n+1}} = NX_{n+1} + Z_{t+1+s_{n+1}} \qquad (5)$$

Note that in creating the new representations (5) we use the first new representation NX_{n+1}, rather than the representation X_{n+1} of x_{n+1}. The reason for that will become clear in the proof for the secrecy preserving nature of the proof of correctness.

In the other case, where $x_{n+1} = x_i \times x_j$, the translation is more complicated. Let again $Y' = (u_1, v_1)$ and $Y'' = (u_2, v_2)$ be the new representations of x_i and x_j, as above. We obtain the representation X_{n+1} of $x_{n+1} = x_i \times x_j$ via four intermediate steps:

$$\begin{aligned} X'_{n+1} &= (u_1 v_1, 0) + Z_{t+1} && (6a)\\ X''_{n+1} &= (u_1 v_2, 0) + Z_{t+2} && (6b)\\ X'''_{n+1} &= (u_2 v_1, 0) + Z_{t+3} && (6c)\\ X''''_{n+1} &= (u_2 v_2, 0) + Z_{t+4} && (6d)\\ X_{n+1} &= X'_{n+1} + X''_{n+1} + X'''_{n+1} + X''''_{n+1}. && (6e) \end{aligned}$$

It is clear from the distributive law that $\mathrm{val}(X_{n+1}) = \mathrm{val}(U') \times \mathrm{val}(U'') = x_i \times x_j = x_{n+1}$.

The first new random representation NX_{n+1} and the subsequent new random representations of x_{n+1} are obtained as in (4) and (5), using new successive random representations Z_q of 0 from the given list.

The translation process of the SLC (1) now proceeds inductively, operation by operation, similarly to the translation of $x_{n+1} = x_i \circ x_j$, using new representations of operands and of zero at every stage.

Thus the outcome of the translation process for the case $x_{n+1} = x_i + x_j$ will be:

$$\begin{aligned} TR \;=\; & X_1, \ldots, X_n, Z_1, \ldots, Z_s, Y_1, \ldots, Y_t, X_{n+1},\\ & NX_{n+1}, Y_{t+1}, \ldots, Y_{t+s_{n+1}}, \ldots, X_L. \qquad (7a) \end{aligned}$$

In (7a), all symbols retain their meaning from the above discussion. That is, $X_1, \ldots, X_n$ are representations of the input values; $Z_1, \ldots, Z_s$ are random representations of 0; $Y_1, \ldots, Y_t$ are new random representations of the input values; X_{n+1} is a representation of x_{n+1}, obtained as in (3); NX_{n+1}, is a next random representation of x_{n+1}, obtained as in (4); $Y_{t+1}, \ldots, Y_{t+s_{n+1}}$ are further random representations of x_{n+1}, obtained as in (5), and X_L is a representation of the output x_L.

In the case $x_{n+1} = x_i \times x_j$, the translation will look like:

$$\begin{aligned} TR \;=\; & X_1, \ldots, X_n, Z_1, \ldots, Z_s, Y_1, \ldots, Y_t,\\ & X'_{n+1}, \ldots, X''''_{n+1}, X_{n+1}, NX_{n+1},\\ & Y_{t+1}, \ldots, Y_{t+s_{n+1}}, \ldots, X_L \qquad (7b) \end{aligned}$$

where $X'_{n+1}, \ldots, X''''_{n+1}, X_{n+1}$ are obtained as in (6a)–(6e).

Note that our notation is such that in a translation TR, the pairs $X_1, \ldots, X_n, X_{n+1}, \ldots, X_L$ correspond to the values $x_1, \ldots, x_n, x_{n+1}, \ldots, x_L$ in the SLC (1). We next show that the pairs X_j actually represent the corresponding values x_j.

Theorem 7 *If* $\mathrm{val}(X_i) = x_i$ *for* $1 \le i \le n$*, then* $\mathrm{val}(X_j) = x_j$ *for* $1 \le j \le L$.

Proof: Consider $x_{n+1} = x_i \circ x_j$. The construction of X_{n+1} in the translation by (3) in the case of $\circ = +$, and by (6a)–(6e) in the case of $\circ = \times$, together with the fact that $\mathrm{val}(Z_t) = 0$ for all $1 \leq t \leq s$, implies that $\mathrm{val}(X_{n+1}) = x_{n+1}$. The proof now proceeds by induction on j. □

6 Verifying Aspects of Translations

We recall that each of the parties $P_1, \ldots, P_n$ has created and submitted to EP K representations of their input values. The EP verifies the digital signatures, the commitments, and the fact that each party P_m has submitted K representations of the same value x_m.

Next EP creates K translations $TR^{(j)}$, $1 \leq j \leq K$, of the SLC (1):

$$\begin{aligned} TR^{(j)} \quad = \quad & X_1^{(j)}, \ldots, X_n^{(j)}, Z_1^{(j)}, \ldots, Z_s^{(j)}, \\ & Y_1^{(j)}, \ldots, Y_t^{(j)}, \ldots, X_{n+1}^{(j)}, NX_{n+1}^{(j)}, \\ & Y_{t+1}^{(j)}, \ldots, Y_{t+s_{n+1}}^{(j)}, \ldots, X_L^{(j)}. \end{aligned} \tag{8}$$

The array $X_1^{(j)}, \ldots, X_n^{(j)}$, consisting of the j-th input pairs submitted to EP by $P_1, \ldots, P_n$, is extended by EP to $TR^{(j)}$ in the manner detailed in Section 5.

The EP now posts all the signed commitments to (coordinates of) the input pairs, and commitments to (coordinates of) all the other pairs in all translations. The EP claims that the posted commitments are to K correct translations of the SLC on the same input values. If that is indeed the case he will be able to respond correctly to all challenges by the Verifier. Thus the proof method is complete. All true statements are provable.

The Verifier will verify the correctness of nine of what we shall loosely call "aspects" of the posted translations.

Aspect 0. As demonstrated in Section 4 above, by randomly choosing $\alpha k = 5.5k$ pairs of translations, the Verifier verifies with probability of error smaller than 2^{-k} that for at least $\beta K = 2K/3$ translations, for every party P_m, all submitted pairs represent the same value x_m. (See Theorem 5.) We shall consider that unique x_m to be P_m's input to the SLC.

Every translation involved in the above verification is discarded and is not used in the following verifications of other aspects of the proof. We now describe how aspects $1, \ldots, 8$ are verified for a given fixed translation, which we denote TR.

Aspect 1. For a posted translation TR, (7a) or (7b), we shall say that TR is *correct with respect to representations of 0*, if for all $1 \leq j \leq s$ we have $\mathrm{val}(Z_j) = 0$.

To verify that TR is correct in Aspect 1, the Verifier requests of EP to reveal (de-commit) all coordinates of all pairs Z_j and checks that for each pair the coordinates sum up to 0.

Aspect 2. We say that TR is correct in Aspect 2 if every computation of a new representation NX_j from a representation X_j, in the manner of (4), is correct.

To verify correctness in Aspect 2, Verifier randomly chooses $c \in \{1, 2\}$ and presents c to EP. If $c = 1$ then EP reveals (de-commits) the first coordinate in all computations of $NX_j = X_j + Z_{e(j)}$ within TR. The Verifier checks that the first coordinates of X_j and $Z_{e(j)}$ sum up to the first coordinate of NX_j. He rejects the whole proof if even one of these checks fails. The case $c = 2$ is handled similarly.

Note that if a translation TR does not satisfy the condition $X_j + Z_{e(j)} = NX_j$ for all indices j, then it will be accepted by the Verifier with probability at most $1/2$.

Aspect 3. We say that TR is correct in Aspect 3 if all computations of the new representations $Y_1, \ldots, Y_t$ of the input-representations $X_1, \ldots, X_n$ and all the further new representations of X_j obtained from NX_j in the manner of (5) are correct.

All of these computations are of the form $Y = X_j + Z$ for the input value representations and $Y = NX_j + Z$ for representations of intermediate results of the SLC, where in each case Z is a specific representation of 0 from the list in TR. So the Verifier has to verify the correctness of all these addition operations. This is again done as in the verification of Aspect 2, with probability of error at most $1/2$.

Aspect 4. We say that TR is correct in Aspect 4 if all the translations of addition operations $x_m = x_i + x_j$ of the SLC, in the manner of (3), as well as all additions of the form $X_m = X_m^{'} + \cdots + X_m^{''''}$ arising in translations of multiplications (see (6e) where $m = n + 1$), are correct.

Thus the Verifier has to check all equalities of the form $X_m = Y' + Y''$ and of the form $X_m = X_m^{'} + \cdots + X_m^{''''}$ in the translation. This is again done by checking correctness of additions, with probability of error at most $1/2$.

Aspects 5–8 deal with correctness of the translations of product computations $x_m = x_i \times x_j$. Let X_m be the representation of x_m and $Y' = (u_1, v_1)$ and $Y'' = (u_2, v_2)$, be respectively the representations of x_i and x_j in TR, used in the translation of the product computation.

Aspect 5. We say that TR is correct in Aspect 5 if for all translations of product operations in the manner of (6a)–(6d), the equations

$$X_m^{'} = (u_1 v_1, 0) + Z, \tag{9}$$

where Z is a specific representation of 0 from the list in TR (a different Z for every m), are true.

Again the Verifier randomly chooses $c \in \{1, 2\}$ and presents c to EP. If $c = 1$ then EP reveals for all translations of products the first coordinates w of $X_m^{'}$, z of Z, and u_1, v_1 of Y', Y''. The verifier accepts only if $w =$

$u_1 \times v_1 + z$ is true for all translations of product computations in SLC. If $c = 2$ then EP reveals for all translations of products the second coordinates w' of $X_m^{'}, z'$ of Z. The verifier accepts only if $w' = z'$ is true for all translations of product computations in SLC. Clearly, if TR is not correct in Aspect 5, then the Verifier will accept with probability at most $1/2$.

Aspects 6, 7 and 8 of a translation TR of the SLC deal with the correctness of the translations of $X_m^{''}$, $X_m^{'''}$ and $X_m^{''''}$ respectively, according to (6b), (6c) and (6d). They are defined, and are checked by the Verifier, in a way similar to the treatment of Aspect 5. In each case the probability of erroneous acceptance is at most $1/2$.

7 Proof of Correctness and Error Probability

Putting together the verification procedures described above, we now describe the overall proof of correctness of the posted result x_L of the SLC, and prove an upper bound of $1/2^k$ for the probability of error.

In the first step of verification, the EP posts K translations of the SLC (1) in the form of commitments to all coordinates of the pairs in the translations.

Aspect 0 of the correctness of the translations $TR^{(j)}$, $1 \leq j \leq K$, is that the arrays $X_1^{(j)}, \ldots, X_n^{(j)}$, $1 \leq j \leq K$, of representations of the input values to the SLC are pairwise value-consistent. The Verifier checks this by randomly choosing $\alpha k = 5.5k$ pairs of translations and performing the tests described in Section 4. As described in Theorem 5, if there are fewer than $\beta K = 2K/3 = 60k$ translations with pair-wise value-consistent input value arrays, then the Verifier will accept the whole proof with probability less than $1/2^k$.

Denote by S_1 the translations not involved in testing Aspect 0, and denote by $K_1 \stackrel{\text{def}}{=} K - 2\alpha k$ the number of translations in S_1. Recalling that $K = \gamma k$, we have $K_1 = (\gamma - 2\alpha)k = 79k$.

Consider the case in which at least βK of the original K translations have pair-wise value-consistent input value arrays. Since $K = \gamma k$, at least $\beta K - 2\alpha k = (\beta\gamma - 2\alpha)k = 49k$ of the $K_1 = (\gamma - 2\alpha)k = 60k$ translations in S_1 have pair-wise value-consistent arrays of inputs. The common values $x_1, \ldots, x_n$ represented by the pairs in those consistent arrays are, by definition, the input values of the SLC.

Fix $\delta \stackrel{\text{def}}{=} 29$. The verifications of the correctness of Aspects 1–8 of the translations in S_1 proceed as follows. The Verifier chooses a set of δk translations uniformly from S_1. For each translation $TR^{(j)}$ of these δk, he randomly chooses an integer $r \in \{1, \ldots, 8\}$ and a challenge $c \in \{1, 2\}$ and performs a check for the correctness of $TR^{(j)}$ in Aspect r. If any of the δk checks fail, then the Verifier rejects.

As described in Section 6, if $TR^{(j)}$ is incorrect in Aspect r, it will pass the test with probability at most $1/2$. Consequently, if $TR^{(j)}$ is incorrect in any one of the Aspects 1–8, it will fail its check with probability at least $1/16$. We will use this observation to prove the following:

Theorem 8 *Fix* $\epsilon \stackrel{\textit{def}}{=} 31$. *Suppose that of the* $K_1 = K - 2\alpha k = (\gamma - 2\alpha)k = 79k$ *translations in* S_1, *fewer than* ϵk *translations are correct in all Aspects 1–8. Then the probability that all* $\delta k = 29k$ *of the Verifier's checks succeed is smaller than* $1/2^k$.

Proof: We view the δk choices from S_1 as being done sequentially, i.e. in the $(s+1)$-st round the translation is chosen from the remaining $K_1 - s$ translations. By assumption, for $0 \leq s < \delta k$, in the $(s+1)$-st round, regardless of the outcomes of previous rounds, there are at most $\epsilon k = 31k$ translations that are correct in all aspects 1–8 among the remaining $(\gamma - 2\alpha)k - s \geq (\gamma - 2\alpha - \delta)k = 50k$ translations. So the $(s+1)$-st translation chosen is correct in all aspects with probability at most $\frac{\epsilon}{\gamma - 2\alpha - \delta} = \frac{31}{50}$, and is incorrect in some aspect with probability at least $\frac{\gamma - 2\alpha - \delta - \epsilon}{\gamma - 2\alpha - \delta} = \frac{19}{50}$. By the observation above, this means that regardless of what has happened before, for each value $0 \leq s < \delta k$, the verification survives the $(s+1)$-st round with probability at most $(1 - \frac{\gamma - 2\alpha - \delta - \epsilon}{16(\gamma - 2\alpha - \delta)}) = (1 - \frac{19}{800})$. Consequently the overall probability that all $\delta k = 29k$ of the Verifier's checks succeed is at most $(1 - \frac{\gamma - 2\alpha - \delta - \epsilon}{16(\gamma - 2\alpha - \delta)})^{\delta k} = (1 - \frac{19}{800})^{29k}$. Since $0.4980 \approx (1 - \frac{19}{800})^{29} < 1/2$, the theorem is proved. □

After performing the verifications of pairwise consistency of translations of input values and the verifications of the correctness of Aspects 1–8 of the translations, there remain $K_2 \stackrel{\text{def}}{=} K_1 - \delta k = (\gamma - 2\alpha - \delta)k = 50k$ untouched translations. The verifier now asks the EP to open all the commitments to the components of the pairs $X_L^{(j)}$ in these K_2 translations. If now $\mathrm{val}(X_L^{(j)}) = x_L$ for all these $X_L^{(j)}$, then the Verifier accepts x_L as the result of the SLC.

Now we can upper bound the probability that the Verifier will accept a wrong value for the output x_m of the SLC (1):

Theorem 9 *[Main Theorem.] Assume that the Verifier accepted all components of the proof of correctness, i.e. the proof of pair-wise value-consistency of the arrays of inputs values of the* K *translations, the proofs of correctness of the translations in respect to Aspects 1–8, and the agreement of all revealed values of the pairs* $X_L^{(j)}$. *Then the common revealed* $x_L = val(X_L^{(j)})$ *is the output value of the SLC with probability of error smaller than* $3/2^k$.

Proof: It was shown above that the successful verification of Aspect 0 – the pair-wise value-consistency of the representations of the input values – assures with probability of error at most $1/2^k$ that at least a $\beta = 2/3$ fraction of the $K = \gamma k = 90k$ translations are pair-wise value-consistent with respect to the arrays of the n input values. Thus at least $K_1-(1-\beta)K = (\gamma-2\alpha)k-(1-\beta)\gamma k = (\beta\gamma-2\alpha) = 49k$ of the remaining $K_1 = (\gamma - 2\alpha)k = 79k$ translations are pair-wise input-value consistent. This defines a unique sequence of common values $x_1, \ldots, x_n$ represented by the pairs in these consistent arrays (which form a majority of the K_1 remaining arrays); these values are, by definition, the input values of the SLC.

By Theorem 8, if the translations in S_1 passed the tests on the randomly chosen $\delta k = 29k$ translations, then with probability of error smaller than $1/2^k$, more than $\epsilon k = 31k$ of the translations are correct in all Aspects 1–8. This implies that among the at least $(\beta\gamma - 2\alpha)k = 49k$ pair-wise input-value consistent translations in S_1, at least $(\beta\gamma - 2\alpha)k + \epsilon k - (\gamma - 2\alpha)k = (\epsilon - \gamma(1-\beta))k = k$ translations are also correct in all aspects, this with probability of error at most $1/2^k$. Let S_3 denote any fixed set of k translations that are correct in all aspects.

Now we observe that the probability that the $\delta k = 29k$ translations randomly chosen from S_1 include all k translations in S_3 is

$$\begin{aligned}\frac{\binom{K_1-k}{\delta k-k}}{\binom{K_1}{\delta k}} &= \frac{\delta k(\delta k-1)\cdots(\delta k-k+1)}{K_1(K_1-1)\cdots(K_1-k+1)} \\ &< \left(\frac{\delta k}{K_1}\right)^k = \left(\frac{\delta}{\gamma-2\alpha}\right)^k = (29/79)^k.\end{aligned}$$

Since $0.3671 \approx 29/79 < 1/2$, we have that with probability of error smaller than $1/2^k$, after the $\delta k = 29k$ translations are removed from S_1, there remains at least one translation that is correct, has the correct representations for the inputs values $x_1, \ldots, x_n$, and was not used in any of the verifications. Since the revealed $\mathrm{val}(X_L^{(j)})$ is the same for all the translations $TR^{(j)}$ not used in any of the verifications, that value is the correct output value x_L of the SLC (1). The total probability of error is less than $3/2^k$. □

8 The Verification of Correctness is Secrecy Preserving

We shall conduct our proof of the secrecy preserving property in the random oracle model for the commitment function COM. Thus we assume that $COM : \{0,1\}^{k+128} \rightarrow \{0,1\}^{k+128}$ is a random permutation. Whenever the EP or the Verifier has an argument value $w \in \{0,1\}^{k+128}$, he can call on COM and get the value $v = COM(w)$. To commit to a number $x \in F_p$, the committer randomly chooses a help value $r \in \{0,1\}^k$ and obtains $v = COM(r||x)$. To de-commit v, the committer reveals r and x, and then the commitment to x is verified by calling the function COM. (See [4] for a related but more sophisticated approach to commitments.)

The EP prepares the K translations of the SLC (1) as detailed in Sections 4 and 5, and posts commitments to all the coordinates of all the pairs appearing in the translations, keeping to himself the help values $r_1, r_2, \ldots$ employed in the commitments.

The main idea of the proof is that in the verification process all that is being revealed are randomly independent elements of F_p, and relations of the form $u_1+u_2+\cdots+u_s = v$ or $u_1 \times u_2 = v$, for randomly independent $u_1, u_2, \ldots$ in F_p. The properties of the commitment scheme ensure that nothing can be learned about a value $u \in F_p$ from a commitment to it.

To simplify the proof of the secrecy preserving nature of the verification process, we assume that every party P_j is proper and submits to the EP K randomly independent representations $X_j^{(1)}, \ldots, X_j^{(K)}$ of his input value x_j. Allowing improper parties does not change the essence of the proof and the result.

We shall consider the verifications of Aspects 0–8 of the translations $TR^{(j)}$, $1 \leq j \leq K$, posted by the EP via commitments.

Aspect 0 relates to the pair-wise value-consistency of the arrays of inputs. In the basic step the Verifier requests of the EP to reveal for two representations $X_m^{(i)}$ and $X_m^{(j)}$ of input x_m submitted by party P_m, the values of, say, their first coordinates $u_m^{(i)}$ and $u_m^{(j)}$. The Verifier then verifies that $u_m^{(i)} - u_m^{(j)}$ equals d_1, a value that was posted by EP. Since, according to the protocol, party P_m used random representations of x_m (see the beginning of Section 4), all these first coordinates are independent random elements of F_p.

We recall that every translation $TR^{(j)}$ (see (8)) contains representations $Z_1^{(j)}, \ldots, Z_s^{(j)}$ of 0; new representations $Y_1^{(j)}, \ldots, Y_t^{(j)}, \ldots$ so that every x_m in the SLC has as many new representations as the number of times it is involved in computations of the SLC; and representations $NX_m^{(j)}$ for every x_m resulting from a computation in the SLC. Aspects 1, 2 and 3 respectively deal with the correctness of these Z, Y and NX representations.

The first lemma addresses the Z's:

Lemma 10 *In the set of translations $\{TR^{(1)}, \ldots, TR^{(K)}\}$, any collection of coordinates of representations of 0 which does not contain both coordinates of the same representation, consists of independently randomly chosen numbers from F_p.*

Proof: This follows from the construction of the random representations of 0. □

The next lemma addresses the Y's and the NX's:

Lemma 11 *In the set of translations $\{TR^{(1)}, \dots, TR^{(K)}\}$, any collection of coordinates of the representations $Y_i^{(j)}$ and $NX_m^{(j)}$ which does not contain both coordinates of the same representation, consists of independently randomly chosen numbers from F_p.*

Proof: Every such representation $Y_i^{(j)}$ or $NX_m^{(j)}$ is the result of an operation of the form $Y_i^{(j)} = X + Z$ or $NX_m^{(j)} = X + Z$, where X is some previous pair in $TR^{(j)}$ and Z is a random representation of 0 from $TR^{(j)}$, and where Z is used only once. The result now follows from the previous Lemma. □

Checking Aspect 1 of a translation involves the revelation by the EP of all coordinates of all representations of 0 in a number of translations. By construction of the translations, all representations $(z, -z)$ of 0 were constructed by the EP using independently random choices of z, and no other value in those translations is revealed. Thus the revealed values are randomly independent and randomly independent from any other values revealed in the total verification.

We recall that in a translation TR, the symbols $X_1, X_2, \dots, X_m, \dots, X_L$ denote representations of the values $x_1, x_2, \dots, x_m, \dots, x_L$ of the SLC (1).

Lemma 12 *Let $U = \{X_{(n+1)}^{(j)}, \dots, X_L^{(j)} \mid 1 \leq j \leq K\}$ be the set of all representations of non-input values $x_{n+1}, \dots, x_L$ in all translations $TR^{(j)}$, $1 \leq j \leq K$, of the SLC (1). Then any collection of coordinates of the representations in U which does not contain both coordinates of the same representation consists of independently randomly chosen numbers from F_p.*

Proof: By the construction of a pair $X_m^{(j)}$ in the translation $TR^{(j)}$, if $x_m = x_i + x_j$ in the SLC (1) then $X_m^{(j)} = Y' + Y''$ where Y', Y'' are new random representations of x_i and x_j (see (3)). Thus the claim follows from Lemma 11. If $x_m = x_i \times x_j$ then $X_m^{(j)}$ is constructed from new random representations Y', Y'' of x_i and x_j according to (6a)–(6e). The use of random representations of 0 in (6a)–(6e) establishes the claim. □

Remark. Under the assumption that all parties $P_1, \dots, P_n$ are proper, Lemma 12 extends to the coordinates of the representations $X_1^{(j)}, \dots, X_L^{(j)}$ of all the numbers $x_1, \dots, x_L$ of the SLC (1).

Lemma 13 *Verifying Aspect 2 of a translation TR involves checking equations of the form $u + z = v$ where all the numbers u, z that are revealed (de-commited) are randomly independent elements in F_p.*

Proof: The equations to be simultaneously verified are of the form $X_j + Z_{e(j)} = NX_j$ where $Z_{e(j)}$ is a new random representation of 0 for every equation. The verification is done by checking equations of the form $u + z = v$ where in each case u, z, v are simultaneously the first or simultaneously the second coordinates of X_j, $Z_{e(j)}$ and NX_j. The random independence claim for the u, v now follows from Lemma 12. □

Lemma 14 *Verifying Aspect 3 of a translation TR involves checking equations of the form $u + z = v$ where all the numbers u, z revealed (de-commited) are randomly independent elements in F_p.*

Proof: Verifying Aspect 3 involves verifying all equations of the form $Y = X_j + Z$ for the input value representations and $Y = NX_j + Z$ for representations of intermediate results of the SLC, where in each case Z is a different representation of 0 from the list $Z_1, \dots, Z_s$ of representations of 0 in TR. The result follows from Lemma 12, the construction of $Z_1, \dots Z_s$, and the fact that verifying such an addition of representations (pairs) involves revelation of either all first coordinates or all second coordinates of the pairs in question. □

Lemma 15 *Verifying Aspect 4 of a translation TR involves checking equations of the form $u_1 + u_2 = v$ and $w_1 + \dots + w_4 = w$ where all the numbers $u_1, u_2, w_1, \dots, w_4$ revealed (de-commited) are randomly independent elements in F_p.*

Proof: This follows from the definition of Aspect 4 in a manner similar to the proof of Lemma 14. □

We move directly to the statement that the verification of Aspect 6 is secrecy preserving. The proof for the secrecy preserving nature of Aspects 5 and 7–8 is similar.

Lemma 16 *Verifying Aspect 6 of a translation TR involves checking equations of the form $u_1 \times v_2 + z = w_1$ where the numbers u_1, v_2, z revealed (de-commited) are randomly independent elements in F_p.*

Proof: Verifying Aspect 6 involves checking in TR simultaneously all equations of the form (6b) arising in translations of multiplications $x_m = x_i \times x_j$ of the SLC (1). Such a translation employs unique random representations $Y' = (u_1, v_1)$ and $Y'' = (u_2, v_2)$ of x_i and x_j and a representation $Z = (z, -z)$ of 0. To be verified simultaneously are all additions $X_m^{''} = (u_1 \times v_2, 0) + Z$ in TR. If the challenge is c and $X_m^{''} = (w_1, w_2)$ then all the u_1, v_2, z and w_1 are revealed by the EP and all equations $u_1 \times v_2 + z = w_1$ are checked by the Verifier. By Lemma 11, all the revealed u_1, v_2, z are randomly independent elements of F_p. □

Theorem 17 *The verification of correctness of the K translations $TR^{(j)}$, $1 \leq j \leq K$, of the SLC (1) is secrecy preserving.*

Proof: The verification process involves randomly choosing $2\alpha k = 11k$ translations for verifying Aspect 0 (the value-consistency of the input arrays) and randomly choosing $\delta k = 29k$ arrays for verifying Aspects 1–8.

Let $C_1, \ldots, C_K$ be a collection of coordinates of representations of values from the translations $TR^{(j)}, 1 \leq j \leq K$, such that no C_j contains both coordinates of the same representation (pair). By the construction of the K translations, the values in any C_j are randomly independent from the values in all other C_i's.

Any one of the $(\alpha + \delta)k = 40k$ translations used in the verification is involved in the verification of just one of the Aspects 0–8, i.e. is used only once.

By the detailed analysis given above for the verification of Aspect 0 and in Lemmas 10–16, all the coordinate values from presentations of a translation $TR^{(j)}$ revealed during the verification satisfy the condition on C_j. Furthermore, they are mutually randomly independent values in F_p, except for relations such as $u + z = v$, $u_1 \times v_2 + z = w_1$, etc. dictated by the structure of the translation process. By the above observation on $C_1, \ldots, C_K$, the verification of Aspects 0–8 only reveals some randomly independent elements of F_p and some sums and products of such elements (which could be computed by the Verifier on his own).

Finally, in every $TR^{(j)}$ not used in the verification of Aspects 0–8, the Verifier asks the EP to de-commit both coordinates of $X_L^{(j)} = (u_L^{(j)}, v_L^{(j)})$. The Verifier checks that all the revealed pairs have the same sum $u_L^{(j)} + v_L^{(j)} = x_L$, where x_L is by definition the result of the SLC (1). The revealed coordinates of all the $X_L^{(j)}$ involved in this final step are again randomly independent values in F_p, subject to the condition that the two coordinates of each pair all sum to the same value.

We are working in the random oracle model for the COM function. Thus for all values x of coordinates of pairs in all translations, the values $v = COM(r\|x)$ are randomly independent elements of $\{0,1\}^{k+128}$. □

9 An Application to Auctions

In this section we sketch an application of our method to secure auctions. After touching on security and privacy concerns particular to cryptographic auctions, we augment the basic approach for straight-line computations described above to handle comparison steps $x \leq y$ and summarize a cryptographic auction protocol using our methods.

9.1 Background and Motivation

Cryptographic auctions are an ideal example to illustrate our work in a real-world context. Auction theory has developed complex pricing algorithms for "strategyproof" auctions (that is, a bidder's best strategy is to bid her true utility), but information about one bid being revealed to another bidder could change the outcome of the auction. Moreover, in many applications, such as wireless spectrum auctions conducted by the FCC, bidders do not want their bids to be revealed to other bidders (because it constitutes proprietary business information) yet the auctions must be transparent to comply with Federal regulations.

Thus we require an auction protocol with the following characteristics: 1) it must be practically efficient enough to compute functions of the bids; 2) bids must be secret, in that no bidder can learn anything about any other bid before the deadline to submit a bid; and 3) the results must be able to be proven correct without revealing the original bids. Our method supports all of these requirements: 1) we have demonstrated our protocol's efficiency in empirical tests; 2) other cryptography, such as cryptographic commitments or time-lapse cryptography [11], can enforce bid secrecy until the auction is closed; and 3) the protocol presented in this work issues a correctness proof that reveals nothing about the bids (clearly, it reveals nothing that is not implied by the results).

The extent of trust we place in an auctioneer is that he will not reveal any information about the bids except for the outcome of the auction and what is implied by announcement of the outcome. For example, in a Vickrey auction where the item goes to the highest bidder at the price bid by the second highest bidder, the announcement will reveal the identity of the winner. Whether the winner's payment will be revealed depends on the announced rules of the auction, but if so, then the second highest bidder's bid is also revealed. When the rules demand it, our protocols can enforce the secrecy of auction payments, so that each bidder receives a private proof of the correctness of any payment without learning additional information.

The rationale for this partial trust model is that illegally and selectively leaking out bid values *before* the closing of the auction, or announcing a false auction result, can benefit particular bidders, the auctioneer, and/or the seller. Our protocols completely prevent such malfeasance. On the other hand, leaking out bid values *after* the end of an auction only helps parties who receive such information in strategizing for future similar auctions. The value of this information advantage is, however, relatively limited. Consequently the auctioneer, who has his business reputation to guard, has a substantial incentive not to leak out information after the conclusion of auctions; and as we will see there are other approaches to building secure systems in which such post-auction leaks can be prevented.

9.2 Summary of an Auction Protocol

In [9] (for a more detailed review, see the literature quoted there), a protocol is proposed for conducting secure and secrecy preserving auctions. Bidders choose their bids, encrypt them using a homomorphic encryption scheme, and send commitments to these encrypted bids to an auctioneer; they do this by posting them on a public bulletin board. After all bids are in, the auctioneer announces that the auction has closed, and the bidders submit their encrypted bids to the bulletin board. These can be easily verified against the previously published commitments. The auctioneer then privately opens the encrypted bids and computes, according to the posted auction rules, who the winner(s) is (are) and their payments. He then posts a publicly verifiable Zero Knowledge Proof for the correctness of the results, based on the encrypted bids published on the bulletin board.

This proof can be done in a manner revealing the identities of the winners and their payments or, if so desired, concealing that information. But in any case, the bids of all other bidders except for those of the winners remain secret. The only trust assumption made is that the auctioneer, who knows the bid values, will not reveal that information after the auciton. The protocol described in [9] employs Paillier's homomorphic encryption scheme [8] for bid secrecy and proofs of correctness; his scheme allows these proofs to be verified by using only the encrypted bids.

It was shown in [9] that the protocols given there are practical and that currently available computing power suffices to implement auctions with thousands of bidders within reasonably practical time (on the order of one day for a single computer). Still, that solution employs special encryption functions and basic Paillier encryption is a relatively heavy computation.

Our theoretical framework for secrecy-preserving, provably correct computation described above is extendible for conducting a sealed-bid auction; to complete the necessary set of primitives we now explain how zero-knowledge comparisons of two values can be handled in our protocol. (This is a general extension of the SLC framework independent of the specific application to auctions.) In Section 9.4 we describe some simple optimizations of the basic approach described in the previous sections that give an improvement in efficiency. In Section 9.5 we give an example of how our augmented approach can be used to prove correctness of a Vickrey auction result.

9.3 Translation of Inequalities $0 \leq x \leq B$ and $x \leq y$.

Let $0 < b < p$ be values that satisfy $32b^2 < p$.

We proceed in three steps in this subsection. We first suppose that the Evaluator-Prover has a value $0 \leq x \leq b$, and we explain how the EP can prove that $-b \leq x \leq 2b$. Next, using this first step, we explain how if the EP has $0 \leq x \leq b^2$ he can give a secrecy preserving proof that $0 \leq x \leq 16b^2$. Finally we describe how this enables him to prove that $0 \leq x \leq y \leq 16b^2$ for two values x, y that satisfy $0 \leq x < y \leq b^2$.

So let us suppose that the EP has a value $0 \leq x \leq b$, and wants to prove that $-b \leq x \leq 2b$, i.e. that either $0 \leq x \leq 2b$ or $p - b \leq x < p$. The following construction is an adaptation of a method of Brickell *et al.* [2] to our context.

The EP selects a random value $0 \leq w_0 \leq b$ and sets $w_1 = w_0 - b$. He sets

$$r = \begin{cases} w_0 + x & \text{if } w_0 + x \leq b; \\ w_1 + x & \text{if } b < w_0 + x. \end{cases} \qquad (10)$$

It can be seen that this r is uniformly distributed in the interval $[0, b]$. If a Verifier checks that the pair (w_0, w_1) satisfies the condition $w_1 + b = w_0$ and that for some $\zeta \in \{0, 1\}$, it is the case that $0 \leq w_\zeta + x \leq b$, then the Verifier may infers that $-b \leq x \leq 2b$ is true.

To enable the verification in a secrecy preserving manner, the EP includes in the translations TR a representation X for x; two representations W', W'', for the values w_0 and w_1; and a representation R for the value r defined by (10). We stress that the two representations W', W'' in the translations occur consecutively (these can follow the Z's in the overall translation of the entire computation, see (8)), *but in an order that is randomly chosen by the verifier*. That is, when the translations are being constructed, the EP randomly decides whether the first representation W' will represent w_0 or w_1 (and then the second representation represents the other value).

From the above description, we have that the translation of the statement $-b \leq x \leq 2b$ requires commitments to eight values in F_p (the two components of each of the four pairs X, W', W'', and R). For the actual verification we modify three of the previously described Aspects (Aspects 1, 2 and 3) as we now describe.

In Aspect 1, we shall now also say that a translation TR is *correct with respect to representations of the w's* if for each couple of pairs W', W'' arising in a comparison step as described above, we have $\text{val}(W') = \text{val}(W'') - b$ or $\text{val}(W'') = \text{val}(W') - b$. To verify that TR is correct in Aspect 1, in addition to checking all zeros as described earlier, the Verifier also requests of EP to reveal (de-commit) all coordinates of all pairs W', W'' and checks that for the values corresponding to each pair, it is indeed the case that one of the two equalities holds.

In Aspect 3, we shall now also say that a translation TR is *correct with respect to representations of the r's* if for each comparison step as described above, it is indeed the case that for some $W^\star \in \{W', W''\}$ we have $\mathrm{val}(R) = \mathrm{val}(W^\star) + \mathrm{val}(X)$. To verify that TR is correct in Aspect 2, in addition to checking all computations of $Y_1, \ldots$ as described earlier by choosing a random $c \in \{1, 2\}$, the following moreover takes place. The EP selects the element of $\{W', W''\}$ that corresponds to the correct value of ζ such that $r = w_\zeta + x$; we refer to the element he selects as $W^\star$. If $c = 1$ then EP reveals (de-commits) the first coordinate in all computations of $R = W^\star + X$. The Verifier checks that the first coordinates of $W^\star$ and X sum up to the first coordinate of R. He rejects if even one of these checks fails. The case $c = 2$ is handled similarly.

In Aspect 2, we shall now also say that a translation TR is *correct with respect to the range of the r's* if the new representation R satisfies $0 \leq \mathrm{val}(R) \leq b$. To verify correctness in Aspect 2, in addition to checking all computations of NX_j as described earlier, the EP de-commits *both coordinates* in all computations of R, the new representation of r. The Verifier sums the two coordinates to obtain $\mathrm{val}(R)$ and checks that the two coordinates add up to a value that lies in the interval $[0, b]$. (Note that by following the protocol, the EP ensures that this value $\mathrm{val}(R)$ is r, which is a "fresh" random value from $[0, b]$ independent of everything else seen by the Verifier; thus secrecy is preserved.)

Now let us suppose that $0 \leq x \leq b^2$. The EP wants to enable a secrecy preserving proof that $0 \leq x \leq 16b^2$. We describe an approach by which he can do this; the approach is similar to one given in [3].

By Lagrange's theorem, there exist nonnegative integers x_1, x_2, x_3, x_4 such that

$$x = x_1^2 + x_2^2 + x_3^2 + x_4^2 \quad \text{with } 0 \leq x_1, x_2, x_3, x_4 \leq b. \tag{11}$$

There is an efficient randomized algorithm known that, given x as input, finds a sum of four squares representation (11) for x [10]. Using this algorithm, the EP computes the Lagrange representation (11) and for each of the values x_1, x_2, x_3, x_4, prepares a translation enabling a proof that $-b \leq x_j \leq 2b$ as described above. He creates representations X for x and $X_1, \ldots, X_4$ for x_1, x_2, x_3, x_4. He prepares translations for the computations $x_j^2 = x_j \times x_j$ for $1 \leq j \leq 4$, and for the equality (11). If a Verifier checks the above relations using the representations, then the Verifier knows that $0 \leq x \leq 4 \cdot 4b^2 = 16b^2$.

Finally, let us suppose that $0 \leq x \leq y \leq b^2$. The EP wants to give a secrecy preserving proof that $0 \leq x \leq y \leq 16b^2$. He does this simply by giving a secrecy preserving proof that $0 \leq x \leq 16b^2$ (which he can do since $0 \leq x \leq b^2$), a secrecy preserving proof that $0 \leq y \leq 16b^2$ (which he can do since $0 \leq y \leq b^2$), and a secrecy preserving proof that $0 \leq y - x \leq 16b^2$ (which he can do since $0 \leq y - x \leq b^2$). It is clear that these bounds establish that $0 \leq x \leq y \leq 16b^2$.

In the next section we shall describe an optimization that let us reduce the number of commitments required for a naïve instantiation of the above approach.

9.4 An optimization: more efficient sum of four squares and other sequences of additions.

Here we briefly note an optimization that can be performed to reduce the number of commitments required to perform the sums of four squares in (11) and certain other sequences of operations.

The optimization is to perform a sequence of additions "in one step", similar to our implementation of a multiplication step. Recall that a multiplication step $x_m = x_i \times x_j$ is implemented as follows: after constructing representations $X_m^{'}$, $X_m^{''}$, $X_m^{'''}$ and $X_m^{''''}$, the EP constructs the final X_m as $X_m^{'} + X_m^{''} + X_m^{'''} + X_m^{''''}$ in one step, rather than performing three pairwise additions (which would necessitate representations for the intermediate sums, new representations for their subsequent use in the overall sum, etc.). (See the verification of Aspect 3 described in Section 6.)

A similar approach can be taken when constructing the sum of four squares $x_1^2 + x_2^2 + x_3^2 + x_4^2$. Since the intermediate pairwise sums are not used we may simply perform all three additions at once and save on the intermediate representations that would otherwise be constructed. A similar approach can be taken for any sequence of consecutive additions that occurs anywhere in the SLC.

9.5 Proving Correctness of a Vickrey Auction Result.

In a Vickrey auction participants $P_1, \ldots, P_n$ submit bids $x_1, \ldots, x_n$. The winner is the highest bidder and the price he pays is the second highest price. In this setting the Auctioneer acts as the EP. Without loss of generality, and excluding the case of equal winning bids, we assume that

$$p/32 > b^2 > x_1 > x_2; x_2 \geq x_3, \ldots, x_2 \geq x_n. \tag{12}$$

Thus the EP has to prepare translations enabling a secrecy preserving proof of the inequalities (12). The EP first prepares translations for proving that $0 \leq x_i \leq 16b^2$ for each $i = 1, \ldots, n$. He then proves that $x_2 < x_1$ (by proving that $0 < x_1 - x_2 \leq 16b^2$), that $x_3 \leq x_2$, that $x_4 \leq x_2$, and so on as described in Section 9.3. Thus there are a total of $2n$ proofs that various values v satisfy $0 \leq v \leq 16b^2$.

9.6 Efficiency of the Protocol

We now analyze the number of commitments that this protocol requires. A careful analysis of the translation of the n-participant Vickrey auction computation reveals that $101n$ pairs are constructed within each translation. As described earlier, the secrecy preserving proof involves $90k$ different translations, and thus all in all the posted proof consists of $90k \cdot 101n \cdot 2$ commitments to values in F_p. (The final factor of two is because there is one commitment for each of the two elements of each pair.)

For security parameter $k = 40$ and number of bids $n = 100$, this means around 72.7 million commitments. For pragmatic reasons, to commit $COM(x)$ we employed the SHA-1 cryptographic hash function on x with a random 128-bit help value r: $COM(x) =$ SHA1$(x||r)$. The more sophisticated theoretical approach of [4] could also be used without a significant effect on efficiency. This yields 160 bits of output for each commitment, for a total proof size of approximately 1.45GB with the above parameters. While constructing the proof requires committing to all values, and the entire proof is downloaded by the verifier, examination of the verification process above shows that no more 5% of the committed values need to be verified by decommitment at the end of the protocol. (To check a commitment, the verifier requests indices of the elements to decommit; the EP sends the random seeds and actual elements; then the verifier rehashes their concatenation and checks for equality.)

We have conducted empirical experiments comparing the performance of our protocol on sealed-bid auctions to that of a previously published auction protocol based on homomorphic encryption [9]. Our results bear out our claim that our solution is significantly faster than solutions based on homomorphic cryptography. There is, however, an important time/space tradeoff: the correctness proofs in our solution are very large, because of the large number of commitments necessary to guarantee correctness with high probability. We have therefore included not only calculations of the cost of computing all of the cryptographic hashes (by far the dominant computation) but also estimated the transfer time for the verifier to download the very large proof of correctness. Although we tested the running time of the other operations necessary to construct and verify a proof for a cryptographic auction, these take at most a few seconds and we omitted them from our discussion here. These operations include generating random data, decomposing the sum-of-four-squares representations, and multiplication and addition of values modulo p.

To yield fair comparisons, we executed our tests using the same 2.8 GHz 32-bit Pentium 4 processor used on the homomorphic cryptographic auction protocol in [9] with which we compare our new approach; obviously use of faster 64-bit processors would significantly improve the efficiency in all cases. We estimate that the timing presented here would be improved by a factor of 2 or 3 if run on 2007 state-of-the-art hardware. We also assume a 2.5 megabyte per second transfer rate for the proof download. Times given in Table 1 reflect a security parameter $k = 40$ for our proposed protocol and a 2048-bit public Paillier key in the homomorphic cryptographic setting.

Table 1. Single-Item Auctions of 100 Bids

Operation	Proposed	Homomorphic
Preparing the proof	4.11 minutes	804 minutes
Downloading the proof	9.67 minutes	<1 minute
Verifying the proof	<1 minute	162 minutes

10 The Secure Co-Processor Model and Implementation

Instead of the EP entity, which may be a person or some organizational entity, in this section we propose a Secure Processor Evaluator-Prover ($SPEP$) for the implementation of verifiable secrecy preserving straight line computations. We emphasize that this is a preliminary proposal; instead of giving detailed formal definitions we shall informally specify the properties and assumptions for the $SPEP$.

The secure processor is programmed to perform the functions of the Evaluator-Prover, as previously described, for accepting input values $x_1, \ldots, x_n$, executing the SLC (1) on these values, preparing a proof of the correctness of the computation and outputting (posting) that proof.

We trust the secure processor to only post the proof and not any other information. We do not trust the processor to correctly execute the SLC. Hence the need for a verifiable proof of correctness.

The secure processor may leak out information in a number of ways. For one thing, the format of the posted proof may leak out information on input and intermediate values of the computation through use of spaces, fonts used, format, etc. How to counter such steganographic leaks lies outside the present authors' expertise. (See e.g. [7] for some background on covert channels.)

Second, and more pernicious, the EP requires a considerable stream of random bits for implementing the translations $TR^{(j)}$, $1 \leq j \leq K$. The secure processor can leak out information on input and other values through appropriate choices of random values that will be revealed in the verification process.

Our proposal for dealing with this covert channel is to have an independent secure co-processor RANDOM with a physical random number generator which acts as a universal source of randomness. Upon request from the EP secure processor, RANDOM sends to the SPEP a list of sequentially numbered and digitally signed random values to be used as help values for the COM operation and as values in F_p to be used in the translations. The SPEP must use these random strings in the order of their numbering according to a publicly known protocol. Whenever a random value from the translations is posted as part of the proof of correctness, the SPEP also posts the signed message from RANDOM as proof of origin. The protocol enables the Verifier to check that the posted messages from RANDOM are used in the posted proof in the mandatory order.

The processor is trusted not to output any information beyond that specified by the protocols, and its communications interfaces can be monitored to verify this. The published proof of correctness assures the participants that the output result is really the correct result of the SLC; this means that the validation of the program run by the secure coprocessor need only address information leakage, not program correctness: the program proves itself correct during its normal operation.

The above is only a rough outline of the secure processor and the RANDOM secure co-processor model. Details will be presented in a subsequent publication.

11 Practical Implementation of the Evaluator-Prover Method

For a practical implementation of the EP method, say for use in secure auctions, we make some pragmatic choices.

We choose $k = 40$, giving a total probability of error smaller than $3 \cdot 10^{-12}$. The COM function for a value in F_p, where p has 128 bits, will be implemented by randomly choosing a help value $r \in \{0,1\}^{40}$ and setting $COM(x) = SHA(r\|x) \in \{0,1\}^{120}$. Note that COM is randomly many-to-one. This practically precludes feasible searches even if some partial information about x is available.

The verification of correctness process will not be interactive. The proof of correctness of the translations of the SLC (1) will be posted. Namely, the EP will prepare the translations $TR^{(j)}$, $1 \leq j \leq K$, and post commitments to all the numbers involved in the translations. Along the lines of the computation of Fiat-Shamir signatures [5], a hash function H will be applied to the concatenation string of all those commitment values.

The EP extracts from the hash value $H(COM(TR^{(1)})\| \ldots \|COM(TR^{(K)}))$ the random challenges used in the verification of the correctness of Aspects 0–8. He then de-commits all the values requested in the challenges and posts the values. Anyone can then verify the correctness of the computation by re-committing the exposed values and by performing additions and multiplications mod p on the exposed values and checking equalities.

Another approach to the creation of the challenges will be for the EP first to post the committed-to translations. After the posting, each of the bidders $P_1, \ldots, P_n$ sends to the EP an encrypted random string $EN(S_1), \ldots, EN(S_n)$. These encryptions are posted by the EP. After that posting the strings $S_1, \ldots, S_n$ are revealed and $S = S_1$ XOR $\ldots$ XOR S_n will define the random challenges used in the verification. From here on the process proceeds as above. The method of Time Lapse Cryptography [11] is used to force opening of all the encrypted strings S_i. A detailed protocol deals with the possibility that not all bidders P_i will submit encrypted strings. Alternatively, $P_1, \ldots, P_n$ must submit the encrypted strings $EN(S_1), \ldots, EN(S_n)$ together with their bids. The revelation of the strings is then timed by the protocol to occur after the posting of the committed-to translations by the EP.

References

[1] G. Brassard, D. Chaum, and C. Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37:156–189, 1988.

[2] E. Brickell, D. Chaum, I. Damgård, and J. V. de Graaf. Gradual and verifiable release of a secret. In *Proceedings of CRYPTO'87*, volume LNCS 293, pages 156–166, 1988.

[3] J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. Full length version of extended abstract in *Proc. Crypto 2003*, available at http://eprint.iacr.org/2002/161.pdf, 2003.

[4] I. Damgård, T. Pedersen, and B. Pfitzmann. Statistical secrecy and multibit commitments. *IEEE Transactions on Information Theory*, 44(3):1143–1151, 1998.

[5] A. Fiat and A. Shamir. How to prove youself: practical solutions to identification and signature problems. In *Proceedings of CRYPTO'86*, pages 186–194, 1987.

[6] J. Kilian. A note on efficient zero-knowledge proofs and arguments. In *Proceedings of STOC'92*, pages 723–732, 1992.

[7] J. McHugh. Covert channel analysis. Chapter 8 of *Handbook for the Computer Security Certification of Trusted Systems*, NRL Technical Memorandum, available at http://chacs.nrl.navy.mil/publications/handbook/COVCHAN.pdf, 1996.

[8] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In J. Stern, editor, *Advances in Cryptology — (EUROCRYPT 1999)*, volume 1592 of *Lecture Notes in Computer Science*, pages 107–122. Springer-Verlag, 1999.

[9] D. C. Parkes, M. O. Rabin, S. M. Shieber, and C. A. Thorpe. Practical secrecy-preserving, verifiably correct and trustworthy auctions. In *Proceedings of the 8th International Conference on Electronic Commerce (ICEC)*, pages 70–81, 2006.

[10] M. O. Rabin and J. O. Shallit. Randomized algorithms in number theory. *Communications in Pure and Applied Mathematics*, 39:239–256, 1986.

[11] M. O. Rabin and C. Thorpe. Time-lapse cryptography. Technical Report TR-22-06, Harvard University School of Engineering and Computer Science, 2006.