BetterTimes

Privacy-assured Outsourced Multiplications for Additively Homomorphic Encryption on Finite Fields

Per Hallgren[1], Martín Ochoa[2,3], and Andrei Sabelfeld[1]

[1] Chalmers University of Technology, Sweden
[2] Technische Universität München, Germany
[3] Singapore University of Technology and Design, Singapore

Abstract. We present a privacy-assured multiplication protocol using which an arbitrary arithmetic formula with inputs from two parties over a finite field $\mathbb{F}_p$ can be jointly computed on encrypted data using an additively homomorphic encryption scheme. Our protocol is secure against malicious adversaries. To motivate and illustrate applications of this technique, we demonstrate an attack on a class of known protocols showing how to compromise location privacy of honest users by manipulating messages in protocols with additively homomorphic encryption. We evaluate our approach using a prototypical implementation. The results show that the added overhead of our approach is small compared to insecure outsourced multiplication.

1 Introduction

There has been an increase of the public awareness about the importance of privacy. This has become obvious with cases such as Snowden [37] and the Tor project [11]. Unfortunately, the current practice is not yet to address privacy concerns by design [7, 36, 27, 32]. It is by far more common that the end consumer has to send privacy-sensitive information to service providers in order to achieve a certain functionality, rather than the service using privacy-preserving technologies. A major challenge of today's research community is to enable services to address privacy without hampering sought functionality and efficiency.

Recent years have brought much attention to secure computations distributed among several participants, a subfield of cryptography generally known as *Secure Multi-party Computation* (SMC). SMC has in recent years been brought to the brink of being widely applicable to real world scenarios [4, 3], although general purpose solutions with strong security guarantees are still too slow to be widely applied in practice.

This paper proposes a novel and efficient approach to jointly compute an arbitrary arithmetic formula using certain additively homomorphic encryption schemes, while maintaining security against malicious adversaries. The solution is shown to be valuable as a vital complement to boost the security of a class of privacy-preserving protocols [12, 34, 19, 33, 38], where *Alice* queries *Bob* for a

function over their combined inputs (see Figure 2). In such scenarios, it is common that *Bob* is intended to learn nothing at all, while still providing *Alice* with useful information such as whether a picture of a face matches a database [33, 12] or whether two principals are close to each other [19, 34, 38]. This work allows such solutions to harden the attacker model from *honest-but-curious* to *malicious* attackers that do not necessarily follow the protocol (both attacker models are standard in SMC and are presented for instance in [17, 28]).

Although some connections have been identified [30, 34, 19], the two communities of Privacy-preserving Services and Secure Multi-Party Computations are still largely separated. One of the goals of this paper is to contribute to bridging the gap, in particular when it comes to rigorously improving the security of efficient protocols using additively homomorphic encryption in the presence of honest-but-curious adversaries, enabling them to also protect against malicious adversaries in an efficient manner.

Problem statement In general in secure two-party computation [28] one considers the case where two parties, *Alice* with inputs $\overrightarrow{x}$ and *Bob* with inputs $\overrightarrow{y}$, want to compute a functionality $f(\overrightarrow{x}, \overrightarrow{y}) = (g(\overrightarrow{x}, \overrightarrow{b}), h(\overrightarrow{x}, \overrightarrow{y}))$, where the procedure f yields a tuple in which *Alice*'s output is the first item and *Bob*'s output is the second item. For the scope of this work, h is always the empty string, and the inputs of both parties are in $\mathbb{F}_p$, such that $\forall x_i \in \overrightarrow{x} : x_i \in \mathbb{F}_p$ and $\forall y_i \in \overrightarrow{y} : y_i \in \mathbb{F}_p$. That is, *Alice* obtains the result of g whereas *Bob* observes nothing (as usual when using partial or full homomorphic encryption). For this reason, in the following we will refer only to $g(\overrightarrow{x}, \overrightarrow{y})$ as the functionality.

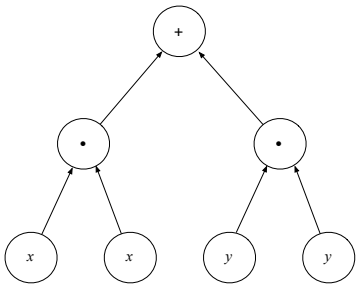

Fig. 1. Arithmetic formula computing $x^2 + y^2$.

Moreover, we set $g(\overrightarrow{x}, \overrightarrow{y})$ to be an arbitrary arithmetic formula over $\overrightarrow{x}$ and $\overrightarrow{y}$ in the operations $(\cdot, +)$ of $\mathbb{F}_p$, that is an arithmetic circuit [35] that is also a directed tree, as the one depicted in Figure 1.

We assume as usual that both *Alice* and *Bob* want *privacy* of their inputs, as much as it is allowed by g. *Bob* is willing to reveal the final output of g, but not any intermediate results, or a different function g' that would compromise the

privacy of his inputs. More precisely, we want a secure two-party computation in the malicious adversary model for a malicious *Alice* [28], as depicted in Figure 2.

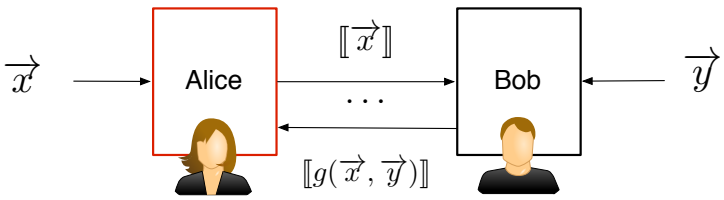

Fig. 2. High-level view of a 2-party computation based on homomorphic encryption, where $[\![\cdot]\!]$ denotes encryption under the public key of *Alice*.

Note that additions in the formula can be done correctly by *Bob* without the help of *Alice* when using an additively homomorphic encryption scheme. This holds also for all multiplications involving *Bob*'s input only, and multiplications with a ciphertext and a value known to *Bob*. The only operations outside of the scope of the additively homomorphic capabilities are multiplications involving inputs from *Alice* only. For instance in Figure 1, *Bob* can not compute x^2 (assuming x is a private input to *Alice*). In this work therefore we focus on a protocol such that *Bob* can outsource such multiplications to *Alice* without disclosing the value of the operands, and such that if *Alice* does not cooperate, the final value of the arithmetic formula is corrupted and useless to her. This will allow us to show that our protocol is fully privacy-preserving in the malicious adversary model of SMC.

Fairness of the computation (that is, all parties receive their intended output) is out of scope for two reasons: it is impossible to guarantee this property for two-party computations in the malicious model [28], but more interestingly, note that since by construction *Bob* is allowed to observe nothing, an early abortion of the protocol by *Alice* will only hamper fairness for herself.

Contributions The paper outlines a novel protocol *BetterTimes* which lets *Bob* outsource multiplications using an additively homomorphic encryption scheme (where he does not hold the private key) while asserting privacy of his inputs. *BetterTimes* provides *Bob* not only with the encrypted product but also the encryption of an assurance value (a field element $a \in \mathbb{F}_p$) which is a random value in $\mathbb{F}_p^*$ if *Alice* does not follow the protocol and an encryption of 0 otherwise. The assurance is added to the final output of g thus making the result useless to *Alice* if she tries to cheat. Our contribution thus brings the state-of-the art forward by efficiently giving *Bob* guarantees in the case that *Alice* is malicious.

We illustrate the usefulness of our approach for a class of protocols from the literature [12, 34, 19, 33, 38], which compute whether the distance between two vectors in the plane is less than a threshold. In the presence of malicious adver-

saries, leakage of private information is possible. A solution using our technique is presented for these protocols. Moreover, we make our implementation fully available to the community[4].

Relation to Zero Knowledge Proofs An alternative solution to the presented problem would be to use a Zero Knowledge schema such that *Bob* can verify that a ciphertext corresponds to a certain multiplication. Such a schema is guaranteed to exist given the general theorem of Goldreich et al. [18]. However, to the best of our knowledge it is not straightforward to constructively devise such a scheme for a given additively homomorphic cryptosystem. Our solution in contrast does not require *Bob* to be able to verify whether a multiplication is correct, but by construction will render the final computation result useless to malicious adversaries.

In a nutshell, the novelty as compared to zero-knowledge proofs is based on the simple realization that *Bob* does not need to know whether *Alice* is cheating or not in order to assure the correctness of the final computation and the privacy of his inputs, which decreases the number of round-trips that such a verification step implies. This is a special case of the *conditional disclosure of secrets* introduced by Gertner et al. [16], where a secret is disclosed using SMC only if some condition is met. In our case, the condition is that $z_i = x_i \cdot y_i$ for each multiplication in the formula, and the secret is the output of g.

To the best of our knowledge, there is no previous solution to accomplish secure outsourced multiplications for additively homomorphic encryption in the malicious model without the use of zero-knowledge proofs.

Outline The paper first introduces necessary background and notation in Section 2. Following, in Section 3 the BetterTimes protocol is described, and its application to computing arbitrary arithmetic formulas is discussed. Section 4 presents the security guarantees in the malicious adversary setting. Section 5 presents benchmarks that allow one to estimate which impact the approach would have in comparison to only protecting against semi-honest adversaries. Section 6 positions this work in perspective to already published work. Finally, Section 7 summarizes the material presented in this paper. Before delving into details, a concrete application of the proposed solution is outlined in Section 1.1.

1.1 Exploits for Proximity Protocols

We illustrate the usefulness of our approach by an attack on a class of protocols from the literature [12, 34, 19, 33, 38], which compute whether the distance between two vectors in the plane is less than a threshold in a privacy-preserving manner. Popular applications of this algorithm are geometric identification and location proximity. For concreteness, this section focuses on the distance computation used in the *InnerCircle* protocol by Hallgren et al. [19]. The same attack

[4] `https://bitbucket.org/hallgrep/bettertimes`

also applies to the other representatives of the same class of protocols [12, 34, 33, 38], but in many cases a successful exploit does not have as visible effects.

Hallgren et al. present a protocol for privacy-preserving location proximity. It is based on the fact that *Bob* can compute the euclidean distances from a point represented as three ciphertexts $[\![2x]\!]$, $[\![2y]\!]$ and $[\![x^2 + y^2]\!]$ to any other point known by *Bob* using additively homomorphic encryption (here $[\![\cdot]\!]$ stands for encryption under the public key of *Alice*). A problem with the approach is that *Bob* has no knowledge of how the ciphertexts are actually related, he sees three ciphertexts $[\![\alpha]\!]$, $[\![\beta]\!]$ and $[\![\gamma]\!]$. In the case that $\gamma \neq (\alpha/2)^2 + (\beta/2)^2$, subsequent computations may leak unwanted information. The distance is expressed as the (squared) distance as shown in Equation (1), computed homomorphically as shown in Equation (2) where only some of *Bob*'s inputs are needed in plaintext.

$$D = x_A^2 + y_A^2 + x_B^2 + y_B^2 - (2x_A x_B + 2y_A y_B) \tag{1}$$

$$[\![D]\!] = [\![x_A^2 + y_A^2]\!] \oplus [\![x_B^2 + y_B^2]\!] \ominus ([\![2x_A]\!] \odot x_B \oplus [\![2y_A]\!] \odot y_B) \tag{2}$$

Here, $\oplus$, $\ominus$ and $\odot$ are the homomorphic operations which in the plaintext space map to $+$, $-$ and $\cdot$ respectively (see Section 2). Now, by replacing the information sent by *Alice* by α, β and γ and observing that *Alice* can choose α and β arbitrarily, the expression becomes as in Equation (3):

$$D = x_B^2 + y_B^2 + \gamma + \alpha x_B + \beta y_B \tag{3}$$

The effects of the attack are very illustrative in [34, 19, 38]. In these works, *Bob* wants to return a boolean $b = (r^2 > D)$ indicating whether two principals are within r from each other. Thus the result given to *Alice* is the evaluation of the function $r^2 > x_B^2 + y_B^2 + \alpha x_B + \beta y_B + \gamma$. This is equivalent to the result of $r^2 - \gamma > x_B^2 + y_B^2 + \alpha x_B + \beta y_B$. Given that *Alice* knows r, she can encode it into the manipulated variables thus forcing the evaluation of $\delta > x_B^2 + y_B^2 + \alpha x_B + \beta y_B + \eta$, with $\gamma = r^2 - \delta - \eta$. By changing α, β and η, *Alice* can move the center of the queried area, and by tweaking δ she can dictate the size of the area, causing unwanted and potentially very serious information leakage (for instance by querying in arbitrarily located and precise areas such as buildings).

Securing affected protocols Based on the novel asserted multiplication presented in Section 3, a new structure for the protocols of Hallgren et al. can be constructed. Similar amendments can easily be constructed in similar form for other afflicted solutions [12, 34, 33, 38]. Using the system proposed in this paper, it is possible to send only the encryption of x_A and y_A in the initial message, and securing the necessary squaring by means of *BetterTimes*.

An arithmetic formula which computes the distance directly using x_A, y_A, x_B and y_B is already defined in Equation (1). Now remains only to model this such that it can be computed by the system presented later in this paper, after which the protocols can proceed to compute the proximity result as they would normally.

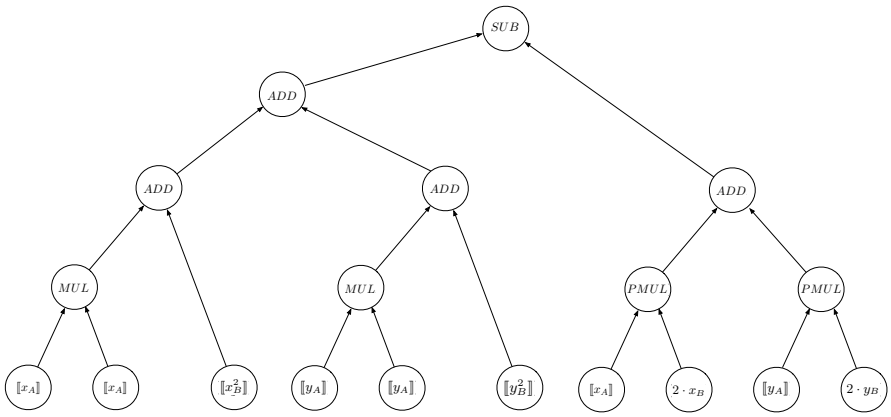

Fig. 3. Tree depicting computation of a secured version of the protocol.

The result is an algorithm modeled using the recursive data structure Ins, which simply is passed to the procedure *evaluate* by *Bob*, see Section 3 and Figure 6. The formula of can be depicted as a tree as in Figure 3. The concrete instructions (instances of Ins) are spelled out in Appendix A.

2 Background

The solution proposed in this paper makes use of any additively homomorphic encryption scheme which provides semantic security and where the plaintext space is a field (for instance such as the DGK Scheme [9]). For a definition of semantic security see [2].

Additively Homomorphic Encryption Schemes Here and henceforth, k is the private key belonging to *Alice* and K and is the corresponding public key. Let the plaintext space $\mathcal{M}$ be isomorphic to the field $(\mathbb{Z}_p, \cdot, +)$ for some prime number p and the ciphertext space $\mathcal{C}$ such that encryption using public key K is a function $E : \mathcal{M} \to \mathcal{C}$ and decryption using a private key k is $D : \mathcal{C} \to \mathcal{M}$.

The vital homomorphic features which is used later in the paper is an addition function $\oplus : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$, a unary negation function $\neg : \mathcal{C} \to \mathcal{C}$, and a multiplication function $\odot : \mathcal{C} \times \mathcal{M} \to \mathcal{C}$.

$$E(m_1) \oplus E(m_2) = E(m_1 + m_2) \tag{4}$$
$$\neg E(m_1) = E(-m_1) \tag{5}$$
$$E(m_1) \odot m_2 = E(m_1 \cdot m_2) \tag{6}$$

Note that in a finite field any non-zero element multiplied with a non-zero random element yields a non-zero uniformly distributed element. Formally:

$$E(m_1) \odot \rho = \begin{cases} E(0) \text{ if } m_1 = 0 \\ E(l) \text{ with } l \in \mathcal{M}^{\mathcal{U}} \text{ otherwise} \end{cases}, \text{ with } \rho \in \mathcal{M}^{\mathcal{U}} \tag{7}$$

where $m_1 \in \mathcal{M}$, $m_2 \in \mathcal{M}$ and $\mathcal{M}^{\mathcal{U}}$ is a uniformly random distribution of all elements in $\mathcal{M} \setminus \{0\}$.

Syntax and conventions For readability, the operations $\oplus$, $\odot$, $\neg$, E and D do not have a key associated to them, we assume they all use the usual k, K pair where *Alice* holds k. The $\ominus$ symbol is used in the following to represent addition by a negated term. That is, $c_1 \oplus \neg c_2$ is written as $c_1 \ominus c_2$. For further brevity, a ciphertext c encrypting a plaintext p under the public key of *Alice* is denoted as $[\![p]\!]$.

The protocol description in Figure 5 and Figure 6 is given in the language pWhile [1]. For the convenience of the reader a few constructs used in the paper are outlined here, but for details the reader is directed to [1]. $a \leftarrow b$ means assigning a value b to a variable a, while $a \xleftarrow{\$} [0..n]$ means assigning a random value between 0 and n to a.

Security Concepts In the following we briefly recall some fundamental concepts from SMC that will be useful for the security guarantees discussion of Sect. 4.

Definition 1 (Negligible functions). *A function $\epsilon : \mathbb{N} \to \mathbb{R}$ is said to be negligible if*

$$\forall\ c \in \mathbb{N}.\ \exists\ n_c \in \mathbb{N}.\ \forall_{n \geq n_c}\ |\epsilon(n)| \leq n^{-c}$$

That is, ϵ decreases faster than the inverse of any polynomial.

Definition 2 (Indistinguishability).

The two random variables $X(n, a)$ and $Y(n, a)$ (where n is a security parameter and a represents the inputs to the protocol) are called computationally indistinguishable and denoted $X \stackrel{c}{\equiv} Y$ if for a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ the following function is negligible:

$$\delta(n) = |\Pr[\mathcal{A}(X(n, a)) = 1] - \Pr[\mathcal{A}(Y(n, a)) = 1]|$$

3 Arithmetic formulas through assured Multiplication

As previously discussed, our goal is a system which can compute any arithmetic formula in the presence of a malicious *Alice* (who holds the private key), without leaking any information derived from *Bob*'s inputs except the result of g. To show how to reach this, we first outline the primary building block, *BetterTimes*.

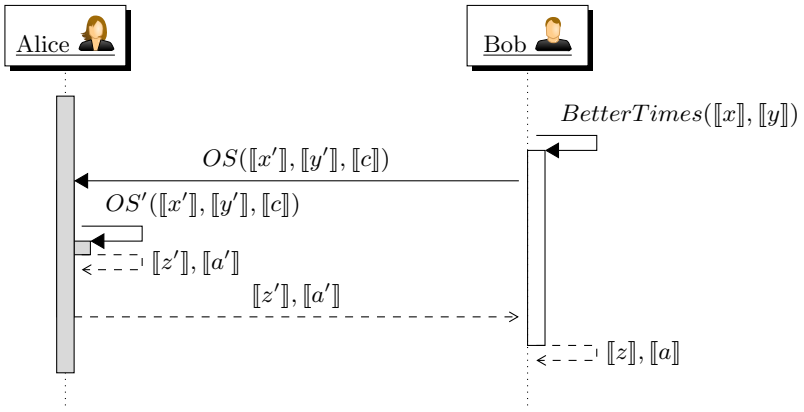

Fig. 4. Visualization of the attested multiplication protocol

3.1 Privacy-assured Outsourced Multiplication

The core of the solution is a novel outsourced multiplication protocol with privacy guarantees, *BetterTimes*. The protocol is visualized in Figure 4 and detailed in Figure 5. *BetterTimes* allows *Bob* to calculate a multiplication by outsourcing to *Alice*, while retaining an attestation value with which it is possible to make sure that *Alice* can learn no unintended information.

The principals interact once during BetterTimes, where *Bob* contacts *Alice* through the procedure OS (for outsource), defined in Figure 5. As a result of this interaction, *Bob* can compute a value $[\![z]\!]$ which corresponds to the encryption of the multiplication $x \cdot y$ if *Alice* is honest and an attestation value a which will be uniformly random if *Alice* does not comply with the protocol. *Alice* can only deviate from the protocol by using $OS' \neq OS$.

BetterTimes contains several random variables, here follows a brief explanation of their names to make the procedures easier to follow. The first two, c_a and c_m, serve to construct the challenge c used in the attestation. c_a and c_m are an additive and multiplicative component, respectively. The second pair, b_x and b_y, are used to blind the operands x and y, respectively, when outsourcing the multiplication. Finally ρ is used to make sure that an attestation which doesn't match the supplied product causes a random offset of the final result.

Note that the attestation is only needed when outsourcing a multiplication. The blinding used in BetterTimes has also been presented and used by, among others, Kolesnikov et al. [22]. The construction using the challenges c_a and c_m yield the following computations in the plaintext, starting with the attestation value a in Equation (8). Through the procedure OS, *Alice* replies with (in the plaintexts) as in Equation (9). Thus, assuming *Alice* is honest, we see that Equation (10) must hold.

Proc. $BetterTimes([\![x]\!], [\![y]\!])$:
$c_a \xleftarrow{\$} \{0..p\}; c_m \xleftarrow{\$} \{1..p\};$
$b_x \xleftarrow{\$} \{0..p\}; b_y \xleftarrow{\$} \{0..p\};$
$\rho \xleftarrow{\$} \{1..p\};$
// Blind operands
$[\![x']\!] \leftarrow [\![x]\!] \oplus [\![b_x]\!]; [\![y']\!] \leftarrow [\![y]\!] \oplus [\![b_y]\!];$
// Create challenge
$[\![c]\!] \leftarrow ([\![x']\!] \oplus [\![c_a]\!]) \odot c_m;$
// Outsource multiplication
$([\![z']\!], [\![a']\!]) \leftarrow OS([\![x']\!], [\![y']\!], [\![c]\!]);$
// Compute assurance value
$[\![a]\!] \leftarrow ([\![a']\!] \ominus [\![z']\!] \odot c_m \ominus [\![y']\!] \odot (c_a \cdot c_m)) \odot \rho;$
// Un-blind multiplication
$[\![z]\!] \leftarrow [\![z']\!] \ominus ([\![x']\!] \odot b_y \oplus [\![y']\!] \odot b_x \oplus [\![b_x \cdot b_y]\!]);$
return $([\![a]\!], [\![z]\!])$;

Proc. $OS([\![x]\!], [\![y]\!], [\![c]\!])$:
return $((E(D([\![x]\!]) \cdot D([\![y]\!]),$
$E(D([\![c]\!]) \cdot D([\![y]\!]))))$;

Fig. 5. The attested multiplication protocol

$$a = (a' - z' \cdot c_m - y' \cdot c_a \cdot c_m) \cdot \rho = (a' - z' - y' \cdot c_a) \cdot c_m \cdot \rho \tag{8}$$

$$a' = ((x' + c_a) \cdot c_m) \cdot y' = (x' \cdot y' + y' \cdot c_a) \cdot c_m \tag{9}$$

$$a = (x' \cdot y' - z') \cdot c_m \cdot \rho \tag{10}$$

Since by assumption *Alice* is honest, $z' = x' \cdot y' \implies a = 0$. To see that this is the case if and only if *Alice* honest, see Section 4.

3.2 Privacy-assured Arithmetic Formulas

Using *BetterTimes* as described above, the following discusses how to construct arbitrary arithmetic formulas. The general idea is to accumulate any errors caused by misbehavior by *Alice* using attestations a_j, one for each outsourced multiplication. The other operations require no attestations as they can be calculated locally by *Bob*. If *Alice* is dishonest during an outsourced multiplication, the corresponding attestation a_j is a uniformly random variable. Once an arithmetic formula has been fully evaluated, and the result obtained as $[\![result]\!]$, *Bob* instead returns the value $[\![result]\!] \oplus \sum a_i$. The returned value is $[\![result]\!]$ if and only if *Alice* is honest, and the encryption of a uniformly random field element if she is dishonest.

Given an arbitrary arithmetic formula g, the system is designed using a recursive data structure **Ins**, modeling an *instruction* representing g. An instruction either contains an operation and two operands or a scalar. Formally, $\mathbf{Ins} \in \{[o, l, r] | x\}$, where o is the operator, l and r are the left- and right-hand side operands, and x is a scalar. The operands are nested instances of **Ins**. The operator is an enum-like variable, with four possible values $\{ADD, SUB, MUL, PMUL\}$. The scalar member holds a ciphertext or a plaintext. An instance *ins*

```
Proc. binOp(ins) :
if isScalar(ins) then :
    return ([[0]], ins) ;
else :
    (a_1, x) ← binOp(ins[1]);
    (a_2, y) ← binOp(ins[2]);
    switch(ins[0]) :
        case ADD :
            return (a_1 ⊕ a_2, x ⊕ y) ;
        case SUB :
            return (a_1 ⊕ a_2, x ⊖ y) ;
        case PMUL :
            return (a_1 ⊕ a_2, x ⊙ y) ;
        case MUL :
            (a_3, z) ← BetterTimes(x, y)
            return (a_1 ⊕ a_2 ⊕ a_3, z) ;
```

```
Proc. evaluate(alg) :
(a, result) ← binOp(alg);
return result ⊕ a;
```

Fig. 6. The procedures to evaluate recursive instructions.

of **Ins** is created using either $Ins(scalar)$, or $Ins(op, ins1, ins2)$. An instruction to compute the addition of two encrypted values $[\![x]\!]$ and $[\![y]\!]$ thus looks like as e.g.: $Ins(ADD, Ins([\![x]\!]), Ins([\![y]\!]))$. At the start of the protocol, *Bob* must collect *Alice*'s encrypted inputs, and hard-wire them into the algorithm. For an example, see Appendix A.

The core of the setup is the recursive procedure *binOp*, defined in Figure 6, which recursively computes an instruction including any nested instructions. The *binOp* return value has the same structure as that of *BetterTimes*, but the attestation in the first part of the return value is now an accumulated value over all nested instructions.

The main function, wrapping all functionality, is the *evaluate* procedure, see Figure 6. It takes as parameter an algorithm, which is modeled using an instruction with nested instructions. Evaluate adds the attestation values and the result of the instructions, creating the final result – which is the output of g if and only if *Alice* is honest. For a visualization of messages exchanged and actions taken by each principal, see Appendix B.

4 Security guarantees

The goal of this section is to show that the result of *evaluate* as defined above is secure in the malicious adversary model for *Alice* (as depicted in Figure 2), following standard SMC security definitions. We have already introduced the fundamental notion of computational indistinguishability in Sect. 2.

Malicious adversary Recall that a malicious *Alice* in possession of the private key can attack the privacy of the inputs of *Bob* by deviating from the original protocol (as discussed in Section 1.1 for a proximity calculation protocol).

Intuitively, a malicious *Alice* will deviate from the protocol every time it fails to answer to the outsourced multiplication with the expected values z' and a' as defined in Figure 5. A deviation would be for example failing to multiply x' with y', in order to change the intended jointly computed arithmetic formula. Formally, we set out to prove the following theorem, which is an instance of the general definition of [28] where the concrete SMC protocol π will depend on the arithmetic formula g to be jointly computed. In the following indistinguishability will be established with respect to the size p of the field $\mathbb{F}_p$ (p is thus the security parameter).

Theorem 1. *For a fixed but arbitrary arithmetic formula $g(\overrightarrow{x}, \overrightarrow{y})$ represented by a recursive instruction $\iota \in \boldsymbol{Ins}$, for every adversary $\mathcal{A}$ against the protocol π resulting from evaluate(ι), there exist a simulator $\mathcal{S}$ such that:*

$$\{\text{IDEAL}_{g,\mathcal{S}(s)}(\overrightarrow{x}, \overrightarrow{y})\} \overset{c}{\equiv} \{\text{REAL}_{\pi,\mathcal{A}(s)}(\overrightarrow{x}, \overrightarrow{y})\}$$

where $\overset{c}{\equiv}$ denotes computational indistinguishability of distributions.

Here the IDEAL function gives the distribution of the output of a simulator $\mathcal{S}$ that interacts with an idealized implementation of the functionality g on behalf of *Alice*, where both parties give their inputs to a trusted third party that computes g and gives it back to $\mathcal{S}$. Recall that in our setting *Bob* receives no output from the ideal functionality. Therefore, it does not make sense for the adversary to abort the protocol. Also, this means that fairness guarantees for *Bob* are out of scope, so we do not account for abortions of the protocol by the simulator.

On the other hand REAL stands for the distribution of the output of a real adversary $\mathcal{A}$ against concrete executions of the protocol π. The parameter s stands for extra information known to the attacker, in this case we assume that the adversary knows the abstract arithmetic formula g and therefore knows how many multiplications it contains.

Before proceeding with the proof, we introduce the following Lemma.

Lemma 1. *In the outsourced multiplication protocol BetterTimes the attestation value* **a** *is equal to 0 if the protocol is followed, and is indistinguishable from a randomly distributed non-zero element otherwise.*

Proof. First recall the calculations from Figure 5:

$$[\![a]\!] \leftarrow ([\![a']\!] \ominus [\![z']\!] \odot c_m \ominus [\![y']\!] \odot (c_a \cdot c_m)) \odot \rho; \tag{11}$$
$$[\![z]\!] \leftarrow [\![z']\!] \ominus ([\![x']\!] \odot b_y \oplus [\![y']\!] \odot b_x \oplus [\![b_x \cdot b_y]\!]) \tag{12}$$

Which in the plaintexts corresponds to:

$$a = (a' - (z' + y' \cdot c_a) \cdot c_m) \cdot \rho \tag{13}$$
$$z = z' - (x' \cdot b_y + y' \cdot b_x + b_x \cdot b_y) \tag{14}$$

where a' and z' are produced by *Alice*. It is easy to see that if a' and z' are computed following the protocol, then $a = 0$ by construction.

To see that if *Alice* does not comply with the protocol then a is a randomly distributed non-zero element with very high probability, first note that there are three cases for non-compliance, either $z' \neq x' \cdot y'$, $a' \neq y' \cdot c$ or both. In any case of non-compliance, the goal of *Alice* is to construct a' and z' such that:

$$a' - (z' + y' \cdot c_a) \cdot c_m = 0$$

since otherwise by construction a will be random. Then it must hold:

$$a' = (z' + y' \cdot c_a) \cdot c_m$$

Note that given $(x' + c_a) \cdot c_m$ (which is known by *Alice*), the probability of guessing c_m is at most $\epsilon = \frac{1}{2^p}$ where p is the size of the field, since multiplication is a random permutation and c_a is unknown and uniformly distributed.

Now by contradiction, lets assume that the probability of *Alice* of computing $a' = (z' + y' \cdot c_a) \cdot c_m$ with $z' \neq x' \cdot y'$ is bigger than ϵ. If this holds, then she can also compute:

$$\alpha = a' - (x' + c_a) \cdot c_m \cdot y' = (z' - x' \cdot y') \cdot c_m$$

But then she could also compute $c_m = \alpha(z' - x' \cdot y')^{-1}$ with probability bigger than ϵ, since by hypothesis $z' \neq x' \cdot y'$ and thus $(z' - x' \cdot y') \in \mathbb{F}_p^*$ is invertible, which contradicts the fact that the probability of guessing c_m is smaller than ϵ. □

Now, for the proof of Theorem 1:

Proof (Theorem 1). Without loss of generality, we assume that $\iota \in \mathbf{Ins}$ has m instructions of type MUL. We will distinguish two cases.

$\mathcal{A}$ *follows the protocol* It is easy to see that all m intermediate messages sent from *Bob* appear uniformly random to *Alice* (and independent) due to the fact that they are all of the type $r_i = (x', y', c)$ where each value is blinded. In the case when $\mathcal{A}$ complies with the protocol, the last message contains the correct output g, since *Bob* is an honest party. This implies that the output of $\mathcal{A}$ depends exclusively on $r_0, \ldots, r_m$ uniformly distributed triples and $g(\overrightarrow{x}, \overrightarrow{y})$, so we can simulate an adversary as:

$$\mathcal{S} := \mathcal{A}(r_0, \ldots, r_m, g(\overrightarrow{x}, \overrightarrow{y}))$$

$\mathcal{A}$ *does not follow the protocol* Note that independently of the cheating strategy of $\mathcal{A}$, all m intermediate messages sent from *Bob* appear uniformly random to $\mathcal{A}$ since the blinding is done by *Bob* locally with randomization independent from $\mathcal{A}$'s inputs. Now, as a consequence of Lemma 1, if $\mathcal{A}$ does not follow the protocol for at least one of the outsourced multiplications, the final message will be blinded by the accumulated attestation value, which is indistinguishable from random. Therefore, the last message will contained the encryption of a random value, denoted r_{m+1}. Therefore we can simulate this in the ideal model as:

$$\mathcal{S} := \mathcal{A}(r_0, \ldots, r_m, r_{m+1})$$

for pairwise independent and random variables r_i. □

Rings Note that additively homomorphic schemes are commonly defined over groups where when multiplying a non zero element γ with a uniformly chosen ρ, the result is not necessarily uniformly distributed, thus potentially affecting the blinding of $g(x, y)$. For instance, in groups such as $\mathbb{Z}_n$ for composite $n = p \cdot q$ (as used by the Paillier [31] encryption scheme) when multiplying a non invertible element with random ρ, the result stays in the subgroup of non-invertible elements. In that setting is thus possible to show a counterexample to the theorem above, which motivates our restriction to constructions over fields.

5 Evaluation

The approach has been implemented in Python using the GMP [13] arithmetic library. The implementation as been benchmarked to show the impact of using our approach compared to the more common approach of naive outsourced multiplications. In the naive approach, *Alice* is honest-but-curious, and the operands are therefore only blinded. For this implementation, the DGK [9] cryptosystem was used.

Table 1. Benchmarks for outsourced multiplication

Plaintext space	Time (in milliseconds)					
	1024 bits			2048 bits		
	This approach	Naive approach	Extra work	This approach	Naive approach	Extra work
2^2	6.286	4.016	56.52%	29.686	19.458	52.56%
2^8	6.400	4.017	59.32%	30.052	19.484	54.24%
2^{16}	6.432	4.148	55.06%	30.188	19.574	54.22%
2^{24}	6.538	4.100	59.46%	30.578	19.801	54.43%

Table 1 shows time in milliseconds for different sizes of plaintexts and keys for the two cases when outsourced multiplication is performed using BetterTimes, or naively. The difference between the two approaches is a small factor of about 1.5 for both key sizes, though slightly smaller for the larger keys. The factor is only marginally increasing as the plaintext space grows from 2^2 to 2^{24}.

The benchmarked time shows only the processing time for each multiplication, the communication overhead is exactly twice for our approach as compared to the naive solution.

6 Related Work

There are three current approaches to compute an arbitrary formula in the two-party setting in the presence of malicious adversaries, Fully Homomorphic Encryption, Enhanced Garbled Circuits and Zero-knowledge proofs.

FHE is by far the most inefficient approach, and its use is often considered not feasible due to the heavy resource consumption. We do not consider FHE a viable alternative to additively homomorphic encryption for practical applications. Garbled Circuits is an excellent tool for boolean circuits, but has been found to not perform as well for arithmetic circuits as approaches built on homomorphic encryption. Zero-knowledge proofs could be used instead of the proposed approach, but at the cost of more computations and/or round trips.

Zero-knowledge Proofs The technique which most resembles *BetterTimes* is that of *Zero-Knowledge* (ZK) proofs. Any statement in NP can be proven using generic, though inefficient, ZK (Goldreich et al. [18]). However, to the best of our knowledge, there is no ad-hoc proof for correct multiplications that directly applies to the setting of additively homomorphic encryption without significantly more overhead than the proposed approach, by e.g. introducing more round trips.

Some protocols in the literature can be used efficiently for proving correct multiplications, with only one additional round trip. One such is the Chaum-Pedersen protocol [6], which however is not trivially applicable to an arbitrary encryption scheme. Another interesting solution was introduced by Damgård and Jurik [10], but which is constructed specifically for the Damgård-Jurik cryptosystem.

Secure Multi-party Computations There are two main categories for private remote computations: Homomorphic Encryption and Garbled Circuits. Through recent research they are both near practical applicability (see [24, 20, 25] and [5, 20, 15]). However, which of the two approaches to choose is typically application-dependent [26, 23]. Our approach brings state-of-the-art SMC solutions based on additively homomorphic cryptographic systems forward by protecting against malicious adversaries when outsourcing multiplications, while remaining strongly competitive to the efficient though less secure approaches which currently are popular examples.

There are several works that combine the use of an additively homomorphic scheme with secret sharing, to compute multiplications securely using threshold encryption. This line of work stems from the SMC schemes developed by Cramer et al. [8]. Note that such approaches are secure only against malicious *minorities*, and are not directly applicable in scenarios with only two parties.

To compare against GC-solutions which can compute arbitrary formulas, some experiments using FastGC, a Garbled Circuit framework by Huang et al. [20] were conducted. Any arithmetic circuit can be expressed as a binary circuit, and vice versa[14]. In this framework for arbitrary computations, integer multiplication of 24-bit numbers needed 332 ms to finish, approximately 5078%

slower than BetterTimes. Note however that FastGC is only secure in the honest-but-curious model, and thus not as secure as the approach presented in this paper. Further work exists in the direction of efficiently providing security against malicious adversaries by the authors of FastGC [21], however where one bit of the input is leaked. Moreover, work on optimizing garbled-circuits in the honest-but-curious model also exists, e.g. recently [29], but so far without enough speedup that it can compare to additively homomorphic encryption for privately computing arithmetic formulas.

7 Conclusions

We have presented a protocol for outsourcing multiplications and have shown how to use it construct a system for computation of arbitrary arithmetic formulas with strong privacy guarantees. We have shown that the construction is secure in the malicious adversary model and that the overhead of using the approach is a small constant factor.

The need for such a protocol is justified by the format attacks we have unveiled in known protocols, and presented a concrete exploit targeting [38] where we can alter the format of a message and gain more than the intended amount of location information. We have made a case for using a more realistic attacker model and identified examples from the literature which are vulnerable to this stronger attacker, while also showing how to amend such vulnerabilities. Moreover, we make our implementation fully available to the community.

As future work we plan to investigate the non-trivial task of applying closely related primitives (such as Zero-Knowledge constructions [6] and Threshold Encryption [8]) to achieve the same security guarantees, and benchmark those solutions to compare them to *BetterTimes*.

Acknowledgments Thanks are due to Allen Au for the useful comments. This work was funded by the European Community under the ProSecuToR project and the Swedish research agencies SSF and VR.

References

1. G. Barthe, B. Grégoire, and S. Z. Béguelin. Formal certification of code-based cryptographic proofs. In Z. Shao and B. C. Pierce, editors, *Proceedings of the 36th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2009, Savannah, GA, USA, January 21-23, 2009*, pages 90–101. ACM, 2009.
2. M. Bellare, A. Boldyreva, and S. Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In B. Preneel, editor, *EUROCRYPT*, volume 1807 of *Lecture Notes in Computer Science*, pages 259–274. Springer, 2000.
3. D. Bogdanov, S. Laur, and J. Willemson. Sharemind: A framework for fast privacy-preserving computations. In S. Jajodia and J. López, editors, *Computer Security - ESORICS 2008, 13th European Symposium on Research in Computer Security, Málaga, Spain, October 6-8, 2008. Proceedings*, volume 5283 of *Lecture Notes in Computer Science*, pages 192–206. Springer, 2008.

4. P. Bogetoft, D. L. Christensen, I. Damgård, M. Geisler, T. P. Jakobsen, M. Krøigaard, J. D. Nielsen, J. B. Nielsen, K. Nielsen, J. Pagter, M. I. Schwartzbach, and T. Toft. Secure multiparty computation goes live. In R. Dingledine and P. Golle, editors, *Financial Cryptography*, volume 5628 of *Lecture Notes in Computer Science*, pages 325–343. Springer, 2009.
5. Z. Brakerski, C. Gentry, and V. Vaikuntanathan. Fully homomorphic encryption without bootstrapping. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:111, 2011.
6. D. Chaum and T. P. Pedersen. Wallet databases with observers. In E. F. Brickell, editor, *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105. Springer, 1992.
7. D. Coldewey. "Girls Around Me" Creeper App Just Might Get People To Pay Attention To Privacy Settings. TechCrunch, March 2012.
8. R. Cramer, I. Damgård, and J. B. Nielsen. Multiparty computation from threshold homomorphic encryption. In B. Pfitzmann, editor, *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding*, volume 2045 of *Lecture Notes in Computer Science*, pages 280–299. Springer, 2001.
9. I. Damgård, M. Geisler, and M. Krøigaard. Efficient and secure comparison for online auctions. In J. Pieprzyk, H. Ghodosi, and E. Dawson, editors, *ACISP*, volume 4586 of *Lecture Notes in Computer Science*, pages 416–430. Springer, 2007.
10. I. Damgård and M. Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In K. Kim, editor, *Public Key Cryptography, 4th International Workshop on Practice and Theory in Public Key Cryptography, PKC 2001, Cheju Island, Korea, February 13-15, 2001, Proceedings*, volume 1992 of *Lecture Notes in Computer Science*, pages 119–136. Springer, 2001.
11. R. Dingledine, N. Mathewson, and P. F. Syverson. Tor: The second-generation onion router. In *USENIX Security Symposium*, pages 303–320. USENIX, 2004.
12. Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft. Privacy-preserving face recognition. In I. Goldberg and M. J. Atallah, editors, *Privacy Enhancing Technologies*, volume 5672 of *Lecture Notes in Computer Science*, pages 235–253. Springer, 2009.
13. Free Software Foundation. The gnu multiple precision arithmetic library. `http://gmplib.org/`, 1991-2013.
14. C. Gentry. Fully homomorphic encryption using ideal lattices. In M. Mitzenmacher, editor, *STOC*, pages 169–178. ACM, 2009.
15. C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In R. Canetti and J. A. Garay, editors, *CRYPTO (1)*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92. Springer, 2013.
16. Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. Protecting data privacy in private information retrieval schemes. *J. Comput. Syst. Sci.*, 60(3):592–629, 2000.
17. O. Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004.
18. O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems. *J. ACM*, 38(3):690–728, July 1991.
19. P. Hallgren, M. Ochoa, and A. Sabelfeld. InnerCircle: A Parallelizable Decentralized Privacy-Preserving Location Proximity Protocol. In *International Conference on Privacy, Security and Trust (PST)*, July 2015.

20. Y. Huang, D. Evans, J. Katz, and L. Malka. Faster secure two-party computation using garbled circuits. In *USENIX Security Symposium*. USENIX Association, 2011.
21. Y. Huang, J. Katz, and D. Evans. Quid-pro-quo-tocols: Strengthening semi-honest protocols with dual execution. In *IEEE Symposium on Security and Privacy, SP 2012, 21-23 May 2012, San Francisco, California, USA*, pages 272–284. IEEE Computer Society, 2012.
22. V. Kolesnikov, A.-R. Sadeghi, and T. Schneider. From dust to dawn: Practically efficient two-party secure function evaluation protocols and their modular design. *IACR Cryptology ePrint Archive*, 2010:79, 2010.
23. V. Kolesnikov, A.-R. Sadeghi, and T. Schneider. A systematic approach to practically efficient general two-party secure function evaluation protocols and their modular design. *Journal of Computer Security*, 21(2):283–315, 2013.
24. V. Kolesnikov and T. Schneider. Improved garbled circuit: Free xor gates and applications. In L. Aceto, I. Damgård, L. A. Goldberg, M. M. Halldórsson, A. Ingólfsdóttir, and I. Walukiewicz, editors, *ICALP (2)*, volume 5126 of *Lecture Notes in Computer Science*, pages 486–498. Springer, 2008.
25. B. Kreuter, A. Shelat, and C. Shen. Billion-gate secure computation with malicious adversaries. In T. Kohno, editor, *Proceedings of the 21th USENIX Security Symposium, Bellevue, WA, USA, August 8-10, 2012*, pages 285–300. USENIX Association, 2012.
26. R. L. Lagendijk, Z. Erkin, and M. Barni. Encrypted signal processing for privacy protection: Conveying the utility of homomorphic encryption and multiparty computation. *IEEE Signal Process. Mag.*, 30(1):82–105, 2013.
27. M. Li, H. Zhu, Z. Gao, S. Chen, L. Yu, S. Hu, and K. Ren. All your location are belong to us: breaking mobile social networks for automated user location tracking. In *MobiHoc*, pages 43–52, 2014.
28. Y. Lindell and B. Pinkas. Secure multiparty computation for privacy-preserving data mining. *IACR Cryptology ePrint Archive*, 2008:197, 2008.
29. C. Liu, X. S. Wang, K. Nayak, Y. Huang, and E. Shi. Oblivm: A programming framework for secure computation. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 359–376. IEEE Computer Society, 2015.
30. A. Narayanan, N. Thiagarajan, M. Lakhani, M. Hamburg, and D. Boneh. Location privacy via private proximity testing. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2011, San Diego, California, USA, 6th February - 9th February 2011*. The Internet Society, 2011.
31. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In J. Stern, editor, *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 1999.
32. I. Polakis, G. Argyros, T. Petsios, S. Sivakorn, and A. D. Keromytis. Where's Wally? Precise User Discovery Attacks in Location Proximity Services. In *ACM Conference on Computer and Communications Security*, Oct. 2015.
33. A. Sadeghi, T. Schneider, and I. Wehrenberg. Efficient privacy-preserving face recognition. In D. Lee and S. Hong, editors, *Information, Security and Cryptology - ICISC 2009, 12th International Conference, Seoul, Korea, December 2-4, 2009, Revised Selected Papers*, volume 5984 of *Lecture Notes in Computer Science*, pages 229–244. Springer, 2009.

34. J. Sedenka and P. Gasti. Privacy-preserving distance computation and proximity testing on earth, done right. In S. Moriai, T. Jaeger, and K. Sakurai, editors, *9th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '14, Kyoto, Japan - June 03 - 06, 2014*, pages 99–110. ACM, 2014.
35. A. Shpilka and A. Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5(3-4):207–388, 2010.
36. M. Veytsman. How I was able to track the location of any Tinder user, February 2014. Web resource: http://blog.includesecurity.com/.
37. M. Wachs, M. Schanzenbach, and C. Grothoff. On the feasibility of a censorship resistant decentralized name system. In J. L. Danger, M. Debbabi, J. Marion, J. García-Alfaro, and A. N. Zincir-Heywood, editors, *Foundations and Practice of Security - 6th International Symposium, FPS 2013, La Rochelle, France, October 21-22, 2013, Revised Selected Papers*, volume 8352 of *Lecture Notes in Computer Science*, pages 19–30. Springer, 2013.
38. G. Zhong, I. Goldberg, and U. Hengartner. Louis, lester and pierre: Three protocols for location privacy. In N. Borisov and P. Golle, editors, *Privacy Enhancing Technologies, 7th International Symposium, PET 2007 Ottawa, Canada, June 20-22, 2007, Revised Selected Papers*, volume 4776 of *Lecture Notes in Computer Science*, pages 62–76. Springer, 2007.

A A concrete instantiation to secure Hallgren et al.

To make the protocol from Hallgren et al., and other afflicted solutions, secure against format attacks from *Alice*, the distance can be computed directly on the coordinates instead of using several correlated values. The secured algorithm could be modeled as follows:

$$
\begin{aligned}
&Ins(SUB,\\
&\quad Ins(ADD,\\
&\qquad Ins(ADD, Ins(MUL, Ins([\![x_A]\!]), Ins([\![x_A]\!])), Ins([\![x_B^2]\!])),\\
&\qquad Ins(ADD, Ins(MUL, Ins([\![y_A]\!]), Ins([\![y_A]\!])), Ins([\![y_B^2]\!]))\\
&\quad),\\
&\quad Ins(ADD,\\
&\qquad Ins(PMUL, Ins([\![x_A]\!]), Ins(2 \cdot x_B)),\\
&\qquad Ins(PMUL, Ins([\![y_A]\!]), Ins(2 \cdot y_B))\\
&\quad),\\
&)
\end{aligned}
$$

B Visualization of privacy-preserving arithmetic formula

Figure 7 depicts the system for privacy-preserving arithmetic formulas presented in this paper, during an execution where *Alice* is honest. *Alice* is the initiating party, and starts by sending her inputs to *Bob*. *Bob* then hardwires both is and *Alice*'s inputs into a instruction of nested operations, forming a tree like in Figure 3. Depending on g, *Bob* computes any local operations and executes

BetterTimes as necessary, with as many iterations as necessary. Finally, he computes the ciphertext $[\![result]\!]$. Since *Alice* by assumption is honest, $[\![result]\!]$ will hold the output of g (and would hold the encryption of a random element in $\mathbb{F}_p$ if *Alice* was dishonest). BetterTimes is simplified here, for a more complete visualization see Figure 4.

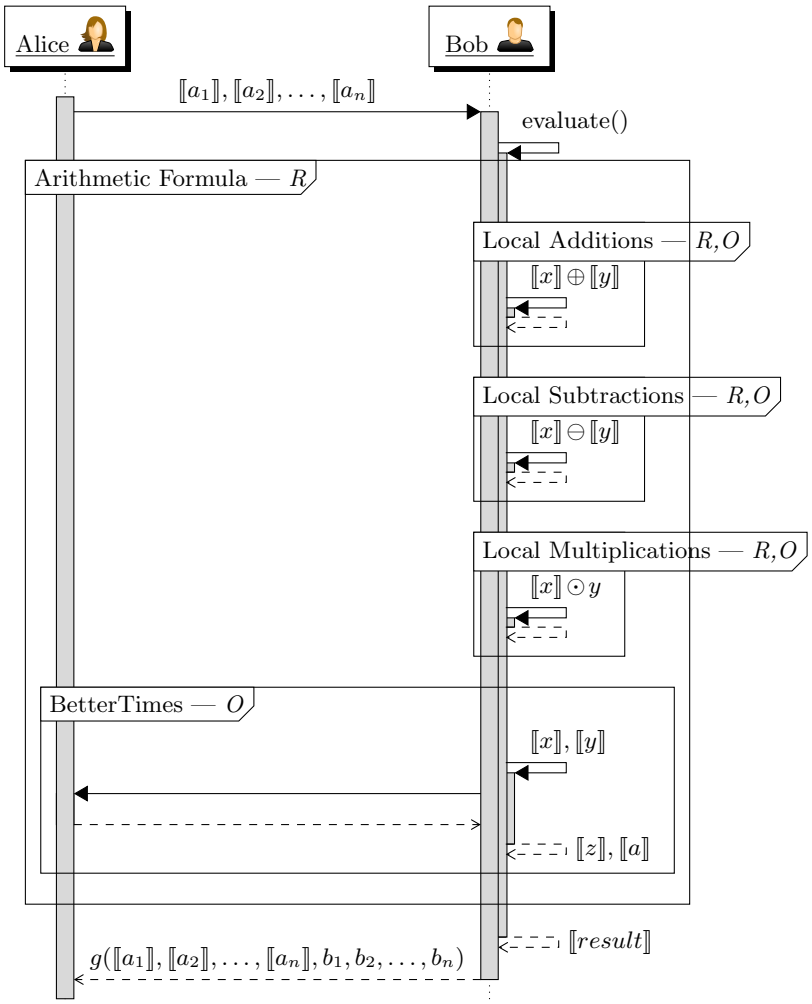

Fig. 7. Visualization of actions by each principal, where R and O means repeatable and optional, respectively.