# SHADE: Secure HAmming DistancE computation from oblivious transfer

Julien Bringer[1], Hervé Chabanne[1,2], and Alain Patey[1,2]

[1] Morpho
[2] Télécom ParisTech
Identity and Security Alliance (The Morpho and Télécom ParisTech Research Center)

**Abstract.** We introduce two new schemes for securely computing Hamming distance in the two-party setting. Our first scheme is a very efficient protocol, based solely on 1-out-of-2 Oblivious Transfer, that achieves full security in the semi-honest setting and one-sided security in the malicious setting. Moreover we show that this protocol is significantly more efficient than the previous proposals, that are either based on garbled circuits or on homomorphic encryption. Our second scheme achieves full security against malicious adversaries and is based on Committed Oblivious Transfer. These protocols have direct applications to secure biometric identification.

**Keywords:** Secure Multi-Party Computation, Hamming Distance, Oblivious Transfer, Biometric Identification

## 1 Introduction

Secure Multiparty Computation (SMC) [40, 15] enables a set of parties to jointly compute a function of their inputs while keeping the inputs private. We here focus on the 2-party case [16], also known as Secure Function Evaluation. Several generic constructions exist in this setting, which apply SMC to any function computed by two parties. In the semi-honest setting, where security is ensured against adversaries following the protocol but trying to gain more information than they should, the Yao's protocol [40, 26] can be used to achieve this purpose using Oblivious Transfers and Garbled Circuits. In the malicious model, where adversaries can follow any strategy, many generic constructions have been proposed [21, 31, 19, 20, 25, 27]. The problem of generic constructions is that they are often far from being optimal when one wants to securely compute specific functions of interest. However, it may happen that generic constructions can be more efficient than specific ones [17].

We here consider the secure computation of the Hamming distance. Concretely, two parties $P_1$ and $P_2$ hold bit strings of the same length $n$, resp. $X = (x_1, \ldots, x_n)$ and $Y = (y_1, \ldots, y_n)$ and want to jointly compute $d_H(X, Y) = \Sigma_{i=1}^{n}(x_i \oplus y_i)$, without $P_1$ (resp. $P_2$) revealing $X$ (resp. $Y$) to $P_2$ (resp. $P_1$). For now, let us consider this problem in the semi-honest setting. It has first been

solved using additive homomorphic encryption [22, 32]. Using this technique, each bit of $P_1$'s input has to be encrypted in one Paillier ciphertext [33] and sent to the other part who can then compute a ciphertext corresponding to the Hamming distance, using homomorphic encryptions. Since Paillier ciphertexts must be at least 2048 bit-long and homomorphic encryptions are multiplications and exponentiations in large groups, this technique is inefficient. However, they also propose in [22] an adaptation of their protocol to the malicious setting. Recently, Huang *et al.* [17] showed that the generic Yao algorithm applied to Hamming distance was more efficient in terms of computation time and bandwidth consumption. Using the Yao algorithm, one needs to describe the function as a binary circuit and then "garble" every gate of this circuit to a table of 4 symmetric ciphertexts. However, using the techniques of [24] and [35], XOR gates do not need to be garbled and garbled gates can be reduced to 3 items. The circuit used in [17] is the succession of $n$ bit-wise (free) XOR's and a *Counter* circuit that adds the results of these XOR's. This Counter Circuit is the bottleneck of their protocol.

The first proposal of our paper achieves full security in the semi-honest model. We almost only rely on 1-out-of-2 oblivious transfer ($OT_1^2$). This primitive enables a receiver to obtain 1 out of 2 elements held by a sender without the sender learning the choice of the receiver and without the receiver learning information on the other element held by the sender. In the Yao algorithm, using oblivious transfers, party $P_2$ gets his input keys for a garbled circuit of the function to compute. However, the keys sent by $P_1$ are independent of $P_1$'s inputs. Here we design our scheme such that, in our oblivious transfers, the elements sent by $P_1$ also depend on the input bits of $P_1$ in such a way that the element obtained by $P_2$ during the $i^{\text{th}}$ $OT_1^2$ depends on $x_i \oplus y_i$. Moreover using the technique of [28, Third Variant], we avoid the use of a costly Counter circuit. We prove, using the OT-hybrid model [6, 25, 16], that our protocol is fully secure in the semi-honest setting or one-sided secure in the malicious setting, depending on the level of security of the underlying $OT_1^2$. This protocol is significantly more efficient than the previous proposals for secure Hamming distance in the semi-honest model [22, 32, 17, 2].

We next extend our first proposal to a second protocol that is fully secure in the malicious setting. Therefore, we use Committed Oblivious Transfer (COT) [8] instead of basic $OT_1^2$. In particular, we use a COT on bit strings with homomorphic commitments, as in [23]. COT enforces that the parties are committed to their inputs to the oblivious transfers and moreover that the receiver is committed to his output. The homomorphic commitment scheme enables us to guarantee that the inputs of the sender are consistent and that the computation run by the receiver on these inputs after the Oblivious Transfers follows the protocol.

We show that our two proposals for efficient secure Hamming Distance computation well extend to the secure computation of weighted Hamming distance used, for instance, in biometric iris matching [11]. We also show that it scales better than previous protocols to the simultaneous computation of several Hamming

distances, which has direct applications in biometric identification, that has so far been one of the main motivations for secure Hamming distance computation [32, 2, 3].

## 2 SMC and Oblivious Transfer

In this section, we first introduce the notions of Oblivious Transfer and Committed Oblivious Transfer, which are the main tools for our proposals. We then recall the definitions of Secure Multi-Party Computation (SMC), more specifically here Secure Two-Party computation. In particular, we recall the definitions of the security properties.

### 2.1 Oblivious Transfer

Oblivious Transfer was first introduced by Rabin [36] as a two-party protocol where a sender has a secret message that he sends to a receiver, which receives it with probability 1/2, without the sender knowing if the message has been received or not. This is however not the version that is now used in secure protocols, but a slightly different primitive called 1-out-of-2 Oblivious Transfer ($OT_1^2$). We here describe this primitive, some extensions to improve its use and a derived version called Committed Oblivious Transfer (COT) [23], used in our second proposal.

**1-out-of-2 Oblivious Transfer** A 1-out-of-2 Oblivious Transfer is a cryptographic primitive that enables a receiver $R$ to obtain 1 out of 2 elements held by a sender, without learning information on the other element and without the sender knowing which element has been chosen. This kind of protocol is stronger than a Private Information Retrieval (PIR) protocol [7] where only the choice of the receiver remains hidden from the sender. The functionality enabled by a $OT_1^2$ is described in Figure 1. For more details on implementations, see for instance [16, Chapter 7]. For instance, the oblivious transfers of [30] and of [34] can be used, respectively, in the semi-honest and in the malicious setting (see Section 2.2 for the security definitions).

- **Inputs:**
  - Sender $S$ inputs two $n$-bit strings $X_0$ and $X_1$
  - Receiver $R$ inputs a choice bit $b$
- **Output:**
  - $S$ learns nothing on $b$
  - $R$ obtains $X_b$ but learns nothing on $X_{1-b}$

**Fig. 1.** The $OT_1^2$ functionality

**Extensions** Several kinds of optimizations can be applied to Oblivious Transfers, independently of the implementation. Two optimizations introduced in [18] are of interest for our proposals. The first one [18, Section 3] enables, in the random oracle model, to compute many OT's with a small elementary cost from $k$ OT's at a normal cost, where $k$ is a security parameter. The second one [18, Appendix B] enables to reduce oblivious transfers of long strings to oblivious transfers of short strings using a pseudo-random generator.

**Committed Oblivious Transfer** Committed Oblivious Transfer (COT) is a combination of $OT_1^2$ and bit commitment, first introduced by Crépeau [8] under the name Verifiable Oblivious Transfer. In this variant, both sender and receiver are committed to their inputs before the oblivious transfer. Moreover, the sender receives a commitment to the receiver's output, and the receiver obtains the randomness for this commitment. To our knowledge, the only scheme that considers COT of bit strings is the one of Kiraz *et al.* [23], which uses an homomorphic cryptosystem as commitment scheme. COT is described in Figure 2, where $Com$ denotes a commitment scheme.

- **Inputs:**
  - $S$ inputs two $n$-bit strings $X_0$ and $X_1$ and two randoms $r_0$ and $r_1$ used for commitment.
  - $R$ inputs a choice bit $b$ and a random $r$ used for commitment
  - The common inputs are $Com(b, r)$, $Com(X_0, r_0)$ and $Com(X_1, r_1)$
- **Output:**
  - $S$ learns nothing on $b$ and $r$
  - $R$ obtains $X_b$ and a random $u$ but learns nothing on $X_{1-b}$, $r_0$ and $r_1$.
  - Both parties obtain $Com(X_b, u)$.

**Fig. 2.** The $COT$ functionality

### 2.2 Secure Two-Party Computation

**Overview** Secure Multi-Party Computation [40] enables a set of parties to jointly compute a function of their inputs while keeping their inputs private. Different kinds of adversaries are considered:

- *semi-honest* adversaries who follow the protocols and try to gain more information than they should on the other parties' inputs,
- *malicious* adversaries who use any kind of strategy to learn information.

There also exists a notion of *covert* adversaries [1] who are malicious but averse to being caught. Notice that we only consider static adversaries.

**The Security Definitions** Informally, security in SMC is ensured by simulating the secure protocol in an ideal model where the inputs of both parties are sent to a trusted party who takes care of the computation and sends the outputs back to the respective parties and showing that all adversarial behaviours in a real execution are simulatable in this ideal model. Full definitions and explanations can be found in [15, 16].

We quickly recall how full security is proved in the malicious setting. Let $\pi$ be a protocol for computing $f(x, y) = (f_1(x, y), f_2(x, y))$. In the real world, a probabilistic polynomial-time (PPT) adversary $A$ sends messages on behalf of the corrupted party and follows an arbitrary strategy while the honest party follows the instructions of $\pi$. In the ideal world, the honest party sends his genuine input $x$ to a trusted party. The adversary sends any input $y'$, of the appropriate size to the trusted party. The trusted party first sends his output $f_1(x, y')$ to the adversary and, if the adversary does not abort, also sends his output $f_2(x, y')$ to the honest party. The adversary is also allowed to abort the protocol at any time. Full Security against a malicious party $P_i$ is ensured if, for any PPT adversary in the real world, there is a PPT adversary in the ideal world such that the distribution of the outputs in the real world is indistinguishable from the distribution of the outputs in the ideal world.

A weaker notion is *Privacy* against a malicious party $P_i$, for $i = 1, 2$, that guarantees that $P_i$ cannot learn any information on the other party's input. However, the execution in the real model might not be simulatable in the ideal model. We say that a protocol achieves *One-Sided Security* in the malicious model if it is fully-secure against a malicious $P_i$ and private against a malicious $P_{3-i}$. See [16, Section 2.6] for further details.

In this paper, we prove security of our schemes in the *OT-hybrid setting* [6, 25, 16]. In this setting, the execution in the real model is slightly modified. The parties have access to a trusted party that computes oblivious transfers for them. We only need to prove indistinguishability between executions in this hybrid model and the ideal model to ensure security.

## 3 Secure Hamming Distance Computation

In the following, the $+$ and $-$ operators respectively denote modular additions and subtractions, we assume that the context is explicit enough and do not recall the moduli in the description of the algorithms. $\bar{x}$, where $x$ is a bit value, denotes $1 - x$. The Hamming distance is denoted by $d_H$.

### 3.1 The Basic Scheme

We here introduce our new scheme based on oblivious transfers. The Yao algorithm [40] also uses oblivious transfers but the inputs of the sender are random keys that are independent of the actual inputs of the sender for the secure computation. In the protocol we propose, the inputs of the sender $P_1$ to the OT's depend on $P_1$'s input bits. Consequently, the output of each oblivious transfer

depends on the input bits $x_i$ of $P_1$ and $y_i$ of $P_2$. We adjust our scheme so that this output depends on $x_i \oplus y_i$. Then, we use a technique inspired by [28, Third Variant] to count the number of bits such that $x_i \oplus y_i = 1$, *i.e.* to compute the Hamming distance.

We assume that parties $P_1$ and $P_2$ respectively hold inputs $X = (x_1, \ldots, x_n)$ and $Y = (y_1, \ldots, y_n)$. Party $P_1$ prepares $n$ random values $r_1, \ldots, r_n \in_R \mathbb{Z}_{n+1}$ and prepares $n$ oblivious transfers, as a sender. The inputs of the $i^{\text{th}}$ transfer are arranged in such a way that a receiver with bit input $y$ gets $r_i + (y \oplus x_i) \bmod n+1$. To do so, input 0 of $P_1$ is set to $r_i + x_i$ and input 1 to $r_i + \bar{x_i}$. Indeed, if $y = 0$, $x_i \oplus y = x_i$ and if $y = 1$, $x_i \oplus y = \bar{x_i}$. $P_2$ acts as a receiver for all these $n$ OT's, with bit inputs $y_1, \ldots, y_n$ and gets $(r_i + (x_i \oplus y_i))_{i=1,\ldots,n}$. Then, $P_2$ adds all these values and gets $T = \Sigma_{i=1}^n r_i + \Sigma_{i=1}^n (x_i \oplus y_i) = R + d_H(X, Y)$, where $R = \Sigma_{i=1}^n r_i$. Finally, depending on the party that is supposed to know the output, either $P_1$ sends $R$ to $P_2$ or $P_2$ sends $T$ to $P_1$, the final output being $D = T - R = d_H(X, Y)$. The protocol is described in Figure 3.

- **Inputs:**
  - $P_1$ inputs a $n$-bit string $X = (x_1, \ldots, x_n)$
  - $P_2$ inputs a $n$-bit string $Y = (y_1, \ldots, y_n)$
- **Output:**
  - $1^{st}$ Option: $P_1$ obtains $d_H(X, Y)$ and $P_2$ obtains nothing
  - $2^{nd}$ Option: $P_2$ obtains $d_H(X, Y)$ and $P_1$ obtains nothing
- **Protocol:**
  1. $P_1$ generates $n$ random values $r_1, \ldots, r_n \in_R \mathbb{Z}_{n+1}$ and computes $R = \Sigma_{i=1}^n r_i$
  2. For each $i = 1, \ldots, n$, $P_1$ and $P_2$ engage in a $OT_1^2$ where
     - $P_1$ acts as the sender and $P_2$ as the receiver.
     - $P_2$'s selection bit is $y_i$.
     - $P_1$'s input is $(r_i + x_i, r_i + \bar{x_i})$.
     - The output obtained by $P_2$ is consequently $t_i = r_i + (x_i \oplus y_i)$.
  3. $P_2$ computes $T = \Sigma_{i=1}^n t_i$
  4. $1^{st}$ Option:
     (a) $P_2$ sends $T$ to $P_1$
     (b) $P_1$ computes and outputs $T - R$
     $2^{nd}$ Option:
     (a) $P_1$ sends $R$ to $P_2$
     (b) $P_2$ computes and outputs $T - R$

**Fig. 3.** The Basic Scheme

**Theorem 1 (Security of the Basic Scheme).**
*Assuming that the underlying $OT_1^2$ is secure in the semi-honest setting, the Basic Scheme achieves full security in the semi-honest setting.*

*Assuming that the underlying $OT_1^2$ is secure in the malicious setting, the Basic Scheme achieves, in the malicious setting:*

– *one-sided security, for the* $2^{nd}$ *option: privacy against a malicious* $P_1$ *and full security against a malicious* $P_2$*,*
– *privacy against a malicious* $P_2$*, for the* $1^{st}$ *option.*

The proofs are detailed in Section 4.1.

### 3.2 The Fully Secure Scheme

**Requirements on the Commitment Scheme** We assume that the commitment scheme used in the Committed Oblivious Transfer we use in our scheme fulfills the following requirements.

First, it must be additively homomorphic, *i.e.* there exist efficient operations $\boxplus$ and $\odot$, such that $Com(x_1, r_1) \odot Com(x_2, r_2) = Com(x_1 + x_2, r_1 \boxplus r_2)$, for any $x_1, x_2, r_1, r_2$.

Second, there must exist a zero-knowledge proof of knowledge $\pi_1^2$, where both parties know a commitment $C = Com(x, r)$ and two values $x_1$ and $x_2$. In this proof, the prover knows $x$, $r$ and proves that $x$ is either $x_1$ or $x_2$. Using the notations of Camenisch and Stadler [4], $\pi_1^2 = PK\{(\alpha, \beta) : C = Com(\alpha, \beta) \wedge (\alpha = x_1 \vee \alpha = x_2)\}$.

Let us consider the commitment scheme used in [23]. This commitment consists of using a (2,2)-threshold homomorphic cryptosystem, *i.e.* $Com(x, r) = Enc(x, r)$ for a homomorphic cryptosystem where the public key is known by both parties and the secret key is shared between the parties. By definition, the first condition is fulfilled (usually $\odot$ is a product and $\boxplus$ an addition). The used cryptosystem can be an additive ElGamal [13] or a Paillier [33] encryption. In both cases, the second condition can be fulfilled (see resp. [5] and [10]). This confirms that our requirements are reasonable.

More details on the COT scheme of [23] can be found in Appendix B and details on the $\pi_1^2$ proofs can be found in Appendix C.

**Our Proposal** Our second scheme adapts the Basic Scheme to the malicious setting. We use a COT with a commitment scheme fulfilling the requirements previously introduced. The commitment, together with the proofs of knowledge of the inputs helps to ensure that the inputs are consistent and that the same values are used along the protocol.

First, $P_1$ and $P_2$ commit to the oblivious transfer inputs and prove that these inputs are well-formed. $P_2$ proves that his inputs are bits and $P_1$ proves that his inputs differ by 1, *i.e.* for each input pair $(a_i, b_i)$, there exists $r_i$ such that $(a_i, b_i) = (r_i, r_i + 1)$ or $(a_i, b_i) = (r_i + 1, r_i)$. COT's are then run with the same inputs as in the basic scheme. Party $P_2$ receives committed outputs, performs the addition of these outputs and a commitment to this addition, thanks to the homomorphic properties of the commitment scheme. $P_2$ can prove, using the commitments, that the value $T$ obtained by adding the results of the COT's is consistent. In the same way, party $P_1$ can prove that the value $R$ is consistent with his inputs to the COT's. Indeed, $\Sigma_{i=1}^n a_i + b_i = \Sigma_{i=1}^n (r_i + r_i + 1) = 2\Sigma_{i=1}^n r_i +$

$\Sigma_{i=1}^{n} 1 = 2R+n$. Using the commitments to the $a_i$'s and to the $b_i$'s, $P_2$ is then able to check if the value $R$ is consistent with the inputs of the COT's. The protocol is described in Figure 4. At any step, if a check fails, the party computing the check should halt the protocol and output $\perp$.

- **Inputs:**
  - $P_1$ inputs a $n$-bit string $X = (x_1, \ldots, x_n)$
  - $P_2$ inputs a $n$-bit string $Y = (y_1, \ldots, y_n)$
- **Output:**
  - $1^{st}$ Option: $P_1$ obtains $d_H(X, Y)$ and $P_2$ obtains nothing
  - $2^{nd}$ Option: $P_2$ obtains $d_H(X, Y)$ and $P_1$ obtains nothing
- **Protocol:**
  1. $P_2$ commits to all his bits $y_i$: he computes and publishes $Com(y_i, \chi_i)$ for each $i = 1 \ldots n$. He also proves, using $\pi_1^2$ proofs on the commitments, that $y_i = 0$ or $y_i = 1$.
  2. $P_1$ generates $n$ random values $r_1, \ldots, r_n$, uniformly from the plaintext space of $Com$, and computes $R = \Sigma_{i=1}^{n} r_i$
  3. For each $i = 1, \ldots, n$, $P_1$ computes $(a_i, b_i) = (r_i + x_i, r_i + \bar{x_i})$ and commits to $a_i$ and $b_i$. He computes and publishes $(A_i = Com(a_i, \alpha_i))_{i=1,\ldots,n}$ and $(B_i = Com(b_i, \beta_i))_{i=1,\ldots,n}$
  4. $P_1$ proves to $P_2$, using $\pi_1^2$ proofs on the commitments, that $|b_i - a_i| = 1$, for each $i = 1, \ldots, n$.
  5. For each $i = 1, \ldots, n$, $P_1$ and $P_2$ engage in a $COT$ where
     - $P_1$ acts as the sender and $P_2$ as the receiver.
     - $P_2$'s selection bit is $y_i$.
     - $P_1$'s input is $(a_i, b_i)$.
     - The output obtained by $P_2$ is $t_i = r_i + (x_i \oplus y_i)$ and $\tau_i$.
     - Both parties obtain $C_i = Com(t_i, \tau_i)$
  6. $P_2$ computes $T = \Sigma_{i=1}^{n} t_i$,
  7. $1^{st}$ Option:
     (a) $P_2$ computes $C = Com(T, \tau) = C_1 \odot \ldots \odot C_n$
     (b) $P_2$ sends $T$ and a zero-knowledge proof that $C$ commits to $T$ to $P_1$
     (c) $P_1$ computes $C = C_1 \odot \ldots \odot C_n$ and checks the proof.
     (d) $P_1$ computes and outputs $T - R$

     $2^{nd}$ Option:
     (a) $P_1$ computes $K = Com(2R+n, \rho) = A_1 \odot \ldots \odot A_n \odot B_1 \odot \ldots \odot B_n$
     (b) $P_1$ sends $R$ and a zero-knowledge proof that $K$ commits to $2R+n$ to $P_2$
     (c) $P_2$ computes $K = A_1 \odot \ldots \odot A_n \odot B_1 \odot \ldots \odot B_n$ and checks that $K = Com(2R + n, \rho)$.
     (d) $P_2$ computes and outputs $T - R$

**Fig. 4.** The Fully Secure Scheme

**Theorem 2 (Security of the Fully Secure Scheme).** *Assuming that the underlying COT is secure in the malicious setting, the Fully Secure Scheme achieves full security in the malicious setting.*

The proofs are detailed in Section 4.2.

# 4 Security Proofs

## 4.1 The Basic Scheme

We here give the proof of security against a malicious $P_2$ in the case of the $2^{nd}$ option. The guarantees of privacy against a malicious $P_2$ for the $1^{st}$ option, or against a malicious $P_1$ for the $2^{nd}$ option are easily deduced from the privacy of the OT's, since no other messages are sent to these parties during the protocol.

**Theorem 3 (Full Security against a Malicious $P_2$-$2^{nd}$ option).** *Assuming that the underlying $OT_1^2$ is secure in the malicious setting, the Basic Scheme, following the $2^{nd}$ option, is fully-secure against a malicious $P_2$ in the OT-hybrid setting.*

The following proof is partially inspired from the proofs of [28]. Indeed, our scheme can be viewed as a reduction of the third variant of their Oblivious Automata Evaluation, with only one state per line of the matrix, but where the lines of the matrix are not identical.

*Proof.* Let $B$ be a PPT adversary controlling $P_2$ in the real world, we describe a simulator $S_B$ who simulates the view of $B$ in the ideal world.

$S_B$ runs $B$ on input $Y$. Since we operate in the OT-hybrid model, $B$ sends $Y' = (y'_1, \ldots, y'_n)$ to the OT oracle. $S_B$ sends $Y'$ to the trusted party and obtains $D = d_H(X, Y')$. $S_B$ picks $n$ random values $t_1, \ldots, t_{n-1}, T \in_R \mathbb{Z}_{n+1}$ and computes $t_n = T + D - \Sigma_{i=1}^{n-1} t_i$. $S_B$ sends the $t_i$'s to $B$ as results of the oblivious transfer. He then sends $T$. $S_B$ then outputs whatever $B$ outputs.

Let us now prove the indistinguishability between the real and the simulated views. Let $V$ be a random subset of size $t$ of $\{1, \ldots, n\}$. ($V$ represents the bit positions where $x_i \oplus y_i = 1$.) Consider the distributions:

- $(D_V)$: Choose $n$ uniformly random values $\{r_1, \ldots, r_n\} \in \mathbb{Z}_{n+1}$. For every $i \in \{1, \ldots, n\}$, let $r'_i = r_i + 1$ if $i \in V$ and $r'_i = r_i$ otherwise. Output $(r'_1, \ldots, r'_n)$.
- $(D'_V)$: Choose $n$ uniformly random values $R, r'_1, \ldots, r'_{n-1} \in \mathbb{Z}_{n+1}$. Let $R' = R + t$ and $r'_n = R' - \Sigma_{i=1}^{n} r'_i$. Output $(r'_1, \ldots, r'_n)$.

It is easy to show that $D_V$ and $D'_V$ are identically distributed and that sampling from $D'_V$ only requires the knowledge of $t$. We can now notice that the distribution $D_V$ represents the view of $B$ in a real execution of the protocol while our simulator $S_B$ samples from $D'_V$, with the only knowledge of the final output. Thus, the view of $P_2$ in the real world and the simulated view of $P_2$ in the ideal world are indistinguishable, which ensures full security against a malicious $P_2$.

□

*Remark 1.* The proofs of security in the semi-honest setting are straightforward, given the security guarantees of the Oblivious Transfer and the arguments explained in the previous proof proving that the outputs of the OT's give no information on the inputs of $P_2$.

### 4.2 The Fully Secure Scheme

We use an adaptation of the OT-hybrid model to Committed Oblivious Transfer. When the parties engage a COT in the COT-hybrid model, parties interact with each other and have access to a trusted party that computes the COT for them. Concretely, the receiver sends $b, Com(b, r)$ to the trusted party, the sender sends $x_0, Com(x_0, r_0)$ and $x_1, Com(x_1, r_1)$ to the trusted party. The trusted party sends $x_b$ and $r'$ back to the receiver and $Com(x_b, r')$ to both parties. This model, for a slightly different COT, has already been used in the proof of security of the binHDOT protocol [22] for malicious adversaries.

Notice that, since we use zero-knowledge proofs of knowledge, our protocol cannot be proved secure in the UC model [6] but in the stand-alone setting only.

**Theorem 4 (Full Security Against a Malicious $P_1$).** *Assuming that the underlying COT is secure in the malicious setting, the Fully Secure Scheme is fully-secure against a malicious $P_1$ in the COT-hybrid setting.*

*Proof.* Let $B$ be a PPT adversary controlling $P_1$ in the real world, we describe a simulator $S_B$ who simulates the view of $B$ in the ideal world.

$S_B$ runs $B$ on input $X$. $S_B$ commits to random bits $(y'_1, \ldots, y'_n)$ and runs the proofs that an honest $P_2$ would run. He receives the commitments of $B$ and checks the proofs like an honest $P_2$. Then, $S_B$ gets the inputs $(a_i, b_i, A_i, B_i)_{i=1,\ldots,n}$ of $B$ to the COT oracle and easily deduces the couples $(r'_i, x'_i)_{i=1,\ldots,n}$ such that $(a_i, b_i) = (r'_i + x'_i, r'_i + \overline{x'_i})$. Such values must exist, and are unique, since $B$ successfully proved that $|a_i - b_i| = 1$. $S_B$ sends $X'$ to the trusted party and obtains $D = d_H(X', Y)$. $S_B$ chooses $n$ random $\tau_i$'s, sets $t'_i = r'_i + 1$, for $i = 1, \ldots, D$ and $t'_i = r'_i$ otherwise. He then computes $(Com(t'_i, \tau_i))_{i=1,\ldots,n}$. $S_B$ sends these commitments back to $B$. $S_B$ then follows the protocol until the end, the checks on the dot-products of commitments ensuring that everything is consistent.

It is relatively easy to show that this simulation is indistinguishable from the view of $P_1$ in the real world, which ensures full security against a malicious $P_1$. □

**Theorem 5 (Full Security Against a Malicious $P_2$).** *Assuming that the underlying COT is secure in the malicious setting, the Fully Secure Scheme is fully-secure against a malicious $P_2$ in the COT-hybrid setting.*

*Proof.* Let $B$ be a PPT adversary controlling $P_2$ in the real world, we describe a simulator $S_B$ who simulates the view of $B$ in the ideal world.

$S_B$ runs $B$ on input $Y$. $S_B$ commits to random inputs $(r'_i, r'_i + 1)_{i=1,\ldots,n}$ and runs the proofs that an honest $P_1$ would run. He receives the commitments of $B$ and checks the proofs like an honest $P_1$. Then, $S_B$ gets the inputs $(y'_i, \chi_i)_{i=1,\ldots,n}$

of $B$ to the COT oracle. $S_B$ sends $Y'$ to the trusted party and obtains $D = d_H(X, Y')$. $S_B$ sets $t'_i = r'_i$ for $i = 1, \ldots, D$ and $t'_i = r'_i$ otherwise. $S_B$ chooses $n$ random $\tau_i$'s and computes $(C'_i = Com(t'_i, \tau_i))_{i=1,\ldots,n}$. $S_B$ sends $(t'_i, \tau_i)_{i=1,\ldots,n}$ back to $B$ as outputs of the COT. $S_B$ then follows the protocol until the end.

Using arguments similar to the proof of Theorem 3, it is relatively easy to show that this simulation is indistinguishable from the view of $P_2$ in the real world, which ensures full security against a malicious $P_2$. □

## 5 Extensions

In the following, we only consider the basic scheme. Similar adaptations can be applied to the fully secure scheme.

### 5.1 Weighted Hamming Distance

Secure Hamming distance computation is an important tool for secure biometric identification, when Hamming distance is the actual matching operation (see [32] for an example of application).

In the field of iris recognition [11], the most common representation of biometric templates is the IrisCode representation. The iris is encoded as a 2048-bit code, together with a 2048-bit mask that indicates whether the corresponding bits in the code represent a genuine iris part or not (some parts are, for instance, covered by eyelids). The matching between two IrisCodes $(X_1, M_1)$ and $(X_2, M_2)$ (where $X_i$ is the code and $M_i$ is the mask) is computed using the weighted Hamming distance operation $HD$:

$$HD(X_1, X_2, M_1, M_2) = \frac{||(X_1 \oplus X_2) \cap M_1 \cap M_2||}{||M_1 \cap M_2||} = \frac{\Sigma_{i=1}^{n}(x_{1i} \oplus x_{2i}) \cdot m_{1i} \cdot m_{2i}}{\Sigma_{i=1}^{n} m_{1i} \cdot m_{2i}}$$

Secure Computation of weighted Hamming distance has already been considered in [2, 3] using homomorphic encryption and garbled circuits. We can help improving these protocols using the first part of our proposals, *i.e.* until the end of the oblivious transfers. Indeed, we can achieve the computation of both numerator and denominator in the weighted Hamming distance computation but not the division. However, in secure biometric identification protocols, the value of interest is not the actual matching score but whether this score is above or under a given threshold. The outputs of the first part of the protocol, modified to compute the numerator and the denominator, can then be used as inputs for another circuit that outputs the desired result, as in [2, 3].

We do not describe the full modification to obtain a secure computation of the numerator and denominator of the weighted Hamming distance, since it is quite straightforward. Only notice that 1-out-of-4 OT's have to be used, instead of $OT_1^2$'s (or even 1-out-of-3 OT's if one rewrites IrisCodes as chains of 2048 elements from $\{0, 1, \epsilon\}$ where $\epsilon$ denotes an erasure, *i.e.* a position where the mask is 0).

### 5.2 Many at Once

Another important advantage of our scheme is that it well extends to the computation of many Hamming distances at the same time. Let $X^1, \ldots, X^m$ be the inputs of $P_1$ and $Y$ still be the input of $P_2$. We can adapt the Basic Scheme as follows. The step indices correspond to the description of Figure 3.

During step 1, instead of generating $n$ random values, $P_1$ generates $m \times n$ random values $(r_i^j)_{i=1,\ldots,n,j=1,\ldots,m}$. During step 2, the inputs of $P_1$ are concatenations of $m$ values: input 0 is $(r_i^1 + x_i^1 || \ldots || r_i^m + x_i^m)$ and input 1 is $(r_i^1 + \overline{x_i^1} || \ldots || r_i^m + \overline{x_i^m})$. The output of $P_2$ is thus $(t_i^1 = r_i^1 + (x_i^1 \oplus y_i) || \ldots || t_i^m = r_i^m + (x_i^m \oplus y_i))$. During step 3, $P_2$ computes, for $i = 1, \ldots, m$, $T^j = \Sigma_{i=1}^n t_i^j$. During step 4, either $P_1$ sends the $R^j$'s to $P_2$ or $P_2$ sends the $T^j$'s to $P_1$ and the outputs are $(D^j = T^j - R^j)_{j=1,\ldots,m}$.

This extension is useful for the context of biometric identification [32, 2, 3], where several matching scores of the same biometric template against a biometric database have to be performed.

### 5.3 Secure Evaluation of A Larger Class of Functions

We can see that our scheme is easily adaptable to the class of functions that are linear combinations of functions taking two binary inputs, *i.e.* all the functions $f$ such that $\forall X = (x_1, \ldots, x_n) \in \{0,1\}^n, Y = (y_1, \ldots, y_n) \in \{0,1\}^n$, $f(X,Y) = \Sigma_{i=1}^n \lambda_i f_i(x_i, y_i)$. The adaptation is straightforward. When looking at Figure 3, the $R$ value is now computed as $R = \Sigma_{i=1}^n \lambda_i r_i$. The inputs of $P_1$ to the OT's are, for each $i$, $f(x_i, 0) + r_i$ and $f(x_i, 1) + r_i$, so that $P_2$ obtains $t_i = r_i + f(x_i, y_i)$. $T$ is now computed as $T = \Sigma_{i=1}^n \lambda_i t_i$. The rest of the protocol is unchanged.

## 6 Efficiency

### 6.1 The Basic Scheme

The cost of the basic scheme described in Figure 3 is essentially the cost of $n$ $OT_1^2$'s of inputs of $\log(n)$ bits. Using the OT extension of [18], when many OT's are performed, the workload turns out to be two evaluations of a hash function for $P_1$ and one for $P_2$ per input bit. The bandwidth requirement is then roughly $2n.\log(n)$ bits.

**Comparison to Previous Schemes** Let us compare our Basic Scheme to two previous protocols [17], [22, 32] for semi-honest secure Hamming Distance computation, previously known as the most efficient proposals.

Other techniques, like Private Set Intersection Cardinality [9] or Private Scalar Product Computation [14] can be easily adapted to perform secure Hamming distance computation. However, in these proposals, use of homomorphic encryption and/or a linear number of exponentiations leads to schemes that are less efficient than our proposal in the semi-honest model.

We first compare to the application of the Yao algorithm to Hamming distance computation described in [17]. In this setting, the Hamming distance function has to be represented as a binary circuit. To get an idea of the cost of the computation, we need to count the number of non-XOR gates in this circuit. Let us assume that the size $n$ of the inputs is a power of 2: $n = 2^N$. The number $G$ of non-free gates is obtained (see the description of the Counter Circuit in [17]) by $G = \Sigma_{i=1}^{N}(2^{N-i}.i) \approx 2^{N+1} = 2n$. Let $k$ be the security parameter of the scheme. For the generation of the circuit, party $P_1$ has to perform $4G$ hash function evaluations. Then, $P_1$ sends the circuit ($3k.G$ bits) and his keys for the circuit ($n.k$ bits). Then $P_1$ and $P_2$ perform $n$ $OT_1^2$'s on $k$-bit strings. $P_2$ has then to perform $G$ hash functions evaluations. Using the OT extension of [18], the workload of $P_1$ is roughly $10n$ hash functions evaluations, the workload of $P_2$ is $3n$ hash function evaluations and the bandwidth is $6kn$ bits. When $m$ Hamming distances on the same input of $P_2$ are evaluated, all these operations but the oblivious transfers of $P_2$'s inputs have to be computed $m$ times.

We now evaluate the workload and bandwidth requirements of the [22, 32] algorithm. The binHDOT protocol presented in [22] enables evaluation of a class of functions depending on Hamming distance. We here consider its reduction to the evaluation of the Hamming distance only. We describe the corresponding protocol in Appendix A. We moreover take into account, in our evaluations, the optimizations presented in [32].

Party $P_2$ prepares $n$ homomorphic ciphertexts, encrypting each of his inputs bits. These ciphertexts are sent to $P_1$ who homomorphically adds and subtracts them to obtain the encryption of the Hamming distance. Taking into account the optimizations of [32] (although we do not separate off-line and on-line phases), $P_1$ has to perform $n$ homomorphic encryptions and $P_2$ $n$ homomorphic additions. They mainly exchange $n$ ciphertexts. When $m$ distances are computed, with the optimizations of [32], $P_2$'s work is almost the same and $P_1$ has to perform $mn/2$ homomorphic additions, once $n$ subtractions and $3.5n$ additions are preprocessed. The bandwidth depends on the option and on the receiver of the result.

The comparison of these 3 protocols is summed up in Table 1, where **hash** means hash function evaluations and $k$ is the security parameter of the Yao algorithm of [17]. We extrapolate to the simultaneous computation of $m$ Hamming distances in the setting of Section 5.2 in Table 2. In the first line of Table 2, the $(+m)$ hom. ciphertexts corresponds to the case where $P_2$ gets the result instead of $P_1$.

For concrete estimations, $k$ should be at least 80 and Paillier ciphertexts at least 2048-bit long. It is easy to see that, for reasonable sizes of $n$, our scheme is more efficient and requires significantly less bandwidth. In these tables, we do not mention the $k$ base OT's that are needed in our basic scheme and in the scheme of [17] for OT extension. They can be performed in a preprocessing phase.

| | $P_1$ | $P_2$ | Bandwidth (bits) |
|---|---|---|---|
| [22, 32] | $n$ hom.add. | $n$ hom.enc. | $n$ hom.ciphertexts |
| [17] | $10n$ **hash** | $3n$ **hash** | $6kn$ |
| The Basic Scheme | $2n$ **hash** | $n$ **hash** | $2n\log(n)$ |

**Table 1.** Secure Computation of One Hamming Distance in the Semi-Honest Model

| | $P_1$ | $P_2$ | Bandwidth (bits) |
|---|---|---|---|
| [22, 32] | $mn/2$ hom.add. | $n$ hom.enc. | $n(+m)$ hom.ciphertexts |
| [17] | $(2+8m)n$ **hash** | $(1+2m)n$ **hash** | $(2+4m)kn$ |
| The Basic Scheme | $2n$ **hash** | $n$ **hash** | $2mn\log(n)$ |

**Table 2.** Secure Computation of $m$ Hamming Distances in the Semi-Honest Model

**Implementation Results** To prove our allegations regarding efficiency improvements in terms of computational workload, we ran the implementation of secure Hamming distance used in [17] and an implementation of our basic scheme using the same framework [38] on the same computer. The framework is implemented in Java and we ran it on a single computer with a 2 GHz Intel Core i7 processor and a 4 GB RAM. We think that the ratio of computation times between the protocols is more relevant than an absolute value of the time of execution of our process. This comparison is illustrated in Figure 5. For inputs with a few thousands bits size, the computation time required for our Basic scheme is approximately 22% of the time required to compute the protocol of [17].

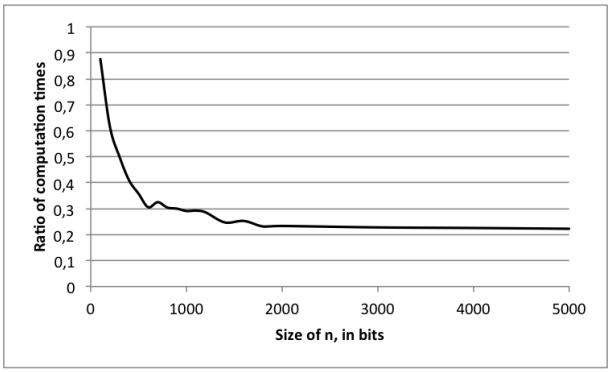


**Fig. 5.** Ratio computation times between our Basic Scheme and the protocol of [17]

### 6.2 The Fully Secure Scheme

We assume that the $COT$ of the Fully Secure Scheme is the one of [23], using a threshold El-Gamal cryptosystem. According to [23], 24 exponentiations are required per $COT$, once the inputs are committed.

$P_1$ performs $2n$ commitments and runs $n$ $\pi_1^2$ proofs on the commitments. He participates in $n$ COT's as a sender. He finally computes a product of $n$ ciphertexts (or $2n$ for the $2^{nd}$ option). $P_2$ performs $n$ commitments and runs $n$ $\pi_1^2$ proofs on the commitments. He participates in $n$ COT's as a receiver. He finally computes a product of $n$ ciphertexts (or $2n$ for the $2^{nd}$ option). The bandwidth mainly comprises $3n$ commitments and $n$ COT's.

In [22], Jarrous and Pinkas also propose an adaptation of their binHDOT protocol to the malicious setting. They also use a particular Committed Oblivious Transfer functionality, with proofs that the inputs differ by a constant number $\Delta$, while we prove that our inputs always differ by 1. However, their protocol (for a more generic functionality) ends with an oblivious polynomial evaluation.

## References


1. Aumann, Y., Lindell, Y.: Security against covert adversaries: Efficient protocols for realistic adversaries. In: Vadhan, S.P. (ed.) TCC. Lecture Notes in Computer Science, vol. 4392, pp. 137–156. Springer (2007)
2. Blanton, M., Gasti, P.: Secure and efficient protocols for iris and fingerprint identification. In: Atluri, V., Díaz, C. (eds.) ESORICS. Lecture Notes in Computer Science, vol. 6879, pp. 190–209. Springer (2011)
3. Bringer, J., Chabanne, H., Favre, M., Patey, A.: Faster secure computation for biometric identification using filtering. In: IAPR International Conference on Biometrics (ICB) (2012)
4. Camenisch, J., Stadler, M.: Efficient group signature schemes for large groups (extended abstract). In: Jr., B.S.K. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 1294, pp. 410–424. Springer (1997)
5. Camenisch, J., Stadler, M.: Proof systems for general statements about discrete logarithms. Tech. rep., Dept. of Computer Science, ETH Zurich (1997)
6. Canetti, R.: Security and composition of multiparty cryptographic protocols. J. Cryptology 13(1), 143–202 (2000)
7. Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private information retrieval. In: FOCS. pp. 41–50. IEEE Computer Society (1995)
8. Crépeau, C.: Verifiable disclosure of secrets and applications (abstract). In: Quisquater, J.J., Vandewalle, J. (eds.) EUROCRYPT. Lecture Notes in Computer Science, vol. 434, pp. 150–154. Springer (1989)
9. Cristofaro, E.D., Gasti, P., Tsudik, G.: Fast and private computation of set intersection cardinality. IACR Cryptology ePrint Archive 2011, 141 (2011)
10. Damgård, I., Jurik, M.: A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In: Kim, K. (ed.) Public Key Cryptography. Lecture Notes in Computer Science, vol. 1992, pp. 119–136. Springer (2001)
11. Daugman, J.: How iris recognition works. In: ICIP (1). pp. 33–36 (2002)

12. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 263, pp. 186–194. Springer (1986)
13. Gamal, T.E.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO. Lecture Notes in Computer Science, vol. 196, pp. 10–18. Springer (1984)
14. Goethals, B., Laur, S., Lipmaa, H., Mielikäinen, T.: On private scalar product computation for privacy-preserving data mining. In: Park, C., Chee, S. (eds.) ICISC. Lecture Notes in Computer Science, vol. 3506, pp. 104–120. Springer (2004)
15. Goldreich, O.: The Foundations of Cryptography - Volume 2, Basic Applications. Cambridge University Press (2004)
16. Hazay, C., Lindell, Y.: Efficient Secure Two-Party Protocols. Springer (2010)
17. Huang, Y., Evans, D., Katz, J., Malka, L.: Faster secure two-party computation using garbled circuits. In: USENIX Security Symposium. USENIX Association (2011)
18. Ishai, Y., Kilian, J., Nissim, K., Petrank, E.: Extending oblivious transfers efficiently. In: Boneh, D. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 2729, pp. 145–161. Springer (2003)
19. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding cryptography on oblivious transfer - efficiently. In: Wagner [39], pp. 572–591
20. Ishai, Y., Prabhakaran, M., Sahai, A.: Secure arithmetic computation with no honest majority. In: Reingold [37], pp. 294–314
21. Jarecki, S., Shmatikov, V.: Efficient two-party secure computation on committed inputs. In: Naor [29], pp. 97–114
22. Jarrous, A., Pinkas, B.: Secure hamming distance based computation and its applications. In: Abdalla, M., Pointcheval, D., Fouque, P.A., Vergnaud, D. (eds.) ACNS. Lecture Notes in Computer Science, vol. 5536, pp. 107–124 (2009)
23. Kiraz, M.S., Schoenmakers, B., Villegas, J.: Efficient committed oblivious transfer of bit strings. In: Garay, J.A., Lenstra, A.K., Mambo, M., Peralta, R. (eds.) ISC. Lecture Notes in Computer Science, vol. 4779, pp. 130–144. Springer (2007)
24. Kolesnikov, V., Schneider, T.: Improved garbled circuit: Free xor gates and applications. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP (2). Lecture Notes in Computer Science, vol. 5126, pp. 486–498. Springer (2008)
25. Lindell, Y., Pinkas, B.: An efficient protocol for secure two-party computation in the presence of malicious adversaries. In: Naor [29], pp. 52–78
26. Lindell, Y., Pinkas, B.: A proof of security of Yao's protocol for two-party computation. J. Cryptology 22(2), 161–188 (2009)
27. Lindell, Y., Pinkas, B.: Secure two-party computation via cut-and-choose oblivious transfer. In: Ishai, Y. (ed.) TCC. Lecture Notes in Computer Science, vol. 6597, pp. 329–346. Springer (2011)
28. Mohassel, P., Niksefat, S., Sadeghian, S.S., Sadeghiyan, B.: An efficient protocol for oblivious DFA evaluation and applications. In: Dunkelman, O. (ed.) CT-RSA. Lecture Notes in Computer Science, vol. 7178, pp. 398–415. Springer (2012)
29. Naor, M. (ed.): Advances in Cryptology - EUROCRYPT 2007, 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Barcelona, Spain, May 20-24, 2007, Proceedings, Lecture Notes in Computer Science, vol. 4515. Springer (2007)
30. Naor, M., Pinkas, B.: Efficient oblivious transfer protocols. In: Kosaraju, S.R. (ed.) SODA. pp. 448–457. ACM/SIAM (2001)

31. Nielsen, J.B., Orlandi, C.: LEGO for two-party secure computation. In: Reingold [37], pp. 368–386
32. Osadchy, M., Pinkas, B., Jarrous, A., Moskovich, B.: Scifi - a system for secure face identification. In: IEEE Symposium on Security and Privacy. pp. 239–254. IEEE Computer Society (2010)
33. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT. Lecture Notes in Computer Science, vol. 1592, pp. 223–238. Springer (1999)
34. Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: Wagner [39], pp. 554–571
35. Pinkas, B., Schneider, T., Smart, N.P., Williams, S.C.: Secure two-party computation is practical. In: Matsui, M. (ed.) ASIACRYPT. Lecture Notes in Computer Science, vol. 5912, pp. 250–267. Springer (2009)
36. Rabin, M.O.: How to exchange secrets with oblivious transfer. Tech. Rep. TR-81, Aiken Computation Lab, Harvard University (1981)
37. Reingold, O. (ed.): Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings, Lecture Notes in Computer Science, vol. 5444. Springer (2009)
38. University of Maryland, University of Virginia: Might be evil: Privacy-preserving computing, http://mightbeevil.com
39. Wagner, D. (ed.): Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings, Lecture Notes in Computer Science, vol. 5157. Springer (2008)
40. Yao, A.C.C.: How to generate and exchange secrets (extended abstract). In: FOCS. pp. 162–167. IEEE Computer Society (1986)

## A A Homomorphic Secure Hamming Distance Computation Protocol

We describe a secure Hamming distance protocol derived from the protocol of [22]. We simplify the protocol of [22] since it is more "powerful" and enables to compute a function depending on the Hamming distance and not only the Hamming distance itself. Similar mechanisms to the protocol of [22] are also present in [2].

We sum up the protocol. Party $P_1$ owns a secret key for an additively homomorphic scheme $E$ and $P2$ knows the corresponding public key. $P_1$ encrypts all his input bits and sends the ciphertexts to $P_2$, who can compute using homomorphic operations the encryption of the Hamming distance between the parties' inputs. Finally, $P_2$ either sends the resulting ciphertext back to $P1$ who decrypts it and learns the result or homomorphically adds a random number and sends the resulting ciphertext to $P_1$ who decrypts and sends the decryption result back to $P_2$ who can subtract the random number and obtain the result. The protocol is described in Figure 6. It is secure in the semi-honest setting.

## B The COT protocol of Kiraz et al.

We here briefly describe the COT protocol of [23]. We first recall the COT functionality in Figure 7.

– **Inputs:**
  - $P_1$ inputs a $n$-bit string $X = (x_1, \ldots, x_n)$
  - $P_2$ inputs a $n$-bit string $Y = (y_1, \ldots, y_n)$
  - $P_1$ owns a secret/public key couple $(sk, pk)$ for an additively homomorphic scheme $E$. $P_2$ only knows the public key $pk$.
– **Output:**
  - $1^{st}$ Option: $P_1$ obtains $d_H(X, Y)$ and $P_2$ obtains nothing
  - $2^{nd}$ Option: $P_2$ obtains $d_H(X, Y)$ and $P_1$ obtains nothing
– **Protocol:**
  1. $P_1$ encrypts the bits of $X$. He computes $E_1 = E_{pk}(x_1), \ldots, E_n = E_{pk}(x_n)$
  2. $P_1$ sends the $E_i$'s to $P_2$
  3. For each $i = 1, \ldots, n$, $P_2$ computes $F_i = E_i^{1-2y_i} \cdot E_{pk}(y_i) = E_{pk}(x_i - 2x_i \cdot y_i + y_i) = E_{pk}(x_i \oplus y_i)$
  4. $P_2$ computes $F = \Pi_{i=1}^{n} F_i = E_{pk}(d_H(X, Y))$
  5. $1^{st}$ Option:
     (a) $P_2$ sends $F$ to $P_1$
     (b) $P_1$ decrypts $F$ and outputs $D_{sk}(F) = d_H(X, Y)$
     $2^{nd}$ Option:
     (a) $P_2$ picks a random $R$, uniformly from the plaintext space of $E$, and computes $G = F \cdot E_{pk}(R) = E_{pk}(d_H(X, Y) + R)$
     (b) $P_2$ sends $G$ to $P_1$
     (c) $P_1$ decrypts $G$ and obtains $H = d_H(X, Y) + R$
     (d) $P_1$ sends $H$ to $P_2$
     (e) $P_2$ computes and outputs $H - R = d_H(X, Y)$

**Fig. 6.** A Homomorphic Encryption-Based Secure Hamming Distance Computation Protocol

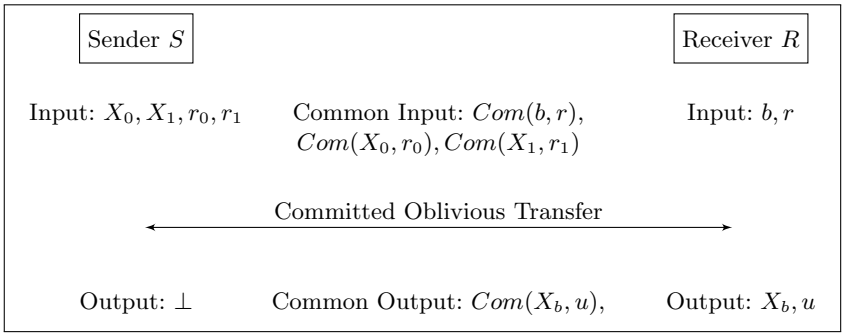


**Fig. 7.** The COT Functionality

Let $E$ be a (2,2)-threshold additive homomorphic scheme, *i.e.* a public-key cryptosystem such that:

– $E(x, r_1) \cdot E(y, r_2) = E(x + y, r_1 + r_2)$ and $E(n \cdot x) = E(x)^n$

– The secret key $sk$ of the scheme is divided into two shares $sk_1$ and $sk_2$.
– The decryption is performed using two algorithms $D$ and $R$ such that

$$R(D_{sk_1}(E(x)), D_{sk_2}(E(x))) = x$$

In the COT scheme of [23], the commitment scheme is a (2,2)-threshold cryptosystem. Consequently, the sender $S$ and the receiver $R$ own, respectively, shares $sk_S$ and $sk_C$ of the secret key. To commit to an element $x$ using random $r$, one computes $Com(x, r) = E(x, r)$.

The scheme is based on the following remark: the output of the receiver is $X_b = b \cdot (X_1 - X_0) + X_0$. Using the homomorphic properties of the encryption $E$, we then have $E(X_b) = E(b)^{X_0 - X_1} \cdot E(X_0)$. This can be computed by the sender, using his private inputs $X_0$ and $X_1$ and the commitments $E(b)$ and $E(X_0)$. The sender sends this value, together with his decryption share to the receiver. The receiver computes the other decryption share and retrieves the output of the oblivious transfer. This process goes with several proofs, that we do not detail here. The scheme of [23] is summed up in Figure 8.

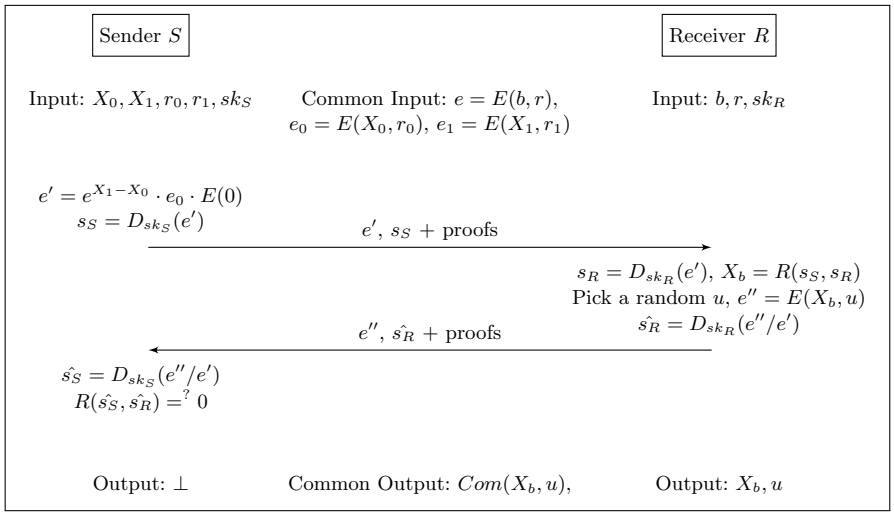


**Fig. 8.** The COT Scheme of [23]

## C $\pi_1^2$ Proofs

We here detail the $\pi_1^2$ proofs required at the beginning of the Fully Secure Scheme, that prove that a given homomorphic ciphertext encrypts one out of two given values. We instantiate them in the interactive setting, without random oracle. They can easily be adapted to a non-interactive setting, in the random oracle model, using the Fiat-Shamir heuristic [12].

### C.1 Paillier Cryptosystem

First, let us consider the case of the Paillier cryptosystem [33]. The proof is introduced in [10, Section 4.2]. Let $E$ be Paillier encryption with public key $(n, g)$.

The inputs of the prover are an element $x \in \mathbb{Z}_n$ and a random $r \in \mathbb{Z}^*_{n^2}$. The common inputs of the prover and the verifier are $x_1$, $x_2$ and $E(x, r) = g^x r^n \mod n^2$. We assume, w.l.o.g. that $x = x_1$. Let $k$ denote the bit-length of $n$ and $t = k/2$. Prover $P$ and Verifier $V$ proceed as follows:

1. $P$ and $V$ compute $u_1 = E(x)/g^{x_1} (= r^n)$ and $u_2 = E(x)/g^{x_2}$
2. $P$ picks a random $z_2 \in_R \mathbb{Z}^*_{n^2}$, a random $k$-bit number $e_2$ and sets $a_2 = z_2^n u_2^{-e_2} \mod n^2$.
3. $P$ picks a random $r_1 \in \mathbb{Z}_{n^2}$ and sets $a_1 = r_1^n \mod n^2$
4. $P$ sends $a_1$ and $a_2$ to $V$
5. $V$ chooses a random $t$-bit number $s$ and sends it to $P$
6. $P$ computes $e_1 = s - e_2 \mod 2^t$ and $z_1 = r_1 r^{e_1} \mod n^2$.
7. $P$ sends $e_1, z_1, e_2, z_2$ to $V$
8. $V$ checks that $s = e_1 + e_2 \mod 2^t$, $z_1^n = a_1 u_1^{e_1} \mod n^2$ and $z_2^n = a_2 u_2^{e_2} \mod n^2$ and accepts if and only if all checks succeed.

### C.2 ElGamal Cryptosystem

Now consider the additive ElGamal cryptosystem [13]. The underlying proof on discrete logarithms can be found in [5, Example 3]. Let $E$ be an ElGamal encryption with public key $h$ in a group $G$ of order $q$ and generator $g$.

The inputs of the prover are an element $x$ and a random $r$. The common inputs of the prover and the verifier are $x_1, x_2$ and $E(x, r) = (g^r, g^x h^r) = (b_1, b_2)$. We assume, w.l.o.g. that $x = x_1$. Prover $P$ and Verifier $V$ proceed as follows:

1. $P$ and $V$ compute $u_1 = b_2/g^{x_1} (= h^r)$ and $u_2 = b_2/g^{x_2}$
2. $P$ picks random $v_1, v_2, c_2 \in_R \mathbb{Z}_q$ and computes $t_1 = h^{v_1}$ and $t_2 = u_2^{c_2} h^{v_2}$.
3. $P$ sends $t_1$ and $t_2$ to $V$
4. $V$ picks a random $c \in_R \mathbb{Z}_q$ and sends it to $V$
5. $P$ computes $c_1 = c - c_2$, $r_1 = v_1 - c_1.r$ and $r_2 = v_2$
6. $P$ sends $c_1, r_1, c_2, r_2$ to $V$
7. $V$ checks that $c = c1 + c_2$, $t_1 = u_1^{c_1} h^{r_1}$ and $t_2 = u_2^{c_2} h^{r_2}$ and accepts if and only if all checks succeed.