Secure Hamming Distance Based Computation and Its Applications

Ayman Jarrous and Benny Pinkas[⋆]

University of Haifa

Abstract. This paper examines secure two-party computation of functions which depend only on the Hamming distance of the inputs of the two parties. We present efficient protocols for computing these functions. In particular, we present protocols which are secure in the sense of full simulatability against malicious adversaries.

We show different applications of this family of functions, including a protocol we call m-point-SPIR, which is an efficient variant of symmetric private information retrieval (SPIR). It can be used if the server's database contains N entries, at most $N/\log N$ of which have individual values, and the rest are set to some default value. This variant of PIR is unique since it can be based on the existence of OT alone.

1 Introduction

There are many known generic constructions of secure two-party and multi-party computation. It is preferable, of course, to use constructions which are secure against malicious adversaries, and where security is proved according to the full simulatability notion defined in [8]. In that case the composition theorem of [8] implies that the resulting protocol can be used as a building-block for more complex protocols, and security can be analyzed assuming that the building-block protocol is implemented by a trusted oracle [8,15]. There are recent efficient constructions of generic protocols which are secure according to this definition (by Lindell and Pinkas [22], and Jarecki and Shmatikov [20]), and there is even an implementation of the former protocol [23]. Our work investigates only the stand-alone setting, but there are also efficient generic constructions of secure two-party protocols in the UC model [19]. The downside of generic constructions is that they impose additional overheads, such as communicating and checking multiple copies of a circuit computing the functionality [22], or computing public key operations for every gate of the circuit [20]. It is therefore important to identify functionalities that are essential for many applications, and design efficient secure constructions of these specific functionalities. This paper performs this task for a functionality denoted as "Hamming distance based oblivious transfer", for which we also demonstrate different interesting applications.

[⋆] Supported by the the Israel Science Foundation (grant No. 860/06), the European Union under the FP7-STREP project CACE, and a European Research Council (ERC) Starting Grant.

The Hamming distance between two strings is defined as the number of characters in which they differ. We define "Hamming distance based oblivious transfer" (HDOT, pronounced "h-dot") as a protocol which allows two parties, a receiver $\mathcal{P}_1$ which has an input w, and a sender $\mathcal{P}_2$ which has an input w', to securely evaluate a function $f(\cdot,\cdot)$ whose output is determined only by the Hamming distance between w and w' (denoted $d_H(w,w')$). More precisely, the output is defined in the following way: Let $|w| = |w'| = \ell$, then $\mathcal{P}_2$ must provide $\ell + 1$ additional inputs $Z_0, \ldots, Z_\ell$, and $\mathcal{P}_1$'s output is set to be Z_d where $d = d_H(w,w')$. With regards to this functionality, this paper contains the following results:

- HDOT protocols secure against semi-honest adversaries:
  - A protocol denoted *bin*HDOT for *binary* inputs $w, w' \in \{0,1\}^\ell$. This protocol operates by computing $O(\ell)$ homomorphic encryptions and only $\log \ell$ invocations of 1-out-of-2 oblivious transfer.
  - A general HDOT protocol, for $w, w' \in \Sigma^\ell$, where Σ can be arbitrary. This protocol uses *bin*HDOT as a building block.
- A *bin*HDOT protocol secure against malicious adversaries (in the stand-alone setting). The protocol uses two primitives that must also be secure against malicious adversaries: Committed Oblivious Transfer with Constant Difference (COTCD), and Oblivious Polynomial Evaluation (OPE). We give a construction for the first primitive, which is an example of a new class of OT protocols, *constrained OT*, which we define. The latter primitive is based on a construction of Hazay and Lindell [17].
- Applications of HDOT. These include several straightforward applications, such as computing the Hamming distance between strings, or transferring one of two words based on whether the two input strings are equal or not (a functionality we denote as *EQ*, for *equality based transfer*). Another application is a variant of symmetric PIR (SPIR) which we denote as m-point-SPIR, and which can be used when the server's database contains N items, of which at most $m = o(N/\log N)$ are unique and the other $N - m$ items have some default value. The receiver does not know whether it learns a unique or a default value. We show a protocol which is based on HDOT and can be reduced to oblivious transfer alone, which computes this functionality more efficiently than known PIR protocols. m-point-SPIR can be used for other applications, as described in Section 6.

2 Preliminaries

We use the standard definitions of secure two-party computation in the stand-alone setting (see Goldreich's book [15, Chapter 7]). Security of protocols is analyzed by comparing what an adversary can do in a real execution of the protocol to what it can do in an ideal scenario that is secure by definition. The ideal scenario involves an incorruptible trusted third party (TTP) which receives the inputs of the parties, computes the desired functionality, and returns to each party its respective output. A protocol is secure if any adversary which participates in the real protocol (where no trusted third party exists) can do no more harm than if it was involved in the above-described ideal computation. The exact definition appears in [15].

The hybrid model. Our protocols use other secure protocols, such as oblivious transfer, as subprotocols. It has been shown in [8] that if the subprotocols are secure according to the right definition (i.e., full simulatability in the case of the malicious adversary scenario), it suffices to analyze the security of the main protocol in a hybrid model. In this model the parties interact with each other and have access to a trusted party that computes for them the functionalities that are implemented by the subprotocols. The composition theorem states that it is not required to analyze the execution in the real model, but rather only compare the execution in the hybrid model to that in the ideal model.

2.1 Cryptographic Primitives and Tools

Homomorphic Encryption. A homomorphic encryption scheme allows to perform certain algebraic operations on an encrypted plaintext by applying an efficient operation to the corresponding ciphertext. In addition, we require in this paper that the encryption scheme be *semantically secure*. In particular, we use an additively homomorphic encryption schemes where the message space is a ring (or a field). There therefore exists an algorithm $+_{pk}$ whose input is the public key of the encryption scheme and two ciphertexts, and whose output is $E_{pk}(m_1 + m_2) = E_{pk}(m_1) +_{pk} E_{pk}(m_2)$. (Namely, given the public key alone this algorithm computes the encryption of the sum of the plaintexts of two ciphertexts.) The new ciphertext is an encryption which is done with fresh and independent randomness. There is also an efficient algorithm $\cdot_{pk}$, whose input consists of the public key of the encryption scheme, a ciphertext, and a constant c in the field, and whose output is $E_{pk}(c \cdot m) = c \cdot_{pk} E_{pk}(m)$.

An efficient implementation of an additive homomorphic encryption scheme with semantic security was given by Paillier [30,31]. In this cryptosystem the encryption of a plaintext from $[0, N-1]$, where N is an RSA modulus, requires two exponentiations modulo N^2. Decryption requires a single exponentiation. Security is based on the decisional composite residuosity (DCR) assumption.

Oblivious Transfer. The paper uses 1-out-of-N oblivious transfer (OT_1^N) as a basic building block. The OT_1^N protocol runs between two parties, a sender that has an input $(X_0, X_1, \ldots, X_{N-1})$, where $X_i \in \{0,1\}^m$, and a receiver that has an input $I \in \{0, 1, \ldots, N-1\}$. By the end of the protocol, the receiver learns X_I and nothing else and the sender does not learn any information about I. In [29] it was shown how to implement OT_1^N using $\log N$ invocations of OT_1^2. There are many efficient implementations of OT_1^2, starting with a protocol of Even, Goldreich and Lempel [10]. Most of these protocols are designed for the semi-honest scenario, or for a malicious scenario where the protocol provides only the privacy property and not full simulatability. We note that while our protocol for the semi-honest scenario can use any OT protocol, the protocol for the malicious adversary scenario must use an OT protocol which is secure in the sense of full simulatability against malicious adversaries. Such protocols were described, e.g., in [6,16,32,17]. (We specifically need a committed OT variant where we can also prove a relation between the inputs of the sender, and therefore we use a protocol which builds on the work of Jarecki and Shmatikov [20].) We also note that in the malicious case we use OT_1^2 and not OT_1^N.

2.2 Related Work

Generic secure computation. Generic protocols (e.g., of [35]) can be used to compute any function. They are typically based on representing the computed function as a binary or an algebraic circuit, and applying the protocol to this representation. The overhead of these protocols depends on the size of the circuit representation of the functions. There are many theoretical constructions of secure generic protocols. Notable examples of *implementations* of secure computation are the Fairplay system [24] for secure two-party computation, and the FairplayMP and SIMAP systems [1,3] for secure multi-party computation. The system described in [23] implements fully simulatable secure two-party computation according to the recent construction of [22].

Computing the Hamming distance. Protocols for computing the scalar product of vectors (which is equal to the Hamming distance if the alphabet is binary) were suggested in [34,14]. These protocols are based on the use of homomorphic encryption, and are only secure against semi-honest adversaries. (Our HDOT protocol for binary alphabets and semi-honest adversaries borrows its first step from these protocols.)

A protocol for secure efficient *approximate* computation of the Hamming distance, with a polylogarithmic communication overhead, was suggested in [18] (previous protocols for this task use $O(\sqrt{\ell})$ communication for ℓ-bit words [12,13]). We wanted to improve upon these protocols for three reasons: (1) These protocols introduce approximation errors. (2) The protocols are only secure against semi-honest adversaries. (3) In addition, these protocols have good asymptotic communication overhead, but use non-trivial components which seem difficult to implement with a performance that will be competitive for reasonable input sizes[1].

3 Hamming Distance Based Oblivious Transfer

A Hamming Distance based Oblivious Transfer protocol (abbrev. HDOT) is run between two parties, a receiver ($\mathcal{P}_1$) and a sender ($\mathcal{P}_2$). It is defined as follows:

– *Input:* $\mathcal{P}_1$'s input is a word $w \in \Sigma^\ell$. $\mathcal{P}_2$'s input contains a word $w' \in \Sigma^\ell$, and $\ell + 1$ values $Z_0, \ldots, Z_\ell$.
– *Output:* $\mathcal{P}_1$'s output is Z_d, where $d = d_H(w, w')$ is the Hamming distance between w and w' (note that $\mathcal{P}_1$ does not learn the Hamming distance itself). $\mathcal{P}_2$ has no output.

[1] For example, the protocol in [18] applies the Naor-Nissim [27] protocol to a circuit which computes vector operations over the Real numbers and samples from a Bernoulli distribution; in addition it uses symmetric PIR protocols.

The paper describes a special protocol, *bin*HDOT, for the case that the input words are binary (i.e., $\Sigma = \{0, 1\}$), and a general protocol which works for alphabets Σ of arbitrary size.

3.1 Straightforward Applications

An HDOT protocol can be immediately used for computing any function which depends on the Hamming distance. Following are some interesting examples of such functions:

- The *Hamming distance* itself can be computed by setting $Z_i = i$ for every $0 \leq i \leq \ell$.
- The *parity* of the exclusive-or of the two inputs is computed by setting Z_i to be equal to the least significant bit of i, for $0 \leq i \leq \ell$.
- *EQ – Equality based transfer, or* $EQ_{\mathcal{V}_0,\mathcal{V}_1}(w, w')$*:* This functionality outputs $\mathcal{V}_0$ if $w = w'$, and $\mathcal{V}_1$ otherwise. The functionality is computed by setting $Z_0 = \mathcal{V}_0$ and $Z_i = \mathcal{V}_1$ for $1 \leq i \leq \ell$, and executing an HDOT protocol. $\mathcal{P}_1$ does not know which of the two cases happens (namely, whether $w = w'$). This is crucial for the applications that are described below.
  Recall that it is easy to design a protocol in which $\mathcal{P}_1$ learns a specific value $\mathcal{V}_0$ if the two inputs are equal, and a *random* value otherwise. (See [11], or consider a protocol where $\mathcal{P}_1$ sends a homomorphic encryption $E(w)$, and receives back $E(r \cdot (w - w') + \mathcal{V}_0)$, where r is a random value.) Our protocol is unique in defining a specific value to be learned if the two inputs are different, and in hiding whether the inputs are equal or not.[2]
- *Threshold HDOT protocol:* The equality based transfer protocol can be generalized to tolerate some errors and have the output be $\mathcal{V}_0$ if the Hamming distance is smaller than a threshold τ, and be $\mathcal{V}_1$ otherwise. In other words, it implements the following functionality:

  $$\mathrm{HDOT}^{\tau}_{\mathcal{V}_0|\mathcal{V}_1}(w, w') = \begin{cases} \mathcal{V}_0, \, d_H(w, w') < \tau \\ \mathcal{V}_1, \, d_H(w, w') \geq \tau \end{cases}$$

  This functionality is implemented by setting $Z_0 = \cdots = Z_{\tau-1} = \mathcal{V}_0$, and $Z_\tau = \cdots = Z_\ell = \mathcal{V}_1$.

The protocol for equality based transfer is the major building blocks of the m-point-SPIR SPIR application described in Section 6.

4 Protocols Secure against Semi-honest Adversaries

We first describe protocols which are secure against semi-honest behavior of the potential adversaries. These protocols are relatively simple, yet they are unique in invoking oblivious transfer a number of times which is only logarithmic in the input length. The malicious adversary scenario is covered in Section 5.

[2] In [2] it was shown how to implement a protocol which transfers one of two strings if $w > w'$, and transfers the other string if $w < w'$ (if $w = w'$ the output is random). It is possible to compute the *EQ* functionality by combining that protocol with a protocol which outputs a specific value if $w = w'$ and a random value otherwise.

4.1 A Protocol for Binary Alphabets (*bin*HDOT)

Consider first the case where the alphabet is binary ($\Sigma = \{0, 1\}$). The *bin*HDOT functionality can be securely implemented by applying Yao's protocol to a circuit computing it. That solution would require running ℓ invocations of OT^2_1. We describe here a protocol which accomplishes this task using only $\log(\ell+1)$ OT^2_1s (see below a comparison of the performance of these two protocols).

The protocol works in the following way: In the first step the parties use homomorphic encryption to count the number of bits in which the two words differ. The result is in the range $[0, \ell]$. Next, the two parties use $\mathrm{OT}^{\ell+1}_1$ (implemented using $\log(\ell + 1)$ OT^2_1s) to map the result to the appropriate output value. The protocol is described in detail in Figure 1.

Correctness. The value d_H is equal to the Hamming distance. In Step 4, $\mathcal{P}_1$ computes (in $\mathcal{F}$) the value $d_H + r$, which can have one of $\ell + 1$ values (namely $r, r + 1, \ldots, r + \ell$). It holds with probability $1 - \ell/|\mathcal{F}|$ that $r < |\mathcal{F}| - \ell$. (And

$bin\mathrm{HDOT}_{\langle Z_0,\ldots,Z_\ell\rangle}(w, w')$ PROTOCOL

INPUT: $\mathcal{P}_1$'s input is a word $w = (w_0, \ldots, w_{\ell-1})$, $\mathcal{P}_2$'s input is $w' = (w'_0, \ldots, w'_{\ell-1})$, where $w_i, w'_i \in \{0, 1\}$. $\mathcal{P}_2$ has additional inputs $(Z_0, \ldots, Z_\ell)$.
OUTPUT: $\mathcal{P}_1$ receives Z_i such that $d_H(w, w') = i$. $\mathcal{P}_2$ learns nothing.
The protocol uses $E_{pk}(\cdot)$, a homomorphic encryption function. The plaintexts are in a ring or a field $\mathcal{F}$. (We emphasize that ℓ and $|\Sigma|$ are negligible compared to $|\mathcal{F}|$. A typical size could be $|\mathcal{F}| = 2^{1024}$.) *pk* is a public key that both parties know, but only $\mathcal{P}_1$ knows the corresponding private key and can decrypt messages.

1. $\mathcal{P}_1$ sends the homomorphic encryption of each bit of the binary representation of $w = \{w_0, \ldots, w_{\ell-1}\}$, where $w_i \in \{0, 1\}$.
2. $\mathcal{P}_2$ receives the encrypted representation $\{E_{pk}(w_0), \ldots, E_{pk}(w_{\ell-1})\}$. For each bit location j it calculates $E_{pk}(\vartheta_j)$, where $\vartheta_j \in \{0, 1\}$ and is equal to 1 if, and only if, $w_j \neq w'_j$. The calculation is done in the following way:
$$E_{pk}(\vartheta_j) = E_{pk}(w_j) \cdot_{pk} (1 - w'_j) +_{pk} (1 -_{pk} E_{pk}(w_j)) \cdot_{pk} w'_j$$
3. Using the homomorphic properties, $\mathcal{P}_2$ sums the results of the previous step and computes $E_{pk}(d_H) = \sum_0^{\ell-1} E_{pk}(\vartheta_i)$. The value d_H is in the range $\{0, 1, \ldots, \ell\}$ and is equal to the Hamming distance between the two input words. In addition, $\mathcal{P}_2$ chooses a random value $r \in \mathcal{F}$, computes the value $E_{pk}(d_H + r)$, and sends it to $\mathcal{P}_1$. (In other words, it shifts the result by a random value r. Note that with overwhelming probability, $1 - \ell/|\mathcal{F}|$, this addition operation does not involve a modular reduction.)
4. $\mathcal{P}_1$ receives $E_{pk}(d_H + r)$ and decrypts the result.
5. Next, the parties map the result to the appropriate Z_i value, by invoking a $\mathrm{OT}^{\ell+1}_1$ protocol where $\mathcal{P}_1$ is the receiver and $\mathcal{P}_2$ is the sender:
   - The input of $\mathcal{P}_1$ is $(d_H + r)\ mod\ (\ell + 1)$.
   - $\mathcal{P}_2$ has inputs $X_0, \ldots, X_\ell$, where $X_i = Z_{(i-r)\ mod\ (\ell+1)}$ (namely, Z_i is mapped to input $(i + r)\ mod\ (\ell + 1)$ of the OT).

   $\mathcal{P}_1$'s output in the OT is its output in the *bin*HDOT protocol.

Fig. 1. The *bin*HDOT protocol

since $|\mathcal{F}|$ is typically very large compared to ℓ, e.g. $|\mathcal{F}| \approx 2^{1024}$ and $\ell < 1000$, we do not consider here the negligible probability that this event does not happen.) Therefore, the computation of $d_H + r$ in $\mathcal{F}$ does not involve a modular reduction and has the same result as adding them over the integers. Reducing the result modulo $\ell + 1$ (in Step 5) is therefore equal to $(r + d_H) \bmod (\ell + 1)$. $\mathcal{P}_1$ uses this result as its input to the 1-out-of-$(\ell+1)$ OT protocol of Step 5. $\mathcal{P}_2$, on the other hand, sets the sender's inputs in the OT such that each Z_i value is the sender's input indexed by $(r + i) \bmod (\ell + 1)$. As a result, the output of $\mathcal{P}_1$ in the OT protocol is Z_{d_H}, as required.

Note that if the parties are only interested in computing the value of the Hamming distance then the protocol can be greatly simplified: $\mathcal{P}_2$ should send to $\mathcal{P}_1$ in Step 3 the encryption $E_{pk}(d_H)$. There is no need to run Steps 4 and 5.

Improving the initial step using non-interactive preprocessing. An additional improvement can be achieved in the first step of the protocol, where $\mathcal{P}_1$ sends an encrypted binary representation of the word. This representation can be precomputed using *non-interactive* preprocessing: $\mathcal{P}_1$ can prepare in advance ℓ encrypted zeros and ℓ encrypted ones, instead of encrypting the input bits online. This preprocessing enables $\mathcal{P}_1$ to send the binary representation directly without spending time online encrypting 0 and 1 values.

Overhead. We compare the overhead of the *bin*HDOT protocol to that of applying Yao's protocol to a circuit computing the same functionality. We note that the runtime of an OT protocol is slower than that of a homomorphic encryption or decryption, and that the runtime of these latter operations is *much* slower than that of a homomorphic addition or a homomorphic multiplication by a constant (which in turn is *much* slower than symmetric encryption or decryption). This relation between run times can be summarized as follows (where $>$ denotes "slower", and $\gg$ denotes slower by an order of magnitude):

$$\text{OT} > \text{homomorphic enc.} \gg \text{homomorphic addition} \gg \text{symmetric enc.}$$

Without using any preprocessing, the *bin*HDOT protocol requires $\mathcal{P}_1$ to compute ℓ encryptions and a single decryption, while $\mathcal{P}_2$ computes $\ell+1$ homomorphic additions, and the two parties run $\log(\ell+1)$ OT_1^2s and $2(\ell+1)$ symmetric encryptions (in order to implement $\text{OT}_1^{\ell+1}$). In Yao's protocol, the parties compute a circuit with ℓ input bits and a total of $O(\ell)$ gates. This requires ℓ executions of an OT_1^2 protocol and $O(\ell)$ symmetric encryptions and decryptions. Both protocols require $O(\ell)$ communication.

The improvement achieved by the *bin*HDOT protocol is noticeable since it reduces the number of OTs, which are the most time consuming operation, from ℓ to $\log(\ell+1)$. In addition, the *bin*HDOT protocol can benefit from the use of *non-interactive* preprocessing to precompute all homomorphic encryption operations even before the parties know of each other. In that case the ℓ encryptions done by $\mathcal{P}_1$ are computed offline, and its online computation is composed of a single decryption and $\log(\ell+1)$ OTs. (Yao's protocol cannot precompute the oblivious transfers without using interaction. We note that if interactive preprocessing is possible, then the OTs themselves can be precomputed in both protocols, and this reduces the overhead of both protocols.)

Security. (sketch) We analyze security assuming that the parties are semi-honest. The proof is simple, and therefore we only give a sketch of the proof: We assume that the OT protocol is secure, and therefore we can prove security in a hybrid model where the OT protocol is implemented by an oracle. In the protocol, $\mathcal{P}_2$ receives homomorphic encryptions of a binary representation of a word, and then it plays the role of the sender in an OT protocol in which it receives no output. Therefore, if $\mathcal{P}_2$ learns anything this information must have leaked from the encryptions it received. In other words, it is easy to write a reduction showing that any algorithm that $\mathcal{P}_2$ might use to learn information can be used to break the security of the semantic security of the encryption. $\mathcal{P}_1$ receives from $\mathcal{P}_2$ a *random* value $(d_H + r)$. It then participates in the OT protocol, which we assume to be implemented by an oracle. $\mathcal{P}_2$ therefore learns nothing but the output of the OT, which is its designated output of the protocol.

4.2 A Protocol for Arbitrary Alphabets (HDOT)

We now describe an HDOT protocol which works over arbitrary alphabets Σ. The protocol is based on applying the *bin*HDOT protocol to every character of the words. More specifically, the parties have inputs $w, w' \in \Sigma^\ell$, respectively. The protocol begins with the parties representing each of the letters of Σ as a binary word of length $\lceil \log |\Sigma| \rceil$, and then running (for each letter location) the equality based transfer (EQ) protocol, which was defined above and is an application of *bin*HDOT. In each execution of the EQ protocol $\mathcal{P}_1$ learns a value α_i if $w_i = w'_i$, or the value $\alpha_i + 1$ otherwise, where α_i is chosen at random by $\mathcal{P}_2$. Then, $\mathcal{P}_1$ sums the values that it has received modulo $\ell + 1$. The result is equal, modulo $\ell + 1$, to $\sum \alpha_i$ plus the Hamming distance of the original words. The parties then run an $\mathrm{OT}_1^{\ell+1}$ protocol to map the result to the desired output. The protocol is detailed in Figure 2.

Correctness. For every $0 \le i \le \ell - 1$, $\mathcal{P}_1$ and $\mathcal{P}_2$ learn in Step 1 values β_i, α_i, respectively, such that $\beta_i = \alpha_i$ if the letters w_i and w'_i are equal, and $\beta_i = \alpha_i + 1$ if the letters are different. Let $S_\alpha = \sum_{i=0}^{\ell-1} \alpha_i$, where here the addition is done in $\mathcal{F}$. Define S_β similarly. Let d be the Hamming distance between the two input words. Then it holds with probability $1 - \ell/|\mathcal{F}|$ that $S_\beta = S_\alpha + d$, where the addition here is done over the integers. Therefore, the values $\sigma_\alpha = S_\alpha \bmod (\ell{+}1)$ and $\sigma_\beta = S_\alpha \bmod (\ell + 1)$ computed in Step 2 satisfy that $\sigma_\beta - \sigma_\alpha \bmod (\ell + 1)$ is equal to the Hamming distance d (which is in the range $[0, \ell]$).

Consider now the OT in Step 3. Assume first that $\sigma_\alpha = 0$. In this case $\mathcal{P}_1$'s input to the OT, σ_β, is equal to the Hamming distance, and the inputs of $\mathcal{P}_2$ to the OT are the values $Z_0, \ldots, Z_\ell$ (in that order). The OT protocol therefore computes the desired output in this case. Now, if $\sigma_\alpha > 0$ then $\mathcal{P}_1$'s input to the OT protocol is cyclically shifted (modulo $\ell + 1$) by σ_α, while the order of $\mathcal{P}_2$'s inputs to the OT is also cyclically shifted (modulo $\ell + 1$) by the same value σ_α. The OT protocol therefore computes the correct result.

HDOT$_{\langle Z_0,\ldots,Z_{\ell+1}\rangle}(w, w')$ PROTOCOL

INPUT: $\mathcal{P}_1$ has an input $w = \langle w_0, w_1, \ldots, w_{\ell-1}\rangle \in \Sigma^\ell$. $\mathcal{P}_2$ has an input $w' = \langle w'_0, w'_1, \ldots, w'_{\ell-1}\rangle \in \Sigma^\ell$, and additional input values $Z_0, \ldots, Z_\ell$. We denote by $\bar{w}_j$ the binary representation of w_j, which is $\lceil \log(|\Sigma|)\rceil$ bits long.
OUTPUT: $\mathcal{P}_1$ learns Z_i such that $d_H(w, w') = i$, $\mathcal{P}_2$ learns nothing.

1. For every $i \in [0, \ell-1]$, $\mathcal{P}_2$ chooses at random a value $\alpha_i \in_R \mathcal{F}$. Both parties then run the protocol EQ$_{\alpha_i,\alpha_i+1}(\bar{w}_i, \bar{w}_i{}')$. ($\bar{w}_i, \bar{w}_i{}'$ denote the binary representations of the letters w_i and w'_i, respectively. The output of this protocol is α_i if $w_i = w'_i$, and $\alpha_i + 1$ otherwise.)
   At the end of the process, $\mathcal{P}_1$ obtains the values $\{\beta_0, \ldots, \beta_{\ell-1}\}$, where
   $$\beta_i = \begin{cases} \alpha_i, & w_i = w'_i \\ \alpha_i + 1, & w_i \neq w'_i \end{cases}$$
2. $\mathcal{P}_1$ sums, modulo $\ell + 1$, the β_i values it received. Namely, it computes $\sigma_\beta = (\sum_0^{\ell-1} \beta_i) \bmod (\ell + 1)$. $\mathcal{P}_2$ sums its α values and computes $\sigma_\alpha = (\sum_0^{\ell-1} \alpha_i) \bmod (\ell + 1)$.
3. Both parties run an OT$_1^{\ell+1}$ protocol with the following inputs:
   - $\mathcal{P}_1$ is the receiver and its input is σ_β.
   - $\mathcal{P}_2$ is the sender and its input is $\{X_0, \ldots, \ldots, X_\ell\}$, where $X_i = Z_{(i-\sigma_\alpha)\ mod\ (\ell+1)}$.

   The value that $\mathcal{P}_1$ receives in the OT is defined as its output in the protocol.

Fig. 2. The HDOT protocol for general alphabets

Overhead. The overhead is that of applying the *bin*HDOT protocol ℓ times over $\log|\Sigma|$ long binary strings, and then running $\log(\ell+1)$ invocations of OT$_1^2$. The parties run $\ell \log\log|\Sigma| + \log(\ell + 1)$ OT$_1^2$s. (A direct implementation of this functionality using Yao's protocol would have required invoking $O(\ell \log|\Sigma|)$ OTs.)

Security. (sketch) Analyzing security in the hybrid model, we assume that the *bin*HDOT and OT protocols are executed by a trusted oracle. Then $\mathcal{P}_2$, being the sender in these protocols, cannot learn any information about the input of $\mathcal{P}_1$. $\mathcal{P}_1$ receives the β_i values in the first step, but it cannot distinguish whether $\beta_i = \alpha_i$ or $\beta_i = \alpha_i + 1$, since each α_i value was chosen randomly by $\mathcal{P}_2$. In the last step, $\mathcal{P}_1$ receives the result of mapping the sum of the β values to the appropriate Z_i value, which is also the result it would have received from the trusted party.

4.3 Weighted Hamming Distance Based OT

The *weighted* Hamming distance between two ℓ-letter strings w, w' is defined in the following way: The function depends on a set of integer weights $\omega_0, \ldots, \omega_{\ell-1}$. We define δ_i, for $0 \leq i \leq \ell - 1$, to be 0 if $w_i = w'_i$, and 1 otherwise. The weighted Hamming distance is $\sum_{i=0}^{\ell-1} \delta_i \omega_i$ (earlier we handled the case where

$\forall i \;\; \omega_i = 1$). This function enables to assign to any letter location a specific weight corresponding to its importance.

It is possible to slightly change the HDOT protocols to support the computation of a weighted Hamming distance based OT. In the binary alphabet case, the revised *bin*HDOT protocol computes in Step 2 the values $E_{pk}(\vartheta_j \omega_j)$ by multiplying $E_{pk}(\vartheta_j)$ by ω_i. The value d_H is defined to be the sum of these values. Let $\Omega = \sum_{i=0}^{\ell-1} \omega_i$. The value of d_H is in the range $[0, \Omega]$. Therefore $\mathcal{P}_2$ has inputs $Z_0, \ldots, Z_\Omega$, and the last step of the protocol computes a 1-out-of-($\Omega + 1$) OT. In the case of an arbitrary alphabet, each β_i value is set to $\alpha_i + \omega_i$ if the two letters are different, and to α_i is they are equal. Again, the last step computes a 1-out-of-($\Omega + 1$) OT.

5 A *bin*HDOT Protocol for Malicious Adversaries

We design a new *bin*HDOT protocol to handle the presence of malicious adversaries. In this protocol the parties use a new variant of OT_1^2 to learn whether corresponding bits of the two words are equal, and then use an Oblivious Polynomial Evaluation (OPE) protocol [28,17] to map the result to an output value. (This is different than the semi-honest case, where homomorphic encryption was used to compare bits, and OT_1^N was used to compute the final result.) The new protocol uses OT and OPE protocols which are efficient and yet are secure in the sense of full simulatability against malicious adversaries. Security can therefore be analyzed in the hybrid model. In more detail, the protocol uses the following tools:

Committed 1-out-of-2 Oblivious Transfer with Constant Difference (or COTCD_1^2), secure against malicious adversaries. A committed OT protocol in an OT protocol where the parties commit to their inputs: the sender commits to its inputs m_0, m_1 and the receiver commits to its input $\sigma \in \{0, 1\}$. During the protocol each party can verify that the other party's input is equal to the corresponding committed value. We define a committed OT with constant difference (COTCD, pronounced "cot-cd") to be a committed OT with an additional auxiliary input composed of a value Δ known to the sender, and a commitment to Δ which is known to the receiver. The protocol lets the receiver verify that the difference of the two inputs of the sender is $\pm\Delta$. In other words, it either holds that $m_1 - m_0 = \Delta$ or that $m_0 - m_1 = \Delta$.

We use a COTCD primitive which is based on the Jarecki and Shmatikov (JS) committed OT protocol [20], which is in turn based on the Camenisch-Shoup (CS) encryption scheme [7]. The details of the COTCD protocol are described in the full version of our paper.[3] We use that protocol since it can be used to transfer

[3] The COTCD protocol is identical to the Jarecki and Shmatikov (JS) protocol [20], with an addition of a preliminary step and a verification step. In the preliminary step, both parties receive their auxiliary inputs: the sender receives a value Δ, which is the difference that must hold between its input values, and the receiver receives the committed value of Δ. In the verification step the sender proves to the receiver in zero-knowledge that the committed values, m_0, m_1, have a difference $\pm\Delta$. It is important to note that the receiver knows only $\mathsf{Com}(\Delta)$ and does not learn Δ.

strings, and since it is easy to add to it an efficient zero-knowledge proof that the messages of the sender have the required difference (it seems much harder to add a proof of this type to other OT protocols which are secure against malicious adversaries, such as the protocols of [17,32]). The JS protocol is UC-secure in the common reference string model and therefore all invocations of that protocol can be run in parallel (as a result, the HDOT protocol we construct can execute in parallel all ℓ invocations of the COTCD protocol). The protocol is proved to be secure under the decisional composite residuosity (DCR) assumption (i.e., the assumption on which the Paillier homomorphic encryption system is based).

Commitment scheme. The Camenisch-Shoup (CS) encryption scheme [7] is used in our protocol as a commitment scheme, as is suggested in [20].

An Oblivious Polynomial Evaluation (OPE) protocol secure against malicious adversaries. An OPE protocol [28] is a protocol where the sender's input is a polynomial $P()$ of a certain degree, and the receiver's input is a value x. The receiver's output is $P(x)$ while the sender learns nothing. We use the OPE construction of Hazay and Lindel [17], which is secure (in the sense of full simulatability) against malicious adversaries, and uses very few exponentiations.

The underlying fields. The output of the COTCD protocol is used as an input of the OPE protocol. The COTCD protocol runs in a group $\mathcal{F} = \mathbb{Z}^*_{n^2}$, where $\mathbb{Z}^*_{n^2}$ is defined by a safe RSA modulus $n = pq$. The encryption scheme of Camenisch and Shoup, which is used in the protocol as a commitment scheme, works in the same group. The OPE protocol of [17] runs in $\mathbb{Z}_N$, with N being an RSA modulus. Our protocol must enable the parties to use the result of the COTCD protocol as an input to the OPE protocol. It must therefore use a group $\mathbb{Z}^*_{n^2}$ and a field $\mathbb{Z}_N$, which satisfy that $|\mathbb{Z}^*_{n^2}| < |\mathbb{Z}_N|$, and therefore we will require that $n^2 < N$. We define a simple mapping $f : \mathbb{Z}^*_{n^2} \to \mathbb{Z}_N$, where the only requirement is that no two elements of $\mathbb{Z}^*_{n^2}$ are mapped by f to the same value in $\mathbb{Z}_N$. The protocol then performs the initial computations in $\mathbb{Z}^*_{n^2}$ and then uses f to map the result to $\mathbb{Z}_N$.

The protocol itself is described in Figure 3. In the protocol, for every bit location i $\mathcal{P}_1$ receives a value t_i^0 if the corresponding bits are equal, and the value $t_i^0 + \Delta$ otherwise. The value Δ, and also all t_i^0 values, are randomly chosen by $\mathcal{P}_2$. (In the semi-honest case $\mathcal{P}_1$ learned one of two values whose difference was 1. Here the difference is a random number Δ in order to prevent attacks by a malicious $\mathcal{P}_1$.) $\mathcal{P}_1$ then sums the values it received, and obtains the result $\sum_{i=1}^{\ell} t_i^0 + d \cdot \Delta$, where d is the Hamming distance. We use the notation $\sigma_r = \sum_{i=1}^{\ell} t_i^0$. $\mathcal{P}_2$ then prepares an OPE where $\forall j \in [0, \ell]$, $P(\sigma_r + j \cdot \Delta) = Z_j$. The parties execute an OPE and $\mathcal{P}_1$ computes $P(\sigma_r + d\Delta)$ and learns the desired result.

The protocol uses an OPE instead of $\text{OT}_1^{\ell+1}$ since the values are mapped to locations in a large range, rather than to indices in the range $[0, \ell]$, in order to prevent a malicious $\mathcal{P}_1$ from learning any Z_i value which does not correspond to the actual Hamming distance. If $\mathcal{P}_1$ evaluates the polynomial at any point other than intended, it is likely to receive a random answer since it does not know Δ and is therefore unlikely to choose any point corresponding to a Z_i value. As for

INPUT: $\mathcal{P}_1$'s input is a word $w = (w_0, \dots, w_{\ell-1})$, $\mathcal{P}_2$'s input is $w' = (w'_0, \dots, w'_{\ell-1})$, where $w_i, w'_i \in \{0, 1\}$. $\mathcal{P}_2$ has additional inputs $(Z_0, \dots, Z_\ell)$.
OUTPUT: $\mathcal{P}_1$ receives Z_i such that $d_H(w, w') = i$ (i.e. the Hamming distance of w and w' is i). $\mathcal{P}_2$ learns nothing.

1. $\mathcal{P}_2$ chooses at random $\Delta \in_R \mathbb{Z}^*_{n^2}$ and sends to $\mathcal{P}_1$ a commitment to Δ. In addition it proves to $\mathcal{P}_1$, using a zero-knowledge proof of knowledge, the knowledge of Δ.
2. For each pair of bits (w_i, w'_i), both parties use COTCD to check whether the bits are equal:
   - $\mathcal{P}_2$ chooses a random value $t^0_i \in_R \mathcal{F}$, and defines $t^1_i = t^0_i + \Delta$.
   - Both parties run a COTCD protocol:
     (a) The auxiliary inputs to the protocol are Δ, known to $\mathcal{P}_2$, and a commitment to Δ, known to $\mathcal{P}_1$.
     (b) $\mathcal{P}_1$ is the receiver and its input is w_i.
     (c) $\mathcal{P}_2$ is the sender. If $w'_i = 0$ then it sets $(x^0_i, x^1_i) = (t^0_i, t^1_i)$; Otherwise, $(x^0_i, x^1_i) = (t^1_i, t^0_i)$.

   In each execution of the protocol, if both bits are equal then $\mathcal{P}_1$ learns t^0_i, otherwise, $\mathcal{P}_1$ learns t^1_i. (If $|x^1_i - x^0_i| \neq \Delta$ then $\mathcal{P}_1$ aborts.)
   By the end of this step, $\mathcal{P}_1$ learns $t^{b_0}_0, \dots, t^{b_{\ell-1}}_{\ell-1}$, where $b_i = w_i \oplus w'_i$, while $\mathcal{P}_2$ does not learn any information.
3. $\mathcal{P}_1$ computes $\sigma_t = \sum t^{b_i}_i$ and $\mathcal{P}_2$ computes $\sigma_r = \sum t^0_i$. These summations are done in $\mathbb{Z}^*_{n^2}$.
4. $\mathcal{P}_2$ constructs a polynomial $P(x) = \sum_0^\ell a_i x^i$ in $\mathbb{Z}_N$, such that $P(f(\sigma_r + i \cdot \Delta)) = Z_i, \forall i \in \{0, 1, \dots, \ell\}$ (where f is the simple mapping from $\mathbb{Z}^*_{n^2}$ to $\mathbb{Z}_N$), and $P(0)$ is random. (This construction succeeds if $0 \notin \{\sigma_r, \dots, \sigma_r + \ell\Delta\}$, which happens with probability $1 - (\ell + 1)/|\mathbb{Z}_N|$.) The degree of P is $\ell + 1$.
5. $\mathcal{P}_1$ and $\mathcal{P}_2$ run an OPE protocol to evaluate $P(f(\sigma_t))$, such that $\mathcal{P}_1$ learns the result while $\mathcal{P}_2$ does not learn any information.

Fig. 3. The *bin*HDOT protocol for the malicious case

a malicious $\mathcal{P}_2$, its inputs w' and $Z_0, \dots, Z_\ell$ can be extracted from its interaction with the OT and OPE protocols, and are used for a simulation based proof.

Theorem 1. *The protocol computes the binHDOT functionality.*

Proof. Let us follow the steps of the protocol. In each execution of the COTCD protocol, $\mathcal{P}_1$ learns t^0_i if both bits are equal, otherwise, it learns $t^1_i = t^0_i + \Delta$. In other words, it learns $t^{b_i}_i$, where $b_i = w_i \oplus w'_i$. Then, in Step 3, $\mathcal{P}_1$ computes $\sigma_t = t^{b_0}_0 + \dots + t^{b_{\ell-1}}_{\ell-1}$, and $\mathcal{P}_2$ computes $\sigma_r = t^0_0 + \cdots t^0_{\ell-1}$. Therefore it holds that $\sigma_t - \sigma_r = \Delta \cdot d_H(w, w')$. In Step 4, $\mathcal{P}_2$ constructs a polynomial $P(x)$ such that: $P(f(\sigma_r)) = Z_0$; $P(f(\sigma_r + \Delta)) = Z_1$;...; $P(f(\sigma_r + \ell \cdot \Delta)) = Z_\ell$. In the last step of the protocol, the parties use an OPE protocol to compute $P(f(\sigma_t)) = Z_{d_H(w,w')}$.

Theorem 2. *The protocol securely computes binHDOT in the presence of malicious adversaries.*

Proof. (Sketch) The security of the protocol is proved in the hybrid model, assuming that the COTCD and OPE primitives, as well as the zero-knowledge proof of knowledge of Δ used in the protocol, are performed by a trusted oracle (or trusted party). This assumption is justified since, as we detailed above, there are constructions of these primitives which have fully simulatable security against malicious adversaries (where the security is based on the Decisional Composite Residuosity (DCR) assumption).

We compare the execution of the protocol between $\mathcal{P}_1$ and $\mathcal{P}_2$ to an execution with a *trusted third party (TTP)*, where the TTP receives the inputs of both parties and computes the following functionality: If the input of $\mathcal{P}_1$ is w and the input of $\mathcal{P}_2$ is $\langle w', Z_0, \ldots, Z_\ell \rangle$, then the output of $\mathcal{P}_1$ is $Z_{d_H(w,w')}$. Otherwise if the input of $\mathcal{P}_1$ is a special symbol ρ then the output of $\mathcal{P}_1$ is a random value; otherwise if the input of either party is a special symbol $\perp$ then the protocol terminates prematurely.

We first prove security in the case that $\mathcal{P}_1$ is corrupt and then in the case that $\mathcal{P}_2$ is corrupt.

$\mathcal{P}_1$ is corrupt. The full proof appears in the full version of the paper. The idea behind the proof is that $\mathcal{P}_1$'s choices in the COTCD protocols define its input w. Then, $\mathcal{P}_1$ is supposed to add the values it received in the COTCD invocations and use the result as its input to the OPE. If it uses a different input to the OPE protocol, then, since it does not know Δ, it happens with overwhelming probability that $\mathcal{P}_1$ queries a value of the polynomial at a point which was not defined by $Z_0, \ldots, Z_\ell$ and receives a random answer.

$\mathcal{P}_2$ is corrupt. The full proof appears in the full version of the paper. The proof is based on the following ideas: (1) the simulator extracts the value of Δ from the zero-knowledge proof of knowledge; (2) the simulator then learns the inputs that $\mathcal{P}_2$ uses in the COTCD invocations, and based on these values the simulator computes w' and σ_r; (3) it also learns the coefficients of the polynomial $P()$ which is $\mathcal{P}_2$'s input to the OPE, and can therefore compute $Z_0 = P(\sigma_r), \ldots, Z_\ell = P(\sigma_r + \ell\Delta)$; (4) finally, the simulator sends $\langle w', Z_0, \ldots, Z_\ell \rangle$ to the TTP.

Efficiency. The overhead of the protocol is composed of running ℓ invocations of the COTCD protocol (which can be run in parallel, since the protocol is UC-secure), and a single invocation of the OPE protocol of [17]. Both of these protocol can be run in a constant number of rounds.

5.1 Securing the Applications against Malicious Adversaries

The protocol described above is secure against malicious behavior of either party. However, it does not enforce any structure of the inputs $Z_0, \ldots, Z_\ell$ of $\mathcal{P}_2$ and therefore a corrupt $\mathcal{P}_2$ can set these inputs to arbitrary values. This "feature" does not affect plain usage of the protocol, but it means that security against malicious adversaries cannot be guaranteed if the protocol is used for computing any functionality that requires specific relations between the Z_i values. Unfortunately, this is relevant to the relations required in the applications detailed in Section 3.1. For example, the EQ application, i.e., equality based transfer, requires that $Z_1 = Z_2 = \cdots = Z_\ell$. As a result, the protocol cannot be used "as is" as a building block for protocols (secure against malicious adversaries) for the HDOT functionality for arbitrary alphabets, or for the EQ functionality.

In order to adapt the protocol for these tasks, it is required to add zero-knowledge proofs which assure $\mathcal{P}_1$ that the Z_i inputs follow the desired structure. This is of course possible in principle, but in this work we have not examined how to optimize the efficiently of such proofs. We will only describe here the steps which are required in order to design and implement an EQ protocol secure against malicious adversaries (protocols for the other applications can be designed in a similar way): (1) The protocol needs an additional step where $\mathcal{P}_1$ obtains a commitment $\mathsf{Com}(\sigma_r)$ to $\sigma_r = \sum t_i^0$. This commitment can be computed given the commitments that $\mathcal{P}_2$ generates in the committed OT protocols; the correctness of the committed value can be proved using $\mathcal{P}_2$'s proofs about the Δ differences of its input pairs. (Namely, $\mathcal{P}_2$ must prove that there exist bits $b_0, \ldots, b_{\ell-1}$ such that $\sum x_i^{b_i} = \sigma_r$, and that $\forall i \; x_i^1 = x_i^0 + \Delta$.) (2) The parties need to use a "committed OPE" protocol, where $\mathcal{P}_2$ commits to the coefficients of its polynomial (such a protocol has not been described yet, but it is not hard to imagine how to implement it using techniques similar to those used for committed OT). (3) $\mathcal{P}_2$ must prove that there are values s, d such that s is committed to in $\mathsf{Com}(\sigma_r)$, d is committed to in $\mathsf{Com}(\Delta)$, and it holds that $P(s+d) = P(s+2d) = \cdots = P(s+\ell d)$. The main challenge in designing this step is that $P(s+d)$ is computed to by multiplying the committed coefficients of P by powers of the value $s+d$. Namely, the proof is about the sum of multiplications of committed values.

6 *m*-Point SPIR

Another application of the HDOT protocol is a new variant of symmetric private information retrieval (SPIR – Symmetric PIR) which we denote as m-point-SPIR. For a definition and discussion of single server PIR and symmetric PIR, see, e.g. [21,5]. In short, a PIR protocol involves a server with a database of N items $x_0, \ldots, x_{N-1}$ and a client who is interested in learning entry x_i of the database. This must be accomplished with $o(N)$ communication, without revealing i to the server, and (in the case of *symmetric* PIR) without revealing to the client anything but x_i.

The m-point-SPIR protocol that we define can be applied if at most m of the items of the server's database have specific values, and all other items have some default value $\bar{x}$. The client must not know whether the value it learns is the default value $\bar{x}$ or one of the unique values. We describe below a couple of applications of m-point-SPIR. The m-point-SPIR functionality is similar to a simpler functionality, where the client learns a *random* value if its input does not match any of the m indices which have specific values. The latter functionality is much simpler to implement (using OPE), as we detail below.

We show a protocol which implements m-point-SPIR with $O(m \log N)$ communication and $O(m \log N)$ computation (the smaller m is, the more efficient the protocol is). Therefore the communication is $o(N)$ as long as $m = o(N/\log N)$.

Another nice property of the m-point-SPIR protocol if that it can be implemented based on the existence of oblivious transfer alone. This property is not known for general SPIR protocols. (Furthermore, it is known that there cannot exist any transparent black-box reduction of PIR to OT [25].)

The m-point-SPIR functionality is defined in the following way. The server has inputs $0 \le p_1, \ldots, p_m \le N-1$, which are all distinct, and additional values $\bar{x}, x_{p_1}, \ldots, x_{p_m}$. The client has an input $0 \le i \le N-1$. The output of the client is x_{p_j} if there is an index $1 \le j \le m$ such that $i = p_j$, or $\bar{x}$ if no such p_j exists.

1-point SPIR. The implementation of 1-point-SPIR is straightforward given our previous protocols. The parties simply execute the protocol $EQ_{x_{p_1},\bar{x}}(i, p_1)$, whose output is x_{p_1} if $i = p_1$, and $\bar{x}$ otherwise. (The EQ protocol is defined in Section 3.1.) The communication overhead is of the order of the length of the index i, namely $O(\log N)$, times the length of the security parameter (i.e., the length of the homomorphic encryption). (This is under the reasonable assumption that the length of the database values (the x values) is in the order of the length of the security parameter; otherwise the communication is $O(\log N \cdot |x|)$.) The computation overhead is $O(\log N)$, and it is composed of $O(\log N)$ homomorphic encryptions and $O(\log \log N)$ OTs.

m-point-SPIR. For the general case of m-point-SPIR, the server first defines m random values $z'_1, \ldots, z'_m$ under the constraint that their exclusive-or is $\bar{x}$. It then defines values $z_1, z_2, \ldots, z_m$ satisfying the constraints $z_1 \oplus z'_2 \oplus \cdots \oplus z'_m = x_1$, $z'_1 \oplus z_2 \oplus z'_3 \oplus \cdots \oplus z'_m = x_2$, up to $z'_1 \oplus \cdots \oplus z'_{m-1} \oplus z_m = x_m$. The parties execute the protocols $EQ_{z_1,z'_1}(i, p_1)$, $EQ_{z_2,z'_2}(i, p_2)$, up to $EQ_{z_m,z'_m}(i, p_m)$. The client then computes the exclusive-or of the m values that it learned in these protocols.

Correctness follows from the fact that if there exists a j coordinate for which $i = p_j$ then the client learns a single z_j value. Otherwise $i \ne p_1, \ldots, p_m$ and the client learns only z'_j values. Therefore the exclusive-or of all the values that the client receives is equal to x_j in the former case, or to $\bar{x}$ in the latter case.

It is easy to verify the security of this protocol (assuming that the parties are semi-honest). Note that the client always performs the same operations and does not recognize whether it learned the value $\bar{x}$ or one of the m special values. The communication overhead is $O(m \log N)$ times the length of the security parameter, and the computation overhead is also $O(m \log N)$. This is therefore a SPIR protocol (with $o(N)$ communication) as long as $m = o(N/\log N)$, and in that case the computation overhead is also $o(N)$. (A “traditional” PIR protocol will have $O(N)$ computation overhead, since it must also process the entries with the default value.)

Basing m-point-SPIR on OT. The EQ protocol (which is essentially the HDOT protocol) is based on using a homomorphic encryption scheme and an oblivious transfer. However, it is easy to see that the usage of homomorphic encryption can be replaced with the usage of oblivious transfer alone (as is done in the HDOT protocol for the malicious case). As a result, m-point-SPIR can be based oblivious transfer alone.

Comparison to other protocols. Our m-point-SPIR protocol can be compared to oblivious polynomial evaluation (OPE), in which the server has an $(m-1)$-degree polynomial P, defined over a field of size at least N, and where the polynomial satisfies $P(p_j) = x_j$ for all $j \in [1, m]$. The client has input $0 \leq j \leq N-1$ and it obliviously computes $P(j)$. The OPE protocol has communication and computation overheads of $O(m)$ field operations, but it has the drawback that for inputs not in $p_1, \ldots, p_m$ the client receives a random output rather than a specific value $\bar{x}$.

The m-point-SPIR protocol can also be compared to PIR protocols of the type of the protocol of Cachin, Micali and Stadler [5] (that protocol is based on the ϕ-hiding assumption rather on general assumptions). These protocols, too, have the property that the server's work depends on the number of items in its database that have non-default values. Namely, it is $O(m)$ if the server has m items in its database, even if the range of the client's input is $[1, N]$. Still, in those protocols the sender is not able to set a "default" value $\bar{x}$ to be returned for all other $N-m$ values of the client's input. Finally, the m-point-SPIR functionality can be implemented using Yao's generic protocol and a circuit of size $O(m \log N)$, and $m \log N$ invocations of OT. The observations in Section 3 comparing the overhead of the HDOT protocol to that of Yao's construction, are relevant in this case, too. We also believe that it is simpler to implement the m-point-SPIR protocol compared to implementing a circuit based solution.

Application I: private matching for cardinality threshold. This is an example where it is important that $\mathcal{P}_1$ receives the default value if no match is found. The scenario involves two parties with private sets of m items, which want to find out if the size of the intersection of the sets is greater than some threshold. The problem was defined in [13] as a variant of the private matching protocol which was the main subject of that paper. The solution there requires the parties to run an OPE for each item x_i of the first party, in which the first party either learns a specific value or a random value, depending on whether x_i is in the set of the second party. The parties then use Yao's protocol to evaluate a circuit whose input is the values learned by $\mathcal{P}_1$, and which computes whether the size of the intersection is greater than the threshold. We can use the m-point-SPIR protocol to replace the OPE: Suppose that $\mathcal{P}_1$'s inputs are $x_1, \ldots, x_n$ and $\mathcal{P}_2$'s inputs are $y_1, \ldots, y_n$. Then for each x_i the parties run an m-point SPIR where $\mathcal{P}_1$ learns α_i if $x_i \in \{y_1, \ldots, y_n\}$, or $\alpha_i + 1$ otherwise, where α is a random number chosen by $\mathcal{P}_2$. We can then ask $\mathcal{P}_1$ to sum the values it learned, and replace Yao's protocol with an OT_1^m , as was done in the *bin*HDOT protocol of Section 4.1. (This was impossible when an OPE was used, since in that case the sum was random if there was even a single item of $\mathcal{P}_1$ which was not in $\mathcal{P}_2$'s set.)

Application II: lottery service As an example of another application of m-point-SPIR, consider a lottery service where the server has a range of tickets, only a few of which are winning tickets. The client uses the protocol to "buy" a ticket, but the client must not know, at least not until some time in the future, whether this is a winning ticket. The server's database contains the prize corresponding to each winning ticket, or the default "no prize" value $\bar{x}$ (which, of course, is associated to most of the tickets). It must be ensured that a client that receives the value $\bar{x}$ cannot identify that this is the default value. The server must not learn which ticket was chosen by the buyer. (A lottery service with many clients must handle many other different issues which we do not describe, but m-point-SPIR seems like a good approach for handling the purchase of tickets.)

References

1. Ben-David, A., Pinkas, B., Nisan, N.: Fairplaymp – a system for secure multi-party computation. In: ACM Conference on Computer and Communications Security—ACM CCS 2008. ACM, New York (2008)
2. Blake, I.F., Kolesnikov, V.: Conditional encrypted mapping and comparing encrypted numbers. In: Crescenzo and Rubin [9], pp. 206–220
3. Bogetoft, P., Damgård, I., Jakobsen, T., Nielsen, K., Pagter, J., Toft, T.: A practical implementation of secure auctions based on multiparty integer computation. In: Crescenzo and Rubin [9], pp. 142–147
4. Boneh, D. (ed.): CRYPTO 2003. LNCS, vol. 2729. Springer, Heidelberg (2003)
5. Cachin, C., Micali, S., Stadler, M.: Computationally private information retrieval with polylogarithmic communication. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 402–414. Springer, Heidelberg (1999)
6. Camenisch, J., Neven, G., Shelat, A.: Simulatable adaptive oblivious transfer. In: Naor [26], pp. 573–590
7. Camenisch, J., Shoup, V.: Practical verifiable encryption and decryption of discrete logarithms. In: Boneh [4], pp. 126–144
8. Canetti, R.: Security and composition of multiparty cryptographic protocols. J. Cryptology 13(1), 143–202 (2000)
9. Di Crescenzo, G., Rubin, A. (eds.): FC 2006. LNCS, vol. 4107. Springer, Heidelberg (2006)
10. Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contracts. In: Advances in Cryptology - Crypto 1982, pp. 205–210 (1982)
11. Fagin, R., Naor, M., Winkler, P.: Comparing information without leaking it. Communications of the ACM 39(5), 77–85 (1996)
12. Feigenbaum, J., Ishai, Y., Malkin, T., Nissim, K., Strauss, M.J., Wright, R.N.: Secure multiparty computation of approximations. ACM Transactions on Algorithms 2(3), 435–472 (2006)
13. Freedman, M.J., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 1–19. Springer, Heidelberg (2004)
14. Goethals, B., Laur, S., Lipmaa, H., Mielikäinen, T.: On private scalar product computation for privacy-preserving data mining. In: Park, C.-s., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 104–120. Springer, Heidelberg (2005)
15. Goldreich, O.: Foundations of Cryptography. Basic Applications, vol. 2. Cambridge University Press, New York (2004)
16. Green, M., Hohenberger, S.: Blind identity-based encryption and simulatable oblivious transfer. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 265–282. Springer, Heidelberg (2007)
17. Hazay, C., Lindell, Y.: Efficient oblivious polynomial evaluation and transfer with simulation-based security (manuscript) (2008)

18. Indyk, P., Woodruff, D.P.: Polylogarithmic private approximations and efficient matching. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 245–264. Springer, Heidelberg (2006)
19. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding cryptography on oblivious transfer - efficiently. In: Wagner [33], pp. 572–591
20. Jarecki, S., Shmatikov, V.: Efficient two-party secure computation on committed inputs. In: Naor [26], pp. 97–114
21. Kushilevitz, E., Ostrovsky, R.: Replication is not needed: single database, computationally-private information retrieval. In: FOCS 1997, pp. 364–373 (1997)
22. Lindell, Y., Pinkas, B.: An efficient protocol for secure two-party computation in the presence of malicious adversaries. In: Naor [26], pp. 52–78
23. Lindell, Y., Pinkas, B., Smart, N.P.: Implementing two-party computation efficiently with security against malicious adversaries. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 2–20. Springer, Heidelberg (2008)
24. Malkhi, D., Nisan, N., Pinkas, B., Sella, Y.: Fairplay - secure two-party computation system. In: USENIX Security Symposium, pp. 287–302. USENIX (2004)
25. Meier, R., Przydatek, B.: On robust combiners for private information retrieval and other primitives. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 555–569. Springer, Heidelberg (2006)
26. Naor, M. (ed.): EUROCRYPT 2007. LNCS, vol. 4515. Springer, Heidelberg (2007)
27. Naor, M., Nissim, K.: Communication preserving protocols for secure function evaluation. In: STOC, pp. 590–599 (2001)
28. Naor, M., Pinkas, B.: Oblivious transfer and polynomial evaluation. In: STOC 1999, pp. 245–254. ACM Press, New York (1999)
29. Naor, M., Pinkas, B.: Computationally secure oblivious transfer. J. Cryptology 18(1), 1–35 (2005)
30. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
31. Paillier, P.: Trapdooring discrete logarithms on elliptic curves over rings. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 573–584. Springer, Heidelberg (2000)
32. Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: Wagner [33], pp. 554–571
33. Wagner, D. (ed.): CRYPTO 2008. LNCS, vol. 5157. Springer, Heidelberg (2008)
34. Wright, R., Yang, Z.: Privacy-preserving bayesian network structure computation on distributed heterogeneous data. In: Proc. of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 713–718. ACM Press, New York (2004)
35. Yao, A.C.-C.: How to generate and exchange secrets (extended abstract). In: FOCS, pp. 162–167. IEEE, Los Alamitos (1986)