On Some Open Questions in Communication-Efficient Cryptocomputing*

Helger Lipmaa

University College London, UK

Abstract. We will give an overview of a recent cryptocomputing method that makes it possible to cryptocompute every language in **L/poly**. We give several nontrivial applications, including: (a) An $\binom{n}{1}$-CPIR protocol with log-squared communication and sublinear server-computation by giving a secure function evaluation protocol for Boolean functions with similar performance, (b) A protocol that makes it possible to compute (say) how similar is client's input to an element in server's database, without revealing any information to the server, (c) A protocol for private database updating with low amortized complexity.

Keywords. Branching programs, computationally-private information retrieval, extended computationally-private information retrieval, private database updating, two-party computation.

1 Introduction

Generic Cryptocomputing Protocol. We start the paper with a different and simpler exposition of the cryptocomputing protocol from [IP07] that is based on earlier papers [KO97,Ste98,Lip05]. This protocol combines a well-known fact that any functionality in complexity class **L/poly** can also be computed by polynomial-size branching programs [Weg00], with a realization that one can privately implement the local selector operation at every node of the branching program by using a suitable two-message 1-out-of-2 computationally-private information retrieval ($\binom{2}{1}$-CPIR) protocol.

More precisely, several existing $\binom{n}{1}$-CPIR protocols [KO97,Ste98,Lip05] can be seen as cryptocomputing a complete ordered binary decision diagram (OBDD), where at every node one uses an efficient two-message $\binom{m}{1}$-CPIR protocol on the second messages of the two-message $\binom{m}{1}$-CPIR protocols of the previous level. The main difference between the CPIR protocols in those papers was the value of m and the use of more and more efficient concrete $\binom{m}{1}$-CPIR protocols. In particular, Lipmaa [Lip05] proposed a very efficient underlying $\binom{2}{1}$-CPIR protocol, based on a length-flexible homomorphic cryptosystem [DJ01,DJ03]. Because the length of server's message in Lipmaa's $\binom{2}{1}$-CPIR protocol has additive length expansion compared to the transmitted message, cryptocomputing an OBDD by using Lipmaa's $\binom{2}{1}$-CPIR protocol results in an $\binom{n}{1}$-CPIR protocol with total communication $\Theta(\log^2 n + \ell \log n)k$ where ℓ is the length of database elements and k is the security parameter [Lip05].

Now, an $\binom{n}{1}$-CPIR protocol can be used to cryptocompute an arbitrary function $f : \{0,1\}^{\log n} \to \{0,1\}^{\ell}$, although with computational cost $\Theta(n)$ for the server which is prohibitive especially when f itself is simple. As noted in [IP07], one can often significantly

* First public draft, March 19, 2008. This draft has been public solely for introducing the ideas to the public. Correlation between this draft and final publications may be very small.

decrease computation by using, instead of a complete OBDD, an efficient branching program [Weg00]. At every internal node of the branching program, one again uses a suitable $\binom{2}{1}$-CPIR protocol on progressively longer inputs. If Lipmaa's $\binom{2}{1}$-CPIR protocol is used, the total communication of this cryptocomputing protocol is quadratic (in the length of client's inputs), and assuming the RAM model, server's computation is linear (in the size of the branching program). In particular, the protocol has polynomial-time computation iff the branching program has polynomial size, which is possible if the decided language belongs to complexity class $\mathbf{L/poly}$. Importantly, the cryptocomputing protocol only reveals the length (i.e., the depth)—but not the shape or even the size—of the branching program to the client which makes it possible for the server to use efficient branching programs that are tailored to her concrete input.

One can use any of the known methods [NP99,AIR01,Kal05,Lip05,IP07,LL07] to transfer the resulting cryptocomputing protocol from a client-private protocol (that corresponds to using a CPIR) to a private protocol (that corresponds to using an oblivious transfer protocol). "Semisimulated" privacy in malicious model is a standard security notion for oblivious transfer protocols [AIR01,Lip05,LL07], and similarly to [FIPR05,IP07] we advocate its use in general cryptocomputing. If needed, however, one can transform the protocols into simulatably secure protocols by using the general methodology proposed in [NN01].

This cryptocomputing protocol can be compared with some other known protocols. By using Yao's two-party protocol [Yao82], one can cryptocompute every function f in $\mathbf{BPP/poly}$ by using communication that is linear in the circuit complexity $C(f)$ of f. In the new cryptocomputing protocol, one can cryptocompute every function f in (probably) smaller class $\mathbf{L/poly}$ but by using communication that is quadratic in the branching program *length* $\mathsf{length}(f)$ that is often significantly smaller than $C(f)$. Sander, Young and Yung [SYY99] proposed a protocol for cryptocomputing everything in a (probably) smaller class $\mathbf{NC^1} \subseteq \mathbf{L/poly}$ but the communication in their protocol was exponential in $\mathsf{length}(f)$ while in the new method it is quadratic. Naor and Nissim [NN01] proposed a protocol that utilized communication-complexity trees. In their protocol, assuming that the communication complexity of the nonprivate protocol is c (note that even for very simple problems $c \geq \log n$, where n is the summatory input length of both parties), the private version has communication complexity $O(c^2)$, computation complexity $O(c2^c)$ and round complexity that is usually larger than 2. They proposed also another protocol that utilized branching programs to achieve better computation but it also had somewhat higher communication and still a larger number of the rounds.

While the same general cryptocomputing protocol was proposed in [IP07] (who were also the first to realize that many previous CPIR protocols [KO97,Ste98,Lip05] cryptocomputed a complete OBDD), they did not propose many concrete non-trivial examples of its applicability. Our aim is to popularize this cryptocomputing protocol, and the best way to do it seems to be to solve a few well-known problems in cryptographic protocol design that are interesting in their own right.

New Applications.

Computation-Efficient CPIR. First, we propose a new protocol for $\binom{n}{1}$-CPIR. Assume that the client has a query $i \in \{0, \ldots, n-1\}$ and the server has a database $D = (D_0, \ldots, D_{n-1})$ of bits. A folklore theorem in this area is that per every query, the server must do $\Omega(n)$ online computation since otherwise she'd know that some element was not queried. We show that this theorem does not hold. Namely, we propose a $\binom{n}{1}$-CPIR protocol where the online computational complexity of the server depends heavily on the concrete database itself and is upperbound by $(1 + o(1))n/\log_2 n$ public-key operations in the worst case. The protocol has log-squared communication in n. We achieve this by writing down a branching program for the function f_D, $f_D(i) := D_i$ and then using the general methodology. The upperbound $(1 + o(1))n/\log_2 n$ on the number of online computations follows from the upperbound on the size of a branching program to compute an arbitrary Boolean function [BHR95].

In fact, this result shows that one can implement secure function evaluation of any $f : \{0,1\}^n \to \{0,1\}$ with communication $\Theta(n^2)$ and computation $(1 + o(1))2^n/n$; the latter upperbound is also tight [BHR95] because there exist Boolean functions for which the size of the best branching program is $(1-o(1))2^n/n$. Thus, this gives us also a worst-case lowerbound for the online computation of $\binom{n}{1}$-CPIR. One can transform the CPIR protocol into an $\binom{n}{1}$-oblivious transfer protocol, where also server's privacy is protected, by say using the computationally efficient transformation of Naor and Pinkas [NP99].

Extended CPIR. Second, we study extended $\binom{n}{1}$-CPIR, a primitive recently defined in [BCPT07]. In extended CPIR, the client submits a private index i and a private secret j to the database server, who replies with the value of a predetermined function on j and server's database ith element. While [BCPT07] only proposed extended CPIR protocols for a few simple functions, we show that by using the described general methodology one can implement any functionality in polynomial-time and log-squared communication. Our results on extended CPIR can for example applied in the next setting in biometric authentication. The client of the extended CPIR protocol collects a fingerprint j of some person, together with her claim that she is the ith employee. Then the client contacts a server who stores encrypted fingerprints of all employees. At the end of the protocol, the client gets to know that person's fingerprint is sufficiently close to the ith fingerprint in database to warrant access, without getting to know fingerprint templates. On the other hand, the server does not get to know which person was trying to enter.

Private database updating. Third, we study private database updating where on client's private index i and a private secret j, the server updates the ith element of his database to j. It is assumed that the database is encrypted, so that the server does not know any elements of the database, nor which database element was updated. Let n be the database size. The first non-trivial solution to this problem, with $\Theta(\sqrt{n})$ communication, was proposed in Crypto 2007 [BKOS07]. Their solution was based on bilinear maps. We propose another non-trivial solution that, for u updates, achieves *amortized* communication complexity $\Theta(u^2 \log^2 n)$; this improves upon the protocol of [BKOS07] for $u = o(\sqrt[4]{n}/\log^4 n)$. Importantly, this protocol only uses a length-flexible additively homomorphic cryptosystem and no bilinear pairings.

Moreover, one can combine ideas of the two last protocols to let the server to store a value of some function of the previous value of the database element and of a secret input of the client. To the best of our knowledge, this problem has not been tackled in the literature at all.

Other applications. Finally, because under many well-known cryptographic assumptions there exist say primitives for computing public-key encryption, signing, collision-resistant hash functions and secret-key encryption in complexity class $\mathbf{NC^1} \subseteq \mathbf{L/poly}$, one can communication-efficiently implement such primitives on encrypted data.

2 Preliminaries, Related Work, Cryptographic Tools

Branching Programs. A branching program/binary decision program on the variable set $X_n = \{x_1, \ldots, x_n\}$ consists of a directed acyclic graph $G = (V, E)$ whose inner nodes (nonsink nodes) have outdegree 2 and a labeling of the nodes and edges. The inner nodes get labels from X_n and the sinks get labels from $\{0, 1\}$. For each inner node, one of the outgoing edges gets the label 0 and the other one gets the label 1. In a BP/BDD on X_n each node v represents a Boolean function f_v defined in the following way. The computation of $f_v(a)$, $a \in \{0,1\}^n$, starts at v. At node labeled by x_i, the outgoing edge labeled by a_i is chosen. Then $f_v(a)$ is equal to the label of the sink which is reached on the considered path. In an ordered BDD, an order of the labels is chosen, and any node on ith level is labeled by the ith label.

It is known that any Boolean formula $f : \{0,1\}^n \to \{0,1\}$ can be computed by a branching program of size $(1 + o(1))2^n/n$ [BHR95], and that every formula of size ℓ can be transformed to a branching program of size $O(\ell^{1+\varepsilon})$ for an arbitrary $\varepsilon > 0$ [Gie01]. In particular, a language (or a Boolean function) has a polynomial-size branching program iff it belongs to the complexity class $\mathbf{L/poly}$.

Denote by $\mathsf{size}(f)$ the minimal size of any branching program computing f. See [Weg00] for an extensive coverage of branching programs.

Public-Key Cryptosystems. Let $\mathsf{pkc} = (\mathsf{gen}, \mathsf{enc}, \mathsf{dec})$ be a length-flexible additively-homomorphic public-key cryptosystem [DJ01], where for every integer $s > 0$, $\mathsf{enc}^s_{\mathsf{pk}}(m)$ maps a plaintext from some set M_s to a ciphertext in some set M_{s+1}. In the case of [DJ01], $M_s = \mathbb{Z}_N^s$ for a large prime N, and thus the plaintext length is sk while the ciphertext length is $(s+1)k$ bits. (In some other length-flexible cryptosystems, the resulting ciphertext is longer, i.e., $(s+2)k$ bits in [DJ03].) Recall that in such a cryptosystem, $\mathsf{enc}^s_{\mathsf{pk}}(m_1) \cdot \mathsf{enc}^s_{\mathsf{pk}}(m_2) = \mathsf{enc}^s_{\mathsf{pk}}(m_1 + m_2)$, and moreover, $\mathsf{enc}^s_{\mathsf{pk}}(m)$ is a valid plaintext of $\mathsf{enc}^{s+1}_{\mathsf{pk}}$, so that one can legally multiple-encrypt messages as say in $\mathsf{enc}^{s+2}_{\mathsf{pk}}(\mathsf{enc}^{s+1}_{\mathsf{pk}}(\mathsf{enc}^s_{\mathsf{pk}}(m)))$.

In the IND-CPA game, the challenger first generates a random $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{gen}$, and sends pk to the attacker. Attacker chooses two messages m_0, m_1 and a length parameter s, and sends them to the challenger. Challenger picks a random bit b, and sends a ciphertext $\mathsf{enc}^s_{\mathsf{pk}}(m_b)$ to attacker. Attacker outputs a bit b', and wins if $b = b'$. A public-key cryptosystem is *IND-CPA*

secure if the probability that any polynomial-time attacker wins in the IND-CPA game is negligibly different from 1/2.

In the α-IND-LFCPA game [Lip05], the challenger first generates a random $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{gen}$, and sends pk to the attacker. Attacker chooses α message pairs m_{i0}, m_{i1} and length parameters s_i, and sends them to the challenger. Challenger picks α random bits b_i, and sends ciphertexts $\mathsf{enc}^{s_i}_{\mathsf{pk}}(m_{ib_i})$ to attacker. Attacker outputs α bits b'_i, and wins if $b_i = b'_i$ for at least one i. A public-key cryptosystem is α-*IND-LFCPA* secure if the probability that any polynomial-time attacker wins in the α-IND-LFCPA game is negligibly different from $1 - 1/2^\alpha$.

We will implicitly assume the existence of a function $\mathsf{compress}$ that, given $\mathsf{enc}^{s'}_{\mathsf{pk}}(m)$, pk, s' and s for $s' \geq s$, returns $\mathsf{enc}^{s'}_{\mathsf{pk}}(m)$. Such a function exists for the cryptosystem from [DJ01] as shown in [Lip05]. In particular, in this case the α-IND-LDCPA assumption can be clearly reduced to the IND-CPA assumption.

Computationally-Private Information Retrieval And Oblivious Transfer. In a 1-out-of-n computationally-private information-retrieval protocol, $\binom{n}{1}$-CPIR, for ℓ-bit strings, the client has a private input $i \in \mathbb{Z}_n$ and the server has a database $D = (D_0, \ldots, D_{n-1})$. The client wants to retrieve D_i. A CPIR protocol is usually required to be client-private in the sense of indistinguishability, i.e., that a malicious server cannot distinguish queries corresponding to any two indexes i_1, i_2. An oblivious transfer protocol provides additionally server-privacy.

In [Lip05], Lipmaa proposed the next basic $\binom{2}{1}$-CPIR protocol for ℓ-bit strings. Let $\mathsf{pkc} = (\mathsf{gen}, \mathsf{enc}, \mathsf{dec})$ be a length-flexible additively homomorphic public-key cryptosystem. Let s be an integer such that $sk \geq \ell \geq (s-1)k$. For $i \in \{0, 1\}$, client sends a new public key pk and $c \leftarrow \mathsf{enc}^s_{\mathsf{pk}}(i)$ to server, who replies with $d \leftarrow (\mathsf{enc}^s_{\mathsf{pk}}(1)/c)^{D_0} \cdot c^{D_1}$. If $i \in \{0, 1\}$ then clearly $d = \mathsf{enc}^s_{\mathsf{pk}}(D_0(1-i) + D_1 \cdot i) = \mathsf{enc}^s_{\mathsf{pk}}(D_i)$.

$\binom{n}{1}$-CPIR By Using Kushilevitz-Ostrovsky Recursion. Kushilevitz and Ostrovsky [KO97] proposed a generic protocol for recursively computing $\binom{n}{1}$-CPIR. In the first step of recursion, server's original database of size n is divided into n/m pieces of m elements, where the value m is carefully chosen. Next, a more basic two-message $\binom{m}{1}$-CPIR is applied to every piece. The second messages of the $\binom{m}{1}$-CPIR protocols are, instead of being sent back to the client, stored at server as an intermediate database of smaller size n/m (but longer string-length). Then a two-message $\binom{n/m}{1}$-CPIR protocol is applied to this database. Next, the same step is applied recursively, with the intermediate database of size n/m being divided into smaller and smaller chunks. The server only returns the answer of the final CPIR when her database has been reduced to contain a small number of relatively long elements.

Kushilevitz and Ostrovsky used a relatively inefficient basic CPIR protocol that was based on the difficulty of the quadratic residuosity problem. A more efficient basic CPIR protocol was proposed by Stern [Ste98] who used a recently proposed additively homomorphic cryptosystem. Finally, Lipmaa [Lip05] used a length-flexible additively homomorphic cryptosys-

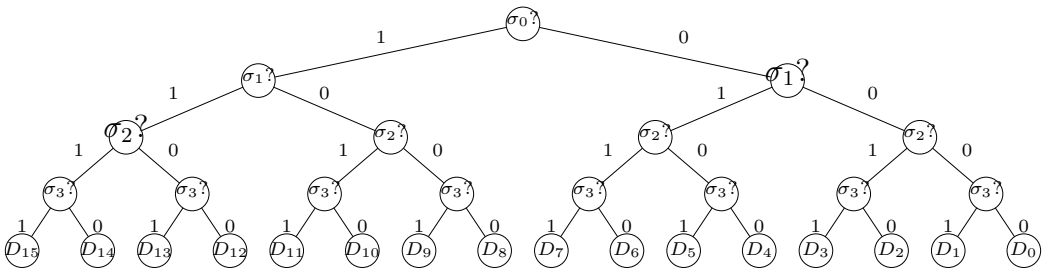

Fig. 1. Lipmaa' CPIR is based on a complete OBDD

tem [DJ01] to construct a very efficient basic $\binom{2}{1}$-CPIR protocol for ℓ-bit database elements. The crucial feature of Lipmaa's basic CPIR protocol is that there the server's communication grows only additively, which makes it possible to achieve log-squared communication in the final $\binom{n}{1}$-CPIR protocol, by applying $\log n$ recursive layers of the Kushilevitz-Ostrovsky basic recursion. In comparison, the protocols from [KO97,Ste98] constructed a basic $\binom{m}{1}$-CPIR protocol with multiplicative length expansion, which resulted in sublinear but still superpolylogarithmic communication for the final $\binom{n}{1}$-CPIR protocol.

Lipmaa's $\binom{n}{1}$-CPIR Protocol. Now, let $t := \lceil \log n \rceil$. In Lipmaa's $\binom{n}{1}$-CPIR protocol [Lip05] for ℓ-bit strings, see Fig. 1, the client sends to the server $\mathsf{enc}_{\mathsf{pk}}^{s+j}(\sigma_j)$ for all bits σ_j of his index $\sigma = \sum_{j=0}^{t-1} \sigma_j 2^j$. Server's input is a database $D = (D_0, \ldots, D_{n-1})$. We also denote D_i by $D_{i_{t-1},\ldots,i_0}$. In this protocol, server's computation can be seen as the cryptocomputing of a complete ordered binary decision diagram (OBDD), where at the ith level, the server branches on the value of σ_i. The leafs are labeled by D_i, where $D_{i_{t-1},\ldots,i_0}$ is at the leaf that corresponds to the branchings made according to tests $[\sigma_j =^? i_j]$ being true. At the every node on the jth level, the server computes Lipmaa's $\binom{2}{1}$-CPIR on client's "message" $c_j = \mathsf{enc}_{\mathsf{pk}}^{s+j}(\sigma_j)$ and "database" b_0, b_1, the server's "message". That is, given $c_j = \mathsf{enc}_{\mathsf{pk}}^{s+j}(\sigma_j)$ and two inputs b_0, b_1 to this node, the server computes the value $(\mathsf{enc}_{\mathsf{pk}}^{s+j}(1)/c_j)^{b_0} \cdot c_j^{b_1} = \mathsf{enc}_{\mathsf{pk}}^{s+j}(\sigma_j \cdot b_1 + (1-\sigma_j) \cdot b_0) = \mathsf{enc}_{\mathsf{pk}}^{s+j}(b_{\sigma_j})$. This value is then used as an input at the level $j-1$.

At the end of Lipmaa's $\binom{n}{1}$-CPIR protocol, the server obtains the value

$$\mathsf{enc}_{\mathsf{pk}}^{s+t}(\mathsf{enc}_{\mathsf{pk}}^{s+t-1}(\ldots \mathsf{enc}_{\mathsf{pk}}^{s}(D_{\sigma_{t-1},\ldots,\sigma_0}) \ldots)) \ ,$$

and sends it to the client. The client uses the secret key to multiple-decrypt server's message.

Semisimulatable Security. Within this work we are using the convention of many previous papers on oblivious transfer protocol [AIR01,Lip05] that only require privacy in the malicious model. Various papers, e.g., [FIPR05,IP07], also recommend to use this convention for other cryptographic protocols. According to this convention, client's privacy is guaranteed in the sense of indistinguishability, while server's privacy is guaranteed in the sense of simulatability.

This assumption makes it possible to design two-message CPIR protocols that are both communication and computation-efficient.

We now give a definition of this notion for a wider class of two-message protocols. For client-privacy, a probabilistic polynomial-time server is required not to be able to distinguish between client's messages corresponding to any two client's inputs σ_0 and σ_1, possibly chosen by himself. For server-privacy, we require the existence of a simulator that, given client's message and client's legitimate output corresponding to this message, generates server's message that is (statistically) indistinguishable from server's message in the real protocol. A protocol is *semisimulatably private* if it is both client-private and server-private according to these definitions.

3 Universal Cryptocomputing Method

In this section, we describe a concrete universal cryptocomputing protocol, based on Lipmaa's efficient $\binom{2}{1}$-CPIR, and on an arbitrary branching program [Weg00] implementing the functionality, by following earlier work by [IP07]. Because we only use a $\binom{2}{1}$-CPIR, our exposition is simpler than the more general exposition of [IP07] with almost straightforward security proofs. Moreover, we allow—and prove that this is correct—the client and the server to implement many different branching programs, and to share client's first round message between several protocols.

Universal Cryptocomputing. Recall that Lipmaa's $\binom{n}{1}$-CPIR protocol in essence cryptocomputed a complete ordered binary decision diagram (OBDD). By using a complete OBDD, one can clearly cryptocompute an arbitrary function. As shown in [IP07], one can generalize this computational process to an arbitrary branching program, where at every node one branches by using an arbitrary but fixed (Boolean) encrypted input of the client. (The concrete protocol of [IP07] is slightly different.) The leaf values of the branching program correspond to server's inputs. Let P be the corresponding branching program, $\mathsf{length}(P)$ its length (i.e., the depth) and $\mathsf{size}(P)$ its size. At every node of the branching program, given input values b_0, b_1, and a (length-flexible additively homomorphic) encryption of some bit σ, the output value of the node will be a (length-flexible additively homomorphic) encryption of b_σ. The output of the branching program is equal to a $\mathsf{length}(P)$-times encryption of an input, selected by the encrypted client inputs.

Now, let ℓ be the output length of the functionality. Then, the client must submit her n inputs in a $\mathsf{length}(i)$-times encrypted form, where $\mathsf{length}(i)$ is the largest depth at which the input variable σ_i is used, i.e., as $\mathsf{enc}_{\mathsf{pk}}^{s+\mathsf{length}(i)}(\cdots \mathsf{enc}_{\mathsf{pk}}^{s}(\sigma_i)\cdots)$. (Recall that given $\mathsf{enc}_{\mathsf{pk}}^{s+1}(m)$, one can cryptocompute $\mathsf{enc}_{\mathsf{pk}}^{s}(m)$ [Lip05], thus one does not have to submit the values $\mathsf{enc}_{\mathsf{pk}}^{j}(m)$ for several different j-s.) Thus client-communication is upper-bounded by $\sum_i(\mathsf{length}(i)+s+1)k \le n\cdot(\mathsf{length}(P)+s+1)k \le n\cdot(\mathsf{length}(P)k+\ell+k)$. Server-communication consists of one $\mathsf{length}(P)$ times encrypted message of length $(\mathsf{length}(P)+s+1)k \le (\mathsf{length}(P)+1)k+\ell$. Server's computation is linear in $\mathsf{size}(P)$. (Here and in the following we assume that the server does a unit amount of work at every node.) Therefore, every functionality f has a

cryptocomputing protocol with total communication $\leq (n+1)((\mathsf{length}(P)+1)k+\ell)$, where P is a branching program that implements f.

If we assume that the branching program is read-once (i.e., at every path there is at most one branch by every variable) then $\mathsf{length}(P) \leq n$ and the total communication is upper-bounded by $(n+1)((n+1)k+\ell)$. More generally, if the branching program is read-ρ-times then the total communication is upper-bounded by $(n+1)((\rho n+1)k+\ell)$.

Finally, note that client is completely oblivious of the shape of the branching program, except the length of it. He just encrypts his input bits by using a length parameter that depends on the length of the branching program (and on the output length ℓ), and then receives back a multiple-encryption of the output. This in particular means that client's inputs can be re-used in many different branching programs, and that the shape of the branching program can depend on server's inputs.

Homomorphic Add-Ons. Because one uses a homomorphic cryptosystem, the server can potentially apply homomorphic operations to the leafs of the branching program. For example, she can let the leaf to be an encryption of 0 iff a certain client's input decrypts to some value held by her. Or, it can be equal to the ciphertext of a sum of certain client's inputs with certain server's inputs.

Non-Binary Branching Programs And Generalisation. One can implement non-binary branching programs by plugging in suitably defined CPIR protocols at the inner nodes. This does usually not improve the communication significantly, and on the other hand, makes security proofs more complicated. As a plus-side, however, using suitable $\binom{n}{1}$-CPIR protocols in every inner node of the branching program makes it possible to generalise the approach from using length-flexible cryptosystems to the existence of suitable CPIR protocols. See [IP07] for more details.

4 Client-Privacy: Statement And Proof

We will prove the security in the same semisimulatability model that is standard for at least two-message oblivious transfer protocols, see Sec. 2. That is, we require that for a server it should be computationally hard to distinguish between client's messages corresponding to two client's inputs, chosen by the server herself. On the other hand, for server's privacy, there should exist a simulator that, given client's (possibly incorrectly formed) input and honest server's output in ideal world after receiving this input, outputs a message from a distribution that is statistically close to the message of honest server that corresponds to client's message.

As in the case of oblivious transfer protocols, we are first giving a proof of client-privacy and then showing how one can add server-privacy to it without almost any additional cost by using universal methods. However, while the proof of client-privacy (presented next) does not depend significantly on the concrete protocol, the proof of server-privacy will be given separately for every protocol.

Client-Privacy Proof. The server only sees (say α) homomorphically encrypted values, and thus client-privacy is guaranteed by the α-IND-LFCPA security of the cryptosystem. Because of the existence of the $\mathsf{compress}$ function, client's privacy follows from the IND-CPA security. More precisely, the server sees up to n ciphertexts $\mathsf{enc}_{\mathsf{pk}}^{s_j}(\sigma_j)$. Let $s = \min_j s_j$. By using the compress function, the attacker computes $\mathsf{enc}_{\mathsf{pk}}^{s}(\sigma_j)$ for all j.

5 $\binom{n}{1}$-CPIR Protocol with Less Than n Public-Key Operations

Assume that $\ell = 1$ and the database size is 2^n, i.e., that the server's database consists of 2^n bits. In this case, we can pose CPIR protocol as follows. Assume client has an input $i \in \{0,1\}^n$ and the server has a Boolean function $f_D : \{0,1\}^n \to \{0,1\}$ such that $f_D(i) = D_i$. The client needs to retrieve $f_D(i)$. It is known that any Boolean function can be computed by a branching program of size $(1+o(1))2^n/n$ that has length $\Theta(n+\log n)$ [BHR95]. Computation of such a branching program may take 2^n time, but evaluation of the branching program on concrete input i takes less than 2^n operations.

In particular, when applying the above-described universal cryptocomputing method, one needs to do $\mathsf{size}(P)$ public-key operations. Because in this case $\mathsf{size}(P) = (1+o(1))2^n/n$, the number of public-key operations is less than 2^n, the size of server's database, for any possible database, i.e., Boolean function D. To the best of our knowledge, this is the first $\binom{2^n}{1}$-CPIR with this property.

Achieving Server-Privacy. Recall that an $\binom{n}{1}$-CPIR protocol that also achieves server-privacy is called an $\binom{n}{1}$-OT protocols. There are many existing CPIR-to-OT transformations [NP99,AIR01,Kal05,Lip05,IP07,LL07]. In the concrete case, because we are interested in computation-efficient, we would recommend the use of the transformation from [NP99].

Computation-Efficient CPIR for ℓ-Bit Strings. Let $f : \{0,1\}^n \to \{0,1\}^\ell$. By the upper bound of [BHR95], clearly $\mathsf{size}(f) \leq \ell \cdot (1 + o(1))2^n/n$. By following the proof of [Weg00], one can easily show that $\mathsf{size}(f) \leq (3 + o(1))2^{n+\log \ell}/(n + \log \ell)$.

Example Application. Many real databases are naturally quite redundant. In addition, $\binom{n}{1}$-CPIR is often used as a part of other cryptographic protocols. As an example, in [AJL04], the authors studied the next question: server has a bit $b =\in \{0,1\}$. Client should learn a coin toss b' with output $\Pr[b' = b] = p/n$ and $\Pr[b' = 1 - b] = 1 - p/n$ for fixed integers p and n. A solution proposed in [AJL04] required the server to construct a random Boolean database $D = (D_1, \ldots, D_n)$, such that exactly p entries of the database are equal to b. After that, the client performed an $\binom{n}{1}$-OT protocol to obtain a random entry from D. Clearly D is highly structured and thus by applying the new $\binom{n}{1}$-CPIR protocol, one can often get a huge win in computation compared to any existing $\binom{n}{1}$-CPIR.

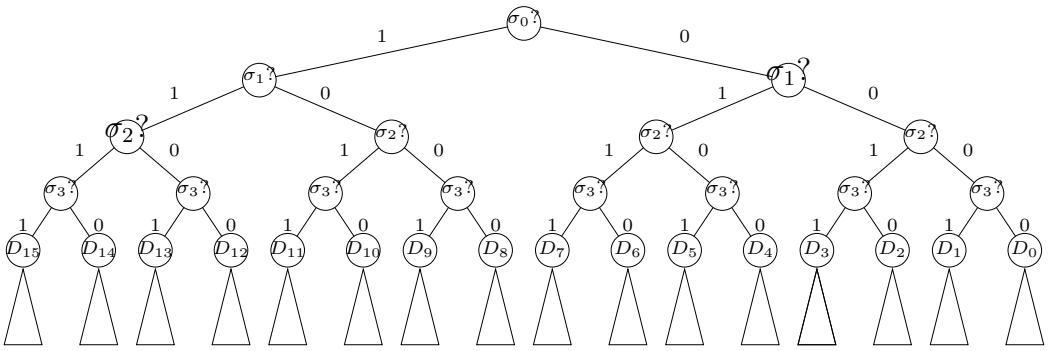

Fig. 2. Extended CPIR based on a complete OBDD for CPIR and arbitrary brancing programs to compute the functionality

6 Extended CPIR

6.1 Protocol And Applications

Preliminaries. In [BCPT07] the authors considered the extended CPIR, where one can obtain, instead of the database element D_σ itself, some function $f(j, D_\sigma)$ of client's second input and the database element. The authors proposed efficient solutions for equality (based on homomorphic encryption) and Hamming weight (based on 2-homomorphic encryption).

New Protocol. One can clearly compute *any* function by using the new methodology. More precisely, assume that client has two inputs i and j, and server has a database $D = (D_0, \ldots, D_{n-1})$. The client must obtain $f(j, D_i)$. Intuitively, we construct a protocol for extended CPIR by plugging in branching programs $P_{f,j}$ for computing $f(j, D_\sigma)$, instead of just a leaf with D_σ, at the bottom of the complete OBDD for computing the CPIR itself. The resulting protocol has length $t + \max_j \mathsf{length}(P_{f,j})$. (See Fig. 2.)

Extensions. One can further optimize this solution by using non-generic techniques. First of all, the top OBDD can be directly replaced by any secure CPIR protocol, like the protocol of Gentry and Ramzan [GR05]. Second, based on the fact that all computation uses homomorphic encryption, one can use homomorphic operations at the bottom of the branching program. For example, to compute equality $f(j, D_\sigma) = [j =^? D_\sigma]$, one can input as leaf the values $\star(j - D_\sigma)$, where $\star$ denotes a suitable random value (a random group element if Elgamal is used, or a slight variation of it when Paillier/Damgård-Jurik is used [LL07]).

Examples. As an example, suppose that the client wants to establish whether $j > D_\sigma$, $j = D_\sigma$ or $j < D_\sigma$, where all values are ℓ-bit long. One can construct a branching program for this with length ℓ and size $2\ell + 1$. Thus without optimisation, this method results in a program with communication $\approx (n+1)((t+\ell)k+\ell) = \Theta(ntk+n\ell)$ and computation $\Theta(n \cdot \ell)$.

However, if one uses the Gentry-Ramzan CPIR protocol, the total communication becomes $n(\ell k+\ell)+O(\log n+\ell k+\ell+k)$.

As another example, suppose that the client wants to establish whether $w_h(j, D_\sigma) < s$, where all values are ℓ-bit long and w_h denotes the Hamming distance. One can construct a branching program for this with length ℓ and size $\sum_{j=0}^{\ell}\sum_{i=0}^{s}\binom{j}{i} = \sum_{i=0}^{s}\binom{\ell+1}{i+1} = \sum_{i=0}^{s+1}\binom{\ell+1}{i} - (\ell+1)$. Now, if $s < \ell/2$ then $\sum_{i=0}^{s+1}\binom{\ell+1}{i}$ is approximately equal to $(\ell - s)\binom{\ell+1}{s+1}/(\ell-2t-1)$. Thus without optimisation, this method results in a program with communication $\approx (n+1)((s+\ell)k+\ell) = \Theta(nsk+n\ell)$ and computation $\Theta(n\cdot\binom{\ell+1}{s+1})$. However, if one uses the Gentry-Ramzan CPIR protocol, the total communication becomes again $n(\ell k+\ell)+O(\log n+\ell k+\ell+k)$.

7 Private Database Updating

7.1 Protocol And Applications

Preliminaries. Assume that client has outsourced his database to the server. Due to the privacy requirements, the database is encrypted. We would like the client to be able to update the database so that the server does not know which element was changed. That is, client has (σ, ν), server has an encrypted database $D = (D_0, \ldots, D_{n-1})$). The client has no output, while server obtains a new encrypted database D' such that D'_i and D_i decrypt to the same value if $i \neq \sigma$, while D'_σ decrypts to ν. It is trivial do construct a protocol for this with $\Theta(n)$ communication. For example, the client and the server first execute a CPIR protocol, so that the client obtains the current value of D_σ. Then client forwards to the server n ciphertexts c_j, where c_σ decrypts to $\nu - D_\sigma$ and other c_j-s decrypt to 0. The server multiplies encryptions of D_j with the new ciphertexts c_j. The first non-trivial protocol for this task was recently proposed in [BKOS07]. Essentially, a variant of it uses bilinear pairings to send $2\sqrt{n}$ ciphertexts c'_j and c''_j—such that the decryption of c_j is equal to the product of decryptions of c_j and c''_j—instead of n ciphertexts.

New Protocol. By using our general methodology, we can achieve amortised communication $u^2 \cdot \log^2 n$ if the number of updates is upper-bounded by u. The idea follows. Client's inputs are $(\sigma_{t-1}, \ldots, \sigma_0; j)$. Server's input is an encrypted database D. The server runs n branching programs P_i in parallel, where P_i returns D_i if $i \neq \sigma$, and j otherwise. (See Fig. 3.) Let the new database be D'. Instead of returning D' to the client, the server stores D' instead of D.

Assume that database elements have length ℓ. Then before any updates the server stores a database with elements of size $(s+1)k$ where s is defined as usually. The first update protocol has communication complexity $\leq (t+1)\cdot(\mathsf{length}(P)k+\ell) = (t^2+t)\cdot k+(t+1)\ell$. The new database has elements of size $(s+t+1)k \leq (t+1)k+\ell$ and thus the next update protocol has communication complexity $(t^2+t)\cdot k+(t+1)(tk+k+\ell)$. Analogously, the ith update protocol has communication complexity $(t^2+t)\cdot k+(t+1)(itk+ik+\ell)$, and thus the first u protocols have total communication complexity $u(t^2+t)\cdot k+(t+1)\sum_{i=0}^{u-1}(itk+ik+\ell) = O(t^2u^2k)$.

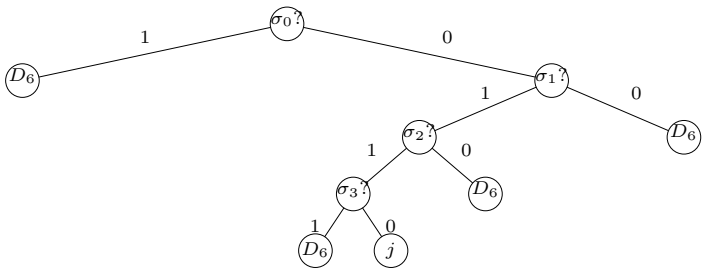

Fig. 3. A branching program that returns j if $\sigma = 6$, and D_6 otherwise

Thus this protocol is more communication-efficient than the protocol of [BKOS07] if $u \leq \sqrt[4]{n}/\log^4 n$. Moreover, it does *not* use pairings. In fact, server's computation is $O(n \log n)$.

Security. Client-privacy follows from the previous general proof. Notice that in this case we are not interested in server-privacy at all.

8 Open Questions

The presented protocols use Lipmaa's underlying $\binom{n}{1}$-CPIR protocol. They could be made more communication-efficient if in this CPIR protocols, client's first message's length would not depend on ℓ, the length of database elements. One could hope to achieve this by designing a length-flexible public-key cryptosystem where one can compute $\mathsf{enc}^{s+1}_{\mathsf{pk}}(m)$ given only $\mathsf{enc}^{s}_{\mathsf{pk}}(m)$. This would enable to get rid of the dependency from $\mathsf{length}(P)$ in communication and thus decrease total communication of the protocols from $O(n^2)$ to $O(n)$ where n is the length of client inputs.

Can one use homomorphic properties more extensively? For example, if the client encrypts values b_i, it'd be nice if the server could branch on arbitrary sums $\sum b_i$. This however needs an existence of a "suitable" one-out-of-many CPIR where the input is not a vector of bit-encryptions (as in Lipmaa's CPIR) but an encryption of the index. If such CPIR could be constructed, one could in fact get exponential decrease of the branching program length in some cases. For example, if server's output has to be the index of the first one in client input set, the server can do a binary search for it: $\sum_{i<n/2} b_i = 0$ iff the first 1 is in the second half, etc.

Construct even more non-trivial examples of usefulness.

References

[AIR01] William Aiello, Yuval Ishai, and Omer Reingold. Priced Oblivious Transfer: How to Sell Digital Goods. In Birgit Pfitzmann, editor, *Advances in Cryptology — EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 119–135, Innsbruck, Austria, May 6–10, 2001. Springer-Verlag.

[AJL04] Andris Ambainis, Markus Jakobsson, and Helger Lipmaa. Cryptographic Randomized Response Techniques. In Feng Bao, Robert H. Deng, and Jianying Zhou, editors, *Public Key Cryptography 2004*, volume 2947 of *Lecture Notes in Computer Science*, pages 425–438, Singapore, March 1–4, 2004. Springer-Verlag.

[BCPT07] Julien Bringer, Hervé Chabanne, David Pointcheval, and Qiang Tang. Extended Private Information Retrieval and Its Application in Biometrics Authentications. In Feng Bao, San Ling, Tatsuaki Okamoto, Huaxiong Wang, and Chaoping Xing, editors, *Cryptology and Network Security, 6th International Conference, CANS 2007*, volume 4856 of *Lecture Notes in Computer Science*, pages 175–193, Singapore, December 8–10, 2007. Springer-Verlag.

[BHR95] Yuri Breitbart, Harry B. Hunt III, and Daniel J. Rosenkrantz. On The Size of Binary Decision Diagrams Representing Boolean Functions. *Theoretical Computer Science*, 145(1&2):45–69, 1995.

[BKOS07] Dan Boneh, Eyal Kushilevitz, Rafail Ostrovsky, and William E. Skeith III. Public Key Encryption That Allows PIR Queries. In Alfred Menezes, editor, *Advances in Cryptology — CRYPTO 2007, 27th Annual International Cryptology Conference*, volume 4622 of *Lecture Notes in Computer Science*, pages 50–67, Santa Barbara, USA, August 19–23, 2007. Springer-Verlag.

[DJ01] Ivan Damgård and Mads Jurik. A Generalisation, A Simplification And Some Applications of Paillier's Probabilistic Public-Key System. In Kwangjo Kim, editor, *Public Key Cryptography 2001*, volume 1992 of *Lecture Notes in Computer Science*, pages 119–136, Cheju Island, Korea, February 13–15, 2001. Springer-Verlag.

[DJ03] Ivan Damgård and Mads Jurik. A Length-Flexible Threshold Cryptosystem with Applications. In Rei Safavi-Naini, editor, *The 8th Australasian Conference on Information Security and Privacy*, volume 2727 of *Lecture Notes in Computer Science*, pages 350–364, Wollongong, Australia, July 9-11, 2003. Springer-Verlag.

[FIPR05] Michael J. Freedman, Yuval Ishai, Benny Pinkas, and Omer Reingold. Keyword Search and Oblivious Pseudorandom Functions. In Joe Kilian, editor, *The Second Theory of Cryptography Conference, TCC 2005*, volume 3378 of *Lecture Notes in Computer Science*, pages 303–324, Cambridge, MA, USA, February 10–12, 2005. Springer Verlag.

[Gie01] Oliver Giel. Branching Program Size Is Almost Linear in Formula Size. *Journal of Computer and System Sciences*, 63(2):222–235, September 2001.

[GR05] Craig Gentry and Zulfikar Ramzan. Single-Database Private Information Retrieval with Constant Communication Rate. In Luis Caires, Guiseppe F. Italiano, Luis Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *The 32nd International Colloquium on Automata, Languages and Programming, ICALP 2005*, volume 3580 of *Lecture Notes in Computer Science*, pages 803–815, Lisboa, Portugal, 2005. Springer-Verlag.

[IP07] Yuval Ishai and Anat Paskin. Evaluating Branching Programs on Encrypted Data. In Salil Vadhan, editor, *The Fourth Theory of Cryptography Conference, TCC 2007*, volume 4392 of *Lecture Notes in Computer Science*, pages 575–594, Amsterdam, The Netherlands, February 21–24, 2007. Springer Verlag.

[Kal05] Yael Tauman Kalai. Smooth Projective Hashing and Two-Message Oblivious Transfer. In Ronald Cramer, editor, *Advances in Cryptology — EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 78–95, Aarhus, Denmark, May 22–26, 2005. Springer-Verlag.

[KO97] Eyal Kushilevitz and Rafail Ostrovsky. Replication is Not Needed: Single Database, Computationally-Private Information Retrieval. In *38th Annual Symposium on Foundations of Computer Science*, pages 364–373, Miami Beach, Florida, October 20–22, 1997. IEEE Computer Society.

[Lip05] Helger Lipmaa. An Oblivious Transfer Protocol with Log-Squared Communication. In Jianying Zhou and Javier Lopez, editors, *The 8th Information Security Conference (ISC'05)*, volume 3650 of *Lecture Notes in Computer Science*, pages 314–328, Singapore, September 20–23, 2005. Springer-Verlag.

[LL07] Sven Laur and Helger Lipmaa. A New Protocol for Conditional Disclosure of Secrets And Its Applications. In Jonathan Katz and Moti Yung, editors, *5th International Conference on Applied Cryptography and Network Security – ACNS'07*, volume 4521 of *Lecture Notes in Computer Science*, pages 207–225, Zhuhai, China, June 5–8, 2007. Springer-Verlag.

[NN01] Moni Naor and Kobbi Nissim. Communication Preserving Protocols for Secure Function Evaluation. In *Proceedings of the Thirty-Third Annual ACM Symposium on the Theory of Computing*, pages 590–599, Heraklion, Crete, Greece, July 6–8 2001. ACM Press.

[NP99] Moni Naor and Benny Pinkas. Oblivious Transfer and Polynomial Evaluation. In *Proceedings of the Thirty-First Annual ACM Symposium on the Theory of Computing*, pages 245–254, Atlanta, Georgia, USA, May 1–4, 1999. ACM Press.

[Ste98] Julien P. Stern. A New And Efficient All Or Nothing Disclosure of Secrets Protocol. In Kazuo Ohta and Dingyi Pei, editors, *Advances on Cryptology — ASIACRYPT '98*, volume 1514 of *Lecture Notes in Computer Science*, pages 357–371, Beijing, China, October 18–22, 1998. Springer-Verlag.

[SYY99] Tomas Sander, Adam Young, and Moti Yung. Non-Interactive CryptoComputing For NC^1. In *40th Annual Symposium on Foundations of Computer Science*, pages 554–567, New York, NY, USA, 17–18 October 1999. IEEE Computer Society.

[Weg00] Ingo Wegener. *Branching Programs and Binary Decision Diagrams: Theory and Applications*. Monographs on Discrete Mathematics and Applications. Society for Industrial Mathematics, 2000.

[Yao82] Andrew Chi-Chih Yao. Protocols for Secure Computations (Extended Abstract). In *23rd Annual Symposium on Foundations of Computer Science*, pages 160–164, Chicago, Illinois, USA, 3–5 November 1982. IEEE Computer Society Press.