Efficient Cryptographic Protocol Design Based on Distributed El Gamal Encryption

Felix Brandt

Stanford University, Stanford CA 94305, USA
brandtf@cs.stanford.edu

Abstract. We propose a set of primitives based on El Gamal encryption that can be used to construct efficient multiparty computation protocols for certain low-complexity functions. In particular, we show how to privately count the number of true Boolean disjunctions of literals and pairwise exclusive disjunctions of literals. Applications include efficient two-party protocols for computing the Hamming distance of two bitstrings and the greater-than function. The resulting protocols only require 6 rounds of interaction (in the random oracle model) and their communication complexity is $\mathcal{O}(kQ)$ where k is the length of bit-strings and Q is a security parameter. The protocols are secure against active adversaries but do not provide fairness. Security relies on the decisional Diffie-Hellman assumption and error probability is negligible in Q.

1 Introduction

Secure multiparty computation (MPC) deals with protocols that allow a group of agents to jointly compute a function of their individual private inputs, so that only the function value is revealed in the end. Since Yao's and Goldreich et al's seminal completeness results [Yao86, GMW87], it is well known that any function can be computed securely if trapdoor permutations exist. However, the general constructions in [Yao86, GMW87] have proven to be rather inefficient and unpractical. It has been shown that general MPC is feasible in a *constant* number of rounds with polynomial communication complexity for various settings such as 2-party MPC (without fairness) [Lin01] or n-party MPC (with an honest majority) [BMR90]. Although theoretically interesting, the constructions of the underlying proofs do not yield practical constant-round MPC schemes due to the extensive use of generic proofs of knowledge.

In this paper, we propose a set of cryptographic techniques that enable the efficient computation of "low-complexity" functions in the presence of active adversaries. These techniques, some of which have been known for some time, can be used straightforwardly to construct round-efficient protocols for the equality function (solving the so-called socialist millionaires' problem), the Boolean OR function (*e.g.*, for veto voting), or the maximum function. Furthermore, we show how to privately count the number of true Boolean disjunctions of literals and pairwise exclusive disjunctions of literals. Applications include efficient

D. Won and S. Kim (Eds.): ICISC 2005, LNCS 3935, pp. 32–47, 2006.

two-party protocols for computing the Hamming distance of two bitstrings and the greater-than function (thus providing a solution to Yao's millionaires' problem [Yao82] in which two millionaires (Alice and Bob) want to find out who is richer without revealing their wealth). Our primary objective when designing these protocols was to minimize round complexity as interaction over a computer network is usually the most time-consuming operation in distributed protocols. Our protocol for the millionaires' problem only requires 6 rounds of interaction (in the random oracle model) and its communication complexity is $\mathcal{O}(kQ)$ where k is the length of bit-strings to be compared and Q is a security parameter. To the best of our knowledge, this is the most efficient constant-round protocol for the millionaires' problem. Under reasonable conditions (see Section 5.2), each party only needs to communicate about 73 Kbytes of data. This is achieved by exploiting the homomorphicity of the underlying encryption scheme when evaluating a modified Boolean formula for the greater-than function. The protocol is correct with error probability $\mathcal{O}(2^{-Q})$. It does not provide fairness, *i.e.*, one party might learn the outcome and leave the other party uninformed by quitting the protocol prematurely. However, fairness can be obtained by using known standard techniques of gradual exchange (see *e.g.*, [GMY04]). For this paper, we assume that there is either a fairness-providing third-party, *i.e.*, a party that does not reveal information or quits prematurely,[1] or that only one of the agents, say Alice, is supposed to learn the function value.

Many representations for *general* secure MPC have been suggested in the literature: Boolean circuits [Yao86], arithmetic circuits [GMW87], branching programs [Kil88], low degree polynomials [BFKR90], randomizing polynomials [IK00], *etc.* Our approach differs in that we provide a set of efficient building blocks (distributed homomorphic encryption, random exponentiation, and verifiable mixing) for which there exist efficient proofs of correctness and use these to construct *special-purpose* protocols for a limited, but nevertheless relevant, group of functions.

Our primitives are built around El Gamal encryption [El 85] because it allows for the efficient generation of distributed keys and because encrypted values can easily be exponentiated with jointly created random numbers. The building blocks can be used for any number of parties. We consider full privacy, *i.e.*, $(n-1)$-privacy, rather than threshold privacy.[2] Any party that deviates from the prescribed protocol can be identified (because it fails to prove its correctness in zero-knowledge) and removed from the set of participants. There are very efficient (honest-verifier) zero-knowledge proofs to show the correctness of each protocol step (see Section 3.2).

The remainder of this paper is structured as follows. In Section 2, we review related work. Section 3 contains building blocks to be used in the protocols. Basic protocols consisting of these primitives are presented in Section 4 whereas more sophisticated protocols based on the evaluation of simple Boolean formulae

[1] Please note that such a third-party will not be able to learn any information besides the outcome.

[2] Threshold privacy can easily be obtained by using standard secret sharing techniques.

are proposed in Section 5. The paper concludes with a summary of the results and a brief outlook in Section 6.

2 Related Work

Various recent advances in efficiency for cryptographic protocols build on homomorphic encryption (*e.g.*, [JJ00, CDN01, Fis01, ST04, BGN05]). Most of these protocols ([JJ00] and [ST04] are exceptions) use factorization-based encryption schemes like Paillier encryption [Pai99] due to their versatility. However, these schemes require the joint generation of an RSA modulus before the actual computation begins. Even though protocols for this task improved over the years [BF97, Gil99, DK01, ACS02], they remain inefficient and unpractical, especially when having to tolerate active adversaries.[3] Distributed key generation for discrete logarithm based schemes like El Gamal, on the other hand, is straightforward and very efficient (see [Ped91, GJKR99, GJKR03]).

Our protocols bear some similarities to mix-and-match [JJ00] as distributed El Gamal random exponentiation and the mixing of ciphertexts are also part of the mix-and-match framework. However, in contrast to our work, mix-and-match allows *general* MPC. As a consequence, their approach is more versatile but less efficient in special cases. *E.g.*, a straightforward mix-and-match implementation of a millionaires' protocol is considerably less efficient in both round complexity and communication complexity than our protocol because mix-and-match does not take advantage of El Gamal's homomorphicity (except for the plaintext equality test).

[BST01] give a protocol for the *socialist millionaires'* problem in which two parties compute whether their inputs are equal. Security is based on the decisional Diffie-Hellman problem and a similar protocol (disregarding fairness) can be constructed by using the techniques presented in this paper (see Section 4.1).

Since the publication of Yao's original protocols in [Yao82], numerous solutions to the *millionaires' problem* have been proposed. While the complexity of some protocols is exponential in k [Sch96], others do not consider active adversaries [Fis01, IG03, LT05] or have quadratic complexity [IG03]. The protocol proposed in [Fis01] is quite efficient but also relies on the prior setup of a distributed RSA modulus which (even for just two parties) requires a large amount of communication. Nevertheless, we adopted [Fis01]'s idea of securely evaluating the greater-than function by shuffling k equality tests. The efficiency of Lin and Tzeng's recent protocol [LT05], which can be based on El Gamal encryption, is similar to that of our protocol, although it is based on a quite different, interesting idea. However, their model only allows passive adversaries and the protocol cannot straightforwardly be turned into one that is secure against active adversaries.

[3] Communication complexity in these protocols always contains high orders of the security parameter. For example, the 2-party key generation proposed in [Gil99] requires about 42MB of data to be sent for setting up a 1024-bit key (while only tolerating *passive* adversaries).

There are *randomized* protocols for the millionaires' problem (*e.g.*, [NN01, PBDL04]) whose communication complexity may be less than linear in the input size [NN01]. However, these protocols cannot provide constant round complexity.[4] The same holds for recently proposed protocols based on evaluating circuits that contain so-called conditional gates [ST04]. The communication complexity of [ST04]'s solution for the Millionaires' problem is comparable to ours but their protocol requires k rounds of interaction.

It is difficult to compare our techniques with advanced two-party protocols based on Yao's "garbled circuit" construction [Yao86], *e.g.*, [NPS99, CC00]. Naturally, these protocols have the advantage of being universally applicable. However, [NPS99] relies on the inefficient "cut-and-choose" technique to prove circuits correct and even [CC00], which uses zero-knowledge proofs instead of cut-and-choose, is less efficient than our approach because the correctness of every single gate has to be proven. Even for simple functions, like the greater-than function, the number of gates is relatively high (at least $5k - 4$ [KO02]), resulting in substantially higher complexity than our protocol for the same function. Moreover, our techniques have the advantage of being applicable to settings with any number of parties which is generally not possible with garbled circuit protocols.

3 Building Blocks

This section contains building blocks to enable the construction of efficient protocols for simple functions. In the heart of the system lies El Gamal encryption because it allows for the easy generation of distributed keys and because encrypted values can be exponentiated with a shared random number in a single round. This random exponentiation will be used as a blinding step in our protocols as it transforms every plaintext, *except* 1, into a meaningless random number.

We suggest the following general methodology for efficiently computing low-complexity functions: All parties publish encryptions of their inputs in a representation—*e.g.*, binary (see Section 5) or unary (see Section 4.3)—that at the same time allows efficient proofs of correctness and further processing in order to compute the function outcome. By exploiting the homomorphic property of the underlying encryption scheme, participants compute a vector of encrypted values that contains the function outcome but may also contain additional, unwanted information. In order to get rid of this information, all agents jointly execute random exponentiations for each vector component. Finally, if needed, components can be shuffled to only reveal if (or how many) vector components equal 1. We stress the fact the we do not rely on a strict Boolean or arithmetic representation of the function. We rather suggest a bottom-up approach, *i.e.*, trying to represent the function by using the mentioned limited set of primitives.

[4] Furthermore, the protocol in [PBDL04] is flawed because the proposed primitive "complex zero test" reveals statistical information.

3.1 El Gamal Encryption

El Gamal cipher [El 85] is a probabilistic and homomorphic public-key cryptosystem. Let p and q be large primes so that q divides $p-1$. $\mathbb{G}_q$ denotes $\mathbb{Z}_p^*$'s unique multiplicative subgroup of order q.[5] All computations in the remainder of this paper are modulo p unless otherwise noted. The *private key* is $x \in \mathbb{Z}_q$, the *public key* is $y = g^x$ ($g \in \mathbb{G}_q$ is an arbitrary, publicly known element). A message $m \in \mathbb{G}_q$ is *encrypted* by computing the ciphertext tuple

$$(\alpha, \beta) = (my^r, g^r)$$

where r is an arbitrary random number in $\mathbb{Z}_q$, chosen by the encrypter. A message is *decrypted* by computing

$$\frac{\alpha}{\beta^x} = \frac{my^r}{(g^r)^x} = m.$$

El Gamal is homomorphic as the component-wise product of two ciphertexts $(\alpha\alpha', \beta\beta') = (mm'y^{r+r'}, g^{r+r'})$ represents an encryption of the plaintexts' product mm'. It has been shown that El Gamal is semantical secure, *i.e.*, it is computationally infeasible to distinguish between the encryptions of any two given messages, if the decisional Diffie-Hellman problem is intractable [TY98]. We will use functions $E(\cdot)$ and $D(\cdot)$ to denote the encryption and the decryption of plain- and ciphertexts, respectively.

In the following, we describe how to apply the El Gamal cryptosystem as a fully private, *i.e.*, non-threshold, MPC scheme for n agents.[6] If a value represents an additive share, this is denoted by a "+" in the index, whereas multiplicative shares are denoted by "×". Underlying zero-knowledge proofs will be presented in the next section.

Distributed key generation [Ped91]: Each participant chooses x_{+i} at random and publishes $y_{\times i} = g^{x_{+i}}$ along with a zero-knowledge proof of knowledge of $y_{\times i}$'s discrete logarithm. The public key is $y = \prod_{i=1}^{n} y_{\times i}$, the private key is $x = \sum_{i=1}^{n} x_{+i}$. This requires n multiplications, but the computational cost of multiplications is usually negligible in contrast to exponentiations. Broadcast round complexity and exponentiation complexity of the key generation are $\mathcal{O}(1)$.[7]

Distributed decryption: Given an encrypted message (α, β), each participant publishes $\beta_{\times i} = \beta^{x_{+i}}$ and proves its correctness by showing the equality of logarithms of $y_{\times i}$ and $\beta_{\times i}$. The plaintext can be derived by computing $\frac{\alpha}{\prod_{i=1}^{n} \beta_{\times i}}$. Like key generation, decryption can be performed in a constant number of rounds, requiring n multiplications and one exponentiation.

[5] We will focus on multiplicative subgroups of finite fields here, although El Gamal can also be based on other groups such as elliptic curve groups.

[6] Please note that this multiparty scheme is limited in the sense that it does not allow the computation of *arbitrary* functions.

[7] Finding "unbiased" parameters p and q requires no extra communication in the random oracle model.

Random Exponentiation: A given encrypted value (α, β) can easily be raised to the power of an unknown random number $M = \sum_{i=1}^{n} m_{+i}$ whose addends can be freely chosen by the participants if each bidder publishes $(\alpha^{m_{+i}}, \beta^{m_{+i}})$ and proves the equality of logarithms. The product of published ciphertexts yields (α^M, β^M) in a single step. The computational cost is two exponentiations and $2n$ multiplications.

When adding a commitment round (*e.g.*, using [Ped91]) during key generation and random exponentiation, security of these primitives is evident. We presume that even without these commitment rounds, security is preserved (see Proposition 1).

3.2 Zero-Knowledge Proofs

In order to obtain security against *malicious* or so-called *active* adversaries, agents are required to prove the correctness of each protocol step. One of the objectives when designing the protocols presented in Sections 4 and 5 was to enable *efficient* proofs of correctness for protocol steps. In fact, the proposed protocols can be proven correct by only using so-called Σ-protocols which just need three rounds of interaction [Dam02, CDS94]. Σ-protocols are not known to be zero-knowledge, but they satisfy the weaker property of *honest-verifier* zero-knowledge. This suffices for our purposes as we can use the Fiat-Shamir heuristic [FS87] to make these proofs non-interactive. As a consequence, the obtained proofs are indeed zero-knowledge *in the random oracle model* and only consist of a single message.[8] We will make use of the following four Σ-protocols.

Proof of knowledge of a discrete logarithm. This is a classic Σ-protocol by Schnorr [Sch91]. Alice and Bob know v and g, but only Alice knows x, so that $v = g^x$.

1. Alice chooses z at random and sends $a = g^z$ to Bob.
2. Bob chooses a challenge c at random and sends it to Alice.
3. Alice sends $r = (z + cx) \mod q$ to Bob
4. Bob checks that $g^r = av^c$.

Alice needs to send $\log p + \log q$ bits.

Proof of equality of two discrete logarithms. When executing the previous protocol in parallel, the equality of two discrete logarithms can be proven [CP92]. Alice and Bob know v, w, g_1, and g_2, but only Alice knows x, so that $v = g_1^x$ and $w = g_2^x$.

[8] The additional assumption of a random oracle is only made for reasons of efficiency. Alternatively, we could employ non-interactive zero-knowledge proofs in the *common random string model* (see [DDO$^+$01] and references therein) to obtain non-interactiveness. However, it has become common practice to use secure hash functions like MD5 or SHA-1 as random oracles for practical applications.

1. Alice chooses z at random and sends $a = g_1^z$ and $b = g_2^z$ to Bob.
2. Bob chooses a challenge c at random and sends it to Alice.
3. Alice sends $r = (z + cx) \mod q$ to Bob
4. Bob checks that $g_1^r = av^c$ and that $g_2^r = bw^c$.

Alice needs to send $2 \log p + \log q$ bits. It is possible to show the equality of any polynomial number of discrete logarithms in parallel. Thus, for showing that the discrete logarithms of k values are equal, Alice only sends $k \log p + \log q$ bits.

Proof that an encrypted value is one out of two values. The following protocol was proposed by Cramer et al [CGS97]. Alice proves that an El Gamal encrypted value $(\alpha, \beta) = (my^r, g^r)$ either decrypts to 1 or to a fixed value $z \in \mathbb{G}_q$ without revealing which is the case, in other words, it is shown that $m \in \{1, z\}$.

1. If $m = 1$, Alice chooses r_1, d_1, w at random and sends (α, β), $a_1 = g^{r_1}\beta^{d_1}$, $b_1 = y^{r_1}\left(\frac{\alpha}{z}\right)^{d_1}$ and $a_2 = g^w$, $b_2 = y^w$ to Bob.
   If $m = z$, Alice chooses r_2, d_2, w at random and sends (α, β), $a_1 = g^w$, $b_1 = y^w$, $a_2 = g^{r_2}\beta^{d_2}$, and $b_2 = y^{r_2}\alpha^{d_2}$ to Bob.
2. Bob chooses a challenge c at random and sends it to Alice.
3. If $m = 1$, Alice sends d_1, $d_2 = c - d_1 \mod q$, r_1, and $r_2 = w - rd_2 \mod q$ to Bob.
   If $m = z$, Alice sends $d_1 = c - d_2 \mod q$, d_2, $r_1 = w - rd_1 \mod q$, and r_2 to Bob.
4. Bob checks that $c = d_1 + d_2 \mod q$, $a_1 = g^{r_1}\beta^{d_1}$, $b_1 = y^{r_1}\left(\frac{\alpha}{z}\right)^{d_1}$, $a_2 = g^{r_2}\beta^{d_2}$, and $b_2 = y^{r_2}\alpha^{d_2}$.

The total amount of bits Alice sends to Bob is $4 \log p + 4 \log q$.

Verifiable shuffle of k encrypted values. A shuffle is a rearrangement and reencryption of input ciphertexts. By proving such a shuffle correct, a party can verifiably rearrange a vector of ciphertexts without revealing the applied permutation. [Gro03] proposed a very efficient way of proving the correctness of a shuffle of El Gamal encryptions in honest-verifier zero-knowledge (in fact, the proof is shorter than the vector itself). As the proof is *public-coin* honest-verifier zero-knowledge, it can be executed in a single round in the random oracle model. Alice needs to send $k(\log p + \log q) + 6 \log p + 3 \log q$ bits to prove the correctness of a shuffle consisting of k ciphertexts. This primitive will be used as a 2-server mix-net in Section 5 in order to hide which component of a vector equals 1.

4 Basic Protocols

In order to demonstrate the applicability of the proposed techniques, we briefly sketch three 4-round protocols that compute the equality-, the OR-, and the maximum-function, respectively.

4.1 Socialist Millionaires' Protocol

Suppose Alice and Bob want to compute the equality function $f(b_1, b_2) = [b_1 = b_2]$. This problem is also known as the socialist millionaires' problem [BST01] and can be solved by executing the following protocol.[9]

- Round 1: Alice and Bob generate a distributed pair of El Gamal keys
- Round 2: Both parties publish El Gamal encryptions of their inputs: $E(b_1)$ and $E(b_2)$.
- Round 3: They jointly compute $A = \left(\frac{E(b_1)}{E(b_2)}\right)^M$ where M is a random number not known to Alice or Bob (see Section 3.1).
- Round 4: Both parties jointly decrypt A. If $D(A) = 1$, both inputs were equal. Otherwise, $D(A)$ is a meaningless random number.

4.2 Veto Protocol

A variation of the previous protocol can be used for veto voting (in other words, the Boolean OR-function): $f(b_1, b_2, \ldots, b_n) = \bigvee_{i=1}^{n} b_i$. Let $Y \in \mathbb{G}_q \backslash \{1\}$ be a publicly known constant. Now, each voter i submits $E(b_i)$ where b_i is 1 if voter i agrees with the issue at hand or Y if he does not agree. The correctness of each vote, *i.e.*, $D(E(b_i)) \in \{1, Y\}$, can be proven by using the zero-knowledge proof given in Section 3.2. Voters than jointly decrypt $\left(\prod_{i=1}^{n} E(b_i)\right)^M$ and only learn whether they unanimously agree or not. No other information is revealed, not even to any (strict) subset of agents.

4.3 Maximum Protocol

Consider a group of n parties that wants to compute the maximum of their private input values: $f(b_1, b_2, \ldots, b_n) = \max\{b_1, b_2, \ldots, b_n\}$. By using a *unary* representation of numbers, this task can be accomplished by the following protocol. Let $\{1, 2, \ldots, k\}$ denote the set of possible input values and let $Y \in \mathbb{G}_q \backslash \{1\}$ again be a publicly known constant. Each participant i publishes $E(b_{ij})$ where $b_{ij} = Y$ if $b_i = j$ or $b_{ij} = 1$ otherwise. Agent i can efficiently prove the correctness of his input by showing that $\forall j \in \{1, 2, \ldots, k\} : D(E(b_{ij})) \in \{1, Y\}$ (Section 3.2) and that $\prod_{j=1}^{k} E(b_{ij}) = E(Y)$ (Section 3.2). Then, all agents jointly compute

$$A_j = \left(\prod_{i=1}^{n} \prod_{d=j}^{k} E(b_{id})\right)^{M_j} \qquad \forall j \in \{1, 2, \ldots, k\}$$

where, as above, M_j are jointly created random numbers. For all j greater than the maximum, $D(A_j)$ is equal to 1. All other $D(A_j)$ are random numbers. Clearly, the drawback of this protocol is that its communication complexity is linear in k, *i.e.*, exponential in the length of bitstrings. Nevertheless, it can be practical for small k.

[9] Similar protocols previously appeared in various contexts, *e.g.*, password authentication key exchange or the "plaintext equality test" in [JJ00].

5 Counting Boolean Disjunctions of Literals

In this section, we will show how the primitives defined in Section 3 can be used to evaluate simple Boolean expressions. Consider n parties whose inputs are bitstrings b_i of length k. We define $E[b]$ to be an (El Gamal) encryption of bit b if $E[0]$ decrypts to 1 and $E[1]$ decrypts to any other number in $\mathbb{G}_q$:

$$D[E[b]] \in \begin{cases} \{1\} & \text{if } b = 0 \\ \mathbb{G}_q \backslash \{1\} & \text{otherwise} \end{cases}.$$

As in Section 4.3, let Y be an arbitrary, publicly known, fixed element of $\mathbb{G}_q \backslash \{1\}$. Before the actual protocol starts each agent publishes encryptions of his individual input bits so that

$$E[b_{ij}] = \begin{cases} E(1) & \text{if } b_{ij} = 0 \\ E(Y) & \text{otherwise} \end{cases}.$$

The correctness of inputs can be efficiently proven by showing that each ciphertext either decrypts to 1 or to Y without revealing which case holds (Section 3.2). Based on this representation, we can count the number of true Boolean disjunctions of literals and pairwise exclusive disjunctions of literals by computing

$$f(b_1, b_2, \ldots, b_n) = \#\left(\bigvee_r L_r \vee \bigvee_{s,t} (L_s \oplus L_t) \right) \tag{1}$$

where $L_r, L_s, L_t \in \{b_{ij}, \neg b_{ij}\}$ for all $i \in \{1, 2, \ldots, n\}$ and $j \in \{1, 2, \ldots, k\}$ and $\#$ is a *count operator* that counts the number of true expressions. The individual operators can be implemented as follows.

- **Negations**
  The Boolean negation of *input* bits can be computed by dividing Y by the input bit's encryption, *i.e.*,
  $$E[\neg b_{ij}] = \frac{Y}{E[b_{ij}]}.$$
- **Disjunctions**
  As in Section 4.2, the product of ciphertexts yields the logical OR of the corresponding plaintext bits:
  $$E\left[\bigvee_r L_r\right] = \prod_r E[L_r].$$
- **Exclusive Disjunctions**
  The exclusive OR of *pairs of literals* can be computed by dividing the encryptions of these bits,
  $$E\left[L_s \oplus L_t\right] = \frac{E[L_s]}{E[L_t]}.$$

When combining these exclusive ORs via the disjunction operator, it has to be made sure that two encryptions that represent $E[1]$ do not "accidently" multiply to $E[0]$. This can be achieved by raising the dth factor to the 2^{d-1}th power:

$$E\left[\bigvee_{d=1}^{k}(L_{s_d} \oplus L_{t_d})\right] = \prod_{d=1}^{k}\left(\frac{E[L_{s_d}]}{E[L_{t_d}]}\right)^{2^{d-1}} .$$

- **Count Operator**
  Finally, we can count the number of true bits in a vector of encrypted bits by consecutively letting each party verifiably shuffle its vector of bits (see Section 3.2), thus effectively emulating a mix-net. After exponentiating each component with a jointly created random number (as described in Section 3.1) and decrypting all components, the number of true bits is exactly the number of components not equal to 1.

5.1 Hamming Protocol

A simple function that can be expressed as an arithmetic formula fitting Equation 1 is the Hamming distance of two bitstrings. The Hamming distance is defined as the number of corresponding bits that are not equal. In other words,

$$f(b_1, b_2) = \#_{j=1}^{k}\left(b_{1j} \oplus b_{2j}\right).$$

An efficient 6-round protocol for computing this function can be designed based on the constructions proposed in the beginning of this section. For reasons of limited space, we just spell out the similar, but more sophisticated, protocol for computing the greater-than function in the following section.

5.2 Millionaires' Protocol

A protocol for the millionaires' problem can be obtained by reformulating the greater-than function $f(b_1, b_2) = [b_1 > b_2]$ to fit Equation 1. Let Alice's and Bob's inputs, b_1 and b_2, be k-bit numbers so that $b_i = \sum_{j=1}^{k} b_{ij} 2^{j-1}$, *i.e.*, the least significant bit is b_{i1} and the most significant bit is b_{ik}. Consider the following Boolean expression for computing $b_1 > b_2$ given in [Fis01].

$$[b_1 > b_2] \quad \Longleftrightarrow \quad \bigvee_{j=1}^{k}\left(b_{1j} \wedge \neg b_{2j} \wedge \bigwedge_{d=j+1}^{k}(\neg(b_{1d} \oplus b_{2d}))\right). \tag{2}$$

The outer disjunction is exclusive, *i.e.*, at most one of the k terms can be satisfied. By applying De Morgan's laws, the right expression in Implication 2 can be rewritten as

$$\bigvee_{j=1}^{k}\left(\neg\underbrace{\left(\neg b_{1j} \vee b_{2j} \vee \bigvee_{d=j+1}^{k}(b_{1d} \oplus b_{2d})\right)}_{B_j :=}\right).$$

Using the techniques proposed at the beginning of this section, the inner term B_j can be computed as follows.

$$E[B_j] = \frac{Y \cdot E[b_{2j}]}{E[b_{1j}]} \cdot \prod_{d=j+1}^{k} \left(\frac{E[b_{1d}]}{E[b_{2d}]} \right)^{2^d - 2} \quad .$$

Recall that the outer disjunction is *exclusive*, *i.e.*, counting the number of *false* B_j's will yield either 0 or 1. This implies that $b_1 > b_2$ holds if and only if $\#(B_j) = k - 1$. For this reason, the following procedure suffices: Alice sends a verifiable shuffle of all $E[B_j]$ to Bob who verifiably shuffles the resulting ciphertexts himself and sends them back to Alice. Finally, both parties raise each $E[B_j]$ to a jointly created random exponent M_j and decrypt all $(E[B_j])^{M_j}$. If any of these values equals 1, then $b_1 > b_2$, *i.e.*, Alice is richer than Bob. The detailed 6-round protocol is given in Figure 1.

In the remainder of this section, we use the millionaires' protocol as an example to analyze security and efficiency of our proposed techniques.

Proposition 1. *The millionaires' protocol is* correct *with negligible error probability and* secure *if the decisional Diffie-Hellman problem is intractable.*

Proof. (sketch)

Correctness: The protocol only fails when the random exponentiation for any outcome vector "accidently" yields a one, *i.e.*, $\sum_{h=1}^{n} m_{ij}^{+h} = 0 \mod q$ for any i and j. Due to the exponential size of $\mathbb{G}_q$ and the polynomial number of output components, the probability of this event is negligible. Error probability of the protocol is $(1 - (1 - 2^{-Q})^k) = \mathcal{O}(2^{-Q})$ where $Q = \log q$. The malleability of El Gamal encryption does not pose a problem because bidders prove that they know each plaintext using non-malleable zero-knowledge proofs.

Security: The security of El Gamal cipher as well as the applied zero-knowledge proofs can be based on the intractability of the decisional Diffie-Hellman assumption [TY98]. The security of *distributed* El Gamal cipher, in particular Pedersen's straightforward key generation [Ped91] which might result in non-uniformly distributed keys, follows from a recent argument by Gennaro et al [GJKR03]. Since encryption keys are essentially distributed by using 2-out-of-2 secret sharing, privacy can not be breached (unless Alice and Bob collude). □

We now investigate the computation and communication complexity of the millionaires' protocol. Typically, the computational cost of performing multiplications is negligible. Exponentiation and communication complexity are identical in the proposed protocol. Zero-knowledge proofs we apply in the protocol are non-interactive and have low constant overhead. Table 1 shows the communication complexity of each protocol step and also gives the complexity of the accompanying zero-knowledge proofs.

Depending on $i \in \{1, 2\}$, the directions address Alice ($i = 1$) or Bob ($i = 2$)).

Round 1: Generate public key

- Choose $x_{+i} \in \mathbb{Z}_q$ and $m_j^{+i}, r_{ij} \in \mathbb{Z}_q$ for each j at random.
- Publish $y_{\times a} = g^{x_{+i}}$ along with a zero-knowledge proof of knowledge of $y_{\times a}$'s discrete logarithm (Section 3.2).
- Compute $y = y_{\times 1} \cdot y_{\times 2}$.

Round 2: Encrypt input

- Publish $\alpha_{ij} = Y^{b_{ij}} \cdot y^{r_{ij}}$ and $\beta_{ij} = g^{r_{ij}}$ for each j.
- Prove that $\forall j : \log_g(\beta_{ij})$ equals $\log_y(\alpha_{ij})$ or $\log_y\left(\frac{\alpha_{ij}}{Y}\right)$ (Section 3.2)

Round 3: Mix output (1/2)

- Compute for each j:

$$\gamma_j = \frac{Y \cdot \alpha_{2j}}{\alpha_{1j}} \cdot \prod_{d=j+1}^{k} \left(\frac{\alpha_{1d}}{\alpha_{2d}}\right)^{2^d - 2} \quad \text{and} \quad \delta_j = \frac{\beta_{2j}}{\beta_{1j}} \cdot \prod_{d=j+1}^{k} \left(\frac{\beta_{1d}}{\beta_{2d}}\right)^{2^d - 2}$$

- Alice ($i = 1$): Verifiably shuffle k vectors (γ_j, δ_j) by index j (Section 3.2).

Round 4: Mix output (2/2)

- Bob ($i = 2$): Verifiably shuffle k vectors (γ_j, δ_j) by index j (Section 3.2).

Round 5: Randomize output

- Compute and publish $\gamma_j^{\times i} = (\gamma_j)^{m_j^{+i}}$ and $\delta_j^{\times i} = (\delta_j)^{m_j^{+i}}$ for each j with a proof of logarithm equality (Section 3.2).

Round 6: Decrypt output

- Publish $\varphi_j^{\times i} = \left(\delta_{ij}^{\times 1} \cdot \delta_{ij}^{\times 2}\right)^{x_{+a}}$ for each j with an accompanying proof of correctness (Section 3.2).
- Compute $v_j = \frac{\gamma_j^{\times 1} \cdot \gamma_j^{\times 2}}{\varphi_j^{\times 1} \cdot \varphi_j^{\times 2}}$ for each j. If $v_j = 1$ for any j, then b_1 is greater than b_2.

Fig. 1. Millionaires' Protocol

The total number of bits each party needs to communicate in the protocol is $(15k + 9)P + (6k + 5)Q$ where $P = \lceil \log p \rceil$ and $Q = \lceil \log q \rceil$. To achieve an appropriate level of security today, 1024 bits for p and 160 bits for q are reasonable settings. Then, in order to compare two 36-bit numbers,[10] each party only needs to send around 73 Kbytes of data.

[10] 36 bits are currently sufficient to compare the wealth of any given pair of human beings with a precision of one US dollar.

Table 1. Communication complexity (number of bits each party sends)

	Body	Zero-Knowledge Proofs
Round 1	P	$P+Q$
Round 2	$2kP$	$4k(P+Q)$
Round 3/4	$2kP$	$k(P+Q)+6P+3Q$
Round 5	$2kP$	$k(2P+Q)$
Round 6	kP	$(k+1)P+Q$
Σ	$(7k+1)P$	$(8k+8)P+(6k+5)Q$
Σ Body+ZK	$(15k+9)P+(6k+5)Q$	

$P = \lceil \log p \rceil$, $Q = \lceil \log q \rceil$

6 Conclusion

We have presented a set of primitives based on El Gamal encryption that can be used to construct efficient MPC protocols for certain low-complexity functions. Due to underlying efficient honest-verifier zero-knowledge proofs, the resulting protocols are secure against active adversaries. Security relies on the decisional Diffie-Hellman assumption. To demonstrate the applicability of the proposed techniques, we constructed protocols to compute the equality, the OR, the maximum, the Hamming and the greater-than functions. The latter requires only 6 rounds of interaction in the random oracle model while communication complexity is linear in the length of bitstrings to be compared, and error probability is exponentially small in the security parameter. To the best of our knowledge, this is the most efficient constant-round protocol for the greater-than function to date. The protocol can serve as a building block for the secure computation of more sophisticated functions such as the median [AMP04].

Future work includes the investigation of a more complete algebraic characterization of functions that can be efficiently computed using the proposed primitives. Furthermore, it might be possible to construct a sub-protocol for checking whether a vector of El Gamal ciphertexts contains an encrypted 1 which is considerably more efficient than consecutive mixing and decrypting.

Acknowledgements

Thanks to Jens Groth, Jesper Nielsen, Kobbi Nissim, and the anonymous referees for helpful comments. This material is based upon work supported by the Deutsche Forschungsgemeinschaft under grant BR 2312/1-1.

References

[ACS02] J. Algesheimer, J. Camenisch, and V. Shoup. Efficient computation modulo a shared secret with application to the generation of shared safe-prime products. In *Proc. of 22th CRYPTO Conference*, volume 2442 of *LNCS*, pages 417–432. Springer, 2002.

[AMP04] G. Aggarwal, N. Mishra, and B. Pinkas. Secure computation of the kth-ranked element. In *Proc. of 21st Eurocrypt Conference*, volume 3027 of *LNCS*, pages 40–55. Springer, 2004.

[BF97] D. Boneh and M. Franklin. Efficient generation of shared RSA keys. In *Proc. of 17th CRYPTO Conference*, volume 1294 of *LNCS*, pages 425–439. Springer, 1997.

[BFKR90] D. Beaver, J. Feigenbaum, J. Kilian, and P. Rogaway. Security with low communication overhead. In *Proc. of 10th CRYPTO Conference*, number 537 in LNCS, pages 62–76. Springer, 1990.

[BGN05] D. Boneh, E. Goh, and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. In *Proc. of 2nd Theory of Cryptography Conference (TCC)*, volume 3378 of *LNCS*, pages 325–341. Springer, 2005.

[BMR90] D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols. In *Proc. of 22nd STOC*, pages 503–513. ACM Press, 1990.

[BST01] F. Boudot, B. Schoenmakers, and J. Traoré. A fair and efficient solution to the socialist millionaires' problem. *Discrete Applied Mathematics*, 111(1-2):23–36, 2001.

[CC00] C. Cachin and J. Camenisch. Optimistic fair secure computation. In *Proc. of 20th CRYPTO Conference*, volume 1880 of *LNCS*, pages 93–111. Springer, 2000.

[CDN01] R. Cramer, I. Damgård, and J. B. Nielsen. Multiparty computation from threshold homomorphic encryption. In *Proc. of 18th Eurocrypt Conference*, volume 2045 of *LNCS*, pages 280–300. Springer, 2001.

[CDS94] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Proc. of 14th CRYPTO Conference*, volume 893 of *LNCS*, pages 174–187. Springer, 1994.

[CGS97] R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Proc. of 14th Eurocrypt Conference*, volume 1233 of *LNCS*, pages 103–118. Springer, 1997.

[CP92] D. Chaum and T. P. Pedersen. Wallet databases with observers. In *Proc. of 12th CRYPTO Conference*, volume 740 of *LNCS*, pages 3.1–3.6. Springer, 1992.

[Dam02] I. Damgård. On Σ-protocols. Lecture Notes, University of Aarhus, Department for Computer Science, 2002.

[DDO$^+$01] A. De Santis, G. Di Crescenzo, R. Ostrovsky, G. Persiano, and A. Sahai. Robust non-interactive zero knowledge. In *Proc. of 21th CRYPTO Conference*, volume 2139 of *LNCS*, pages 566–598. Springer, 2001.

[DK01] I. Damgård and M. Koprowski. Practical threshold RSA signatures without a trusted dealer. In *Proc. of 18th Eurocrypt Conference*, volume 2045 of *LNCS*, pages 152–165. Springer, 2001.

[El 85] T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985.

[Fis01] M. Fischlin. A cost-effective pay-per-multiplication comparison method for millionaires. In *Proceedings of the Cryptographers' Track at the 10th RSA Conference*, volume 2020 of *LNCS*, pages 457–472, 2001.

[FS87] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Proc. of 12th CRYPTO Conference*, LNCS, pages 186–194. Springer, 1987.

[Gil99] N. Gilboa. Two party RSA key generation. In *Proc. of 19th CRYPTO Conference*, volume 1666 of *LNCS*, pages 116–129. Springer, 1999.

[GJKR99] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *Proc. of 16th Eurocrypt Conference*, volume 1592 of *LNCS*, pages 295–310. Springer, 1999.

[GJKR03] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Applications of Pedersen's distributed key generation protocol. In *Proc. of Cryptographers' Track at the 12th RSA Conference*, volume 2612 of *LNCS*, pages 373–390. Springer, 2003.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *Proc. of 19th STOC*, pages 218–229. ACM Press, 1987.

[GMY04] J. Garay, P. MacKenzie, and K. Yang. Efficient and secure multi-party computation with faulty majority and complete fairness. To appear, 2004.

[Gro03] J. Groth. A verifiable secret shuffle of homomorphic encryptions. In *Proc. of 6th PKC Conference*, volume 2567 of *LNCS*, pages 145–160, 2003.

[IG03] I. Ioannidis and A. Grama. An efficient protocol for Yao's millionaires' problem. In *Proc. of 36th Hawaii International Conference on System Sciences (HICSS)*, pages 205–210. IEEE Press, 2003.

[IK00] Y. Ishai and E. Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *Proc. of 41st FOCS Symposium*, pages 294–304. IEEE Press, 2000.

[JJ00] M. Jakobsson and A. Juels. Mix and match: Secure function evaluation via ciphertexts. In *Proc. of 6th Asiacrypt Conference*, volume 1976 of *LNCS*, pages 162–177. Springer, 2000.

[Kil88] J. Kilian. Founding cryptography on oblivious transfer. In *Proc. of 20th ACM STOC*, pages 20–31. ACM Press, 1988.

[KO02] K. Kurosawa and W. Ogata. Bit-slice auction circuit. In *Proc. of 7th European Symposium on Research in Computer Security (ESORICS)*, volume 2502 of *LNCS*, pages 24–38. Springer, 2002.

[Lin01] Y. Lindell. Parallel coin-tossing and constant-round secure two-party computation. In *Proc. of 21st CRYPTO Conference*, volume 2139 of *LNCS*, pages 171–189. Springer, 2001.

[LT05] H.-Y. Lin and W.-G. Tzeng. An efficient solution to the Millionaires' Problem based on homomorphic encryption. In *Proc. of 3rd International Conference on Applied Cryptography and Network Security (ACNS)*, volume 3531 of *LNCS*, pages 456–466, 2005.

[NN01] M. Naor and K. Nissim. Communication preserving protocols for secure function evaluation. In *Proc. of 33rd STOC*, pages 590–599. ACM Press, 2001.

[NPS99] M. Naor, B. Pinkas, and R. Sumner. Privacy preserving auctions and mechanism design. In *Proc. of 1st ACM Conference on E-Commerce*, pages 129–139. ACM Press, 1999.

[Pai99] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proc. of 16th Eurocrypt Conference*, volume 1592 of *LNCS*, pages 223–238. Springer, 1999.

[PBDL04] K. Peng, C. Boyd, E. Dawson, and B. Lee. An efficient and verifiable solution to the millionaire problem. In *Proc. of 7th International Conference on Information Security and Cryptology (ICISC)*, volume 3506 of *LNCS*, pages 51–66. Springer, 2004.

[Ped91] T. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In J. Feigenbaum, editor, *Proc. of 11th CRYPTO Conference*, volume 576 of *LNCS*, pages 129–140. Springer, 1991.

[Sch91] C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[Sch96] B. Schneier. *Applied Cryptography.* John Wiley and Sons, Inc., 2nd edition, 1996.

[ST04] B. Schoenmakers and P. Tuyls. Practical two-party computation based on the conditional gate. In *Proc. of 10th Asiacrypt Conference*, number 3329 in LNCS, pages 119–136. Springer, 2004.

[TY98] Y. Tsiounis and M. Yung. On the security of ElGamal-based encryption. In *Proc. of 1st International Workshop on Practice and Theory in Public Key Cryptography (PKC)*, volume 1431 of *LNCS*, pages 117–134. Springer, 1998.

[Yao82] A. C. Yao. Protocols for secure computation. In *Proc. of 23th FOCS Symposium*, pages 160–164. IEEE Computer Society Press, 1982.

[Yao86] A. C. Yao. How to generate and exchange secrets. In *Proc. of 27th FOCS Symposium*, pages 162–167. IEEE Computer Society Press, 1986.