CED^2: Communication Efficient Disjointness Decision

Luciana Marconi[1], Mauro Conti[2], and Roberto Di Pietro[3]

[1] "Sapienza" Università di Roma, Department of Computer Science,
Roma 00198, Italy
marconi@di.uniroma1.it
[2] Vrije Universiteit Amsterdam, Department of Computer Science,
Amsterdam HV 1081, The Netherlands
mconti@few.vu.nl
[3] Università di Roma Tre, Department of Mathematics,
Roma 00146, Italy
dipietro@mat.uniroma3.it

Abstract. Enforcing security often requires the two legitimate parties of a communication to determine whether they share a secret, without disclosing information (e.g. the shared secret itself, or just the existence of such a secret) to third parties—or even to the other party, if it is not the legitimate party but an adversary pretending to impersonate the legitimate one. In this paper, we propose CED^2 (Communication Efficient Disjointness Decision), a probabilistic and distributed protocol that allows two parties—each one having a finite set of elements—to decide about the disjointness of their sets. CED^2 is particularly suitable for devices having constraints on energy, communication, storage, and bandwidth. Examples of these devices are satellite phones, or nodes of wireless sensor networks. We show that CED^2 significantly improves the communication cost compared to the state of the art, while providing the same degree of privacy and security. Analysis and simulations support the findings.

Keywords: sets disjointness test, communication complexity, privacy, security, probabilistic algorithms.

1 Introduction

Secure communications often require the involved parties to share a secret. As an example, two parties can use a pre-loaded shared symmetric key to encrypt the communication between them [4]. However, a problem arising in such scenarios is for the parties to determine whether they share such a secret. In this paper, we deal with this problem. In particular, we aim at minimizing the communication effort needed by the two parties to discover whether they share any common element from a given set. Note that this approach is mandatory where the communication cost is a driving system parameter. For instance, this is usually the case in satellite communications—where bandwidth can be limited or it has to be shared between multiple users at the same time—or when elements exchange is unfeasible because of the set or elements size.

Note that in this paper we present the problem of deciding whether there is an intersection between two sets for a security purpose. However, set intersection operations are required in a wide range of applications, particularly in the area of information integration across databases [1,9]. Further examples of applications are: finding common volumes between large libraries; finding common friends or interests in social networks without exchanging the corresponding lists; and, public welfare survey establishing how many welfare recipients are treated for a specific illness [1,9].

For the majority of these applications, there are often privacy and security concerns requiring the use of privacy-preserving techniques that usually relies on expensive asymmetric cryptographic primitives and a considerable communication cost [8,15]. However, there are scenarios that call for inexpensive cryptographic primitives and reduced communication cost, such as the ones cited before; these are the niche applications where CED^2 comes at hand.

Contribution. In this work, we propose CED^2 (Communication Efficient Disjointness Decision): a probabilistic and distributed protocol for deciding set intersection. We assume that each of the two parties of a communication has a set of secrets (or more generally, elements): the sets being $\mathcal{A}$ and $\mathcal{B}$, with elements from a domain $\mathcal{D}$. CED^2 is particularly concerned in minimizing the communication cost. Hence, focusing on communication complexity means to reduce the bits of information that the two parties exchange until at least one of them discovers whether $\mathcal{A} \cap \mathcal{B} = \emptyset$. The proposed solution does not require the parties to actually send any element of the set. CED^2 leads to a global communication saving compared to the state of the art—that, to the best of our knowledge, is represented by the algorithm by Kurtz and Manber [10] (we later refer to this algorithm as KM). Finally, this improvement is achieved providing the same level of privacy and security of KM—our solution does not disclose more information. The main contribution of this work is in the reduced communication overhead compared to KM, that is a building block protocol for many security settings where two parties need to know (efficiently) whether they share a secret.

Roadmap. The rest of the paper is organized as follows. Section 2 describes the related work in the area. Section 3 presents our solution, that is the CED^2 protocol. Section 4 provides the analysis of CED^2, while Section 5 is devoted to the protocol evaluation and comparison with the state of the art. Finally, Section 6 reports some concluding remarks.

2 Related Work

The literature on set intersection decision problem makes available a wide range of applications from set theory, combinatorial optimization, database searching, circuit complexity and applied cryptography [2,3,5,15]. In this paper, we focus on the evaluation of communication complexity of the disjointness problem. This problem is characterized as follows: two parties, Alice (A) and Bob (B), hold subsets $\mathcal{A}$ and $\mathcal{B}$ respectively, both of n elements from a given domain $\mathcal{D}$. Alice and

Bob follow a protocol to jointly decide whether they share some elements or not. That corresponds to the computation of the disjointness function $Disj(\mathcal{A}, \mathcal{B})$, defined as follows:

$$Disj(\mathcal{A}, \mathcal{B}) = \begin{cases} 1, & \text{if } \mathcal{A} \cap \mathcal{B} = \emptyset \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

The two parties do not know each other's input. To determine the output value, they alternatively exchange bits according to the protocol. In a deterministic protocol, their answer must always be correct, i.e., equal to $Disj(\mathcal{A}, \mathcal{B})$ for every input pair $\mathcal{A}, \mathcal{B}$. In a probabilistic protocol, the algorithms of the parties depend on unbiased coin tosses, and are required to be correct with a bounded probability (1/2) on every input. Interested readers can refer to [11] for a survey.

We briefly remind the notion of the communication complexity, introduced by Yao [14]. The communication complexity of a protocol P is the number of bits exchanged by the involved parties during the protocol run. In general, the communication complexity of a function f is that of the best possible protocol that computes f. The probabilistic communication complexity, denoted by $R(f)$, takes into account also the coin tosses used. Two access models to the random bits distinguish the private coin model, in which each party tosses his private coin, from the common randomness model, in which both parties share a common random bit string. It is known that $R(Disj)$ is $\Theta(n)$: the lower bound $\Omega(n)$ is given in [7], [11]; the upper bound corresponds for both parties of just sending all of their input. More recently, Håstad and Widgerson studied the communication complexity of the disjoint function [6]. They proved that in the model of common randomness:

- $R(Disj_n^k) = O(n)$, for all n;
- $R_0(Disj_n^k) = O(n)$, for instances of disjoint sets, and $R_0(Disj_n^k) = O(n + \log k)$ for not-disjoint sets. $R_0(f)$ represents the number of bits exchanged to compute f with the Las-Vegas type probabilistic algorithm, where the answer is required to be always correct (zero-error) [13].

$Disj_n^k(A, B)$ indicates the disjoint function of sets of size n whose elements are represented as bit strings of length k.

Thus computing $|\mathcal{A} \cap \mathcal{B}|$ requires $\Theta(n)$ communication. Therefore, even without taking any other requirement into consideration (e.g. privacy), the communication complexity of any set intersection algorithm is at least proportional to the input size. Moreover, Freedman et al. [5] showed a reduction from disjointness, proving that the communication cost of an approximation algorithm for the intersection size is lower-bounded by $\Omega(n)$.

The cited works study the formal properties of the disjointness problem considered as a communication problem. However, in order to evaluate the design of our solution and compare its performance, we consider a specific solution to the disjointness problem. This solution is provided by the algorithm from Kurtz and Manber appeared in 1987 [10]. The authors describe a distributed probabilistic algorithm that solves the disjointness problem in $O(\log_2 \log_2 n)$ rounds. The solution requires to exchange a message of $O(cn)$ bits at each round (where c is the number of bits ($O(\log_2 n)$) for the representation of the vector's indexes (see Section 4).

The basic idea of their solution is to reduce sets $\mathcal{A}$ and $\mathcal{B}$ at each round, eliminating all elements that are not in the intersection. The algorithm terminates when either i) no more elements are left—in which case the sets are guaranteed to be disjoint—or ii) when, with high probability, a set of candidates belonging to the intersection is left. The core of the KM solution is to use random hash functions taken from a pre-determined class of hash functions, to establish which elements are not in the intersection. The KM solution can be summarized as follows. At a generic round $i > 0$, the parties agree on a random hash function H_i. The agreement can be reached in several ways. For instance, we can assume that only one party chooses (uniformly at random) the function from the family, and then it sends a description of the function to the other party. Alternatively, if we suppose that the family of hash functions is an ordered set, one party can send to the other just the index of the selected function . Let be x a vector of size n with all values initialized to *false*, and let $\mathcal{A}^i$ denote the elements of $\mathcal{A}$ that are not eliminated after round i. A computes $H_i(a_s)$ for each $a_s \in \mathcal{A}^i$ and set $x[j] = x[j] \vee (H_i(a_s) = j)$. Note that $x[j]$ is *true* iff there exists at least one element $a_s \in \mathcal{A}^i$ such that $H_i(a_s) = j$ (i.e. a_s is hashed into the j^{th} position). The party B executes the same computation using a vector y. The corresponding vectors x and y (of length n) are then exchanged. This requires sending n bits. A can now eliminate all elements of $\mathcal{A}^i$ that were hashed into position j, such that $y[j] =$ *false*; B does the same for $\mathcal{B}^i$. Intuitively, this is equivalent to a bins and balls model where balls are the set elements and bins are the vector positions. Hashing is assumed to be equivalent to random throwing balls in bins (A throwing in vector x and B throwing in vector y). At each round, the algorithm eliminates all the balls for which the corresponding bin of the other party is an empty bin.

In our solution we use similar techniques. That is, the same probabilistic model (bins and balls) and the same simulation technique of the model (hashing). Exploiting a result on the bins and balls model contained in [12], we consider at each round only the maximum loaded bin. Thus, exchanging only one index at each round we build a protocol that computes the disjointness function with a global saving in the number of bits exchanged.

3 Our Solution: CED^2

In this section, we propose CED^2 (Communication Efficient Disjointness Decision), a communication efficient protocol for deciding whether there are elements in the intersection of two given sets. Section 3.1 introduces the system model and the notation used in the paper. Section 3.2 gives an overview of the proposed solution, while the protocol description can be found in Section 3.3.

3.1 System Model and Notation

Let us consider two sets $\mathcal{A} = \{a_1, a_2, ..., a_n\}$ and $\mathcal{B} = \{b_1, b_2, ..., b_m\}$, with $n, m \in \mathbb{N}$. We can assume w.l.o.g that $\mathcal{A}, \mathcal{B} \subseteq \{0, 1, ..., 2^k - 1\}$ with $k \in \mathbb{N}$ and $m = n$.

$\mathcal{A}$ and $\mathcal{B}$ are stored at two different parties, A and B respectively, that can exchange messages.

We would like to establish whether $\mathcal{A} \cap \mathcal{B} = \emptyset$ or not. Moreover, we would like to establish what is the communication cost payed in terms of total numbers of exchanged bits. Table 1 summarizes the notation used in this paper.

Table 1. Notation Table

$\mathcal{A}$	input set
$\mathcal{B}$	input set
A	(Alice); protocol party
B	(Bob); protocol party
$\mathcal{X}$	$\mathcal{A} \cap \mathcal{B}$
$\mathcal{H}$	family of hash functions
H_i	hash function randomly selected from $\mathcal{H}$ at round i
j_M	index of the max loaded bin
$\mathcal{A}^i$	elements not eliminated from $\mathcal{A}$ after round i
$\mathcal{B}^i$	elements not eliminated from $\mathcal{B}$ after round i
$U^i_{\mathcal{A}}[j]$	elements from $\mathcal{A}$ hashed, at round i, in position j
$U_{\mathcal{A}}[j]$	configuration of j-th bin of A at a generic round
$U^i_{\mathcal{B}}[j]$	elements from $\mathcal{B}$ hashed, at round i, in position j
$U_{\mathcal{B}}[j]$	configuration of j-th bin of $\mathcal{B}$ at a generic round
$\log n$	natural logarithm
$\log_2 n$	base 2 logarithm
$R_0(f)$	communication complexity of a Las-Vegas probabilistic protocol for function f
$R(f)$	communication complexity of a Monte Carlo probabilistic protocol for function f
$Disj^k_n$	the disjointness function: $Disj^k_n(A, B) = 1$ iff $\mathcal{A} \cap \mathcal{B} = \emptyset$, $\mathcal{A}$ and $\mathcal{B}$ having n elements
KM	Kurtz and Manber algorithm [10]

3.2 Protocol Overview

CED^2 works through different rounds. In particular, similarly to KM [10], the idea of CED^2 is to reduce the sets $\mathcal{A}$ and $\mathcal{B}$ at each subsequent protocol round. CED^2 also uses the bins and balls concept and, at the beginning of each protocol step, the remaining set's elements (balls) are assigned to bins accordingly to a hash function—different for each round. The basic idea is to eliminate, at each round, elements that are not in the intersection, exchanging the minimum number of bits for this purpose. To achieve the goal, we focus only on one particular bin at each round, the most loaded one. This allows us to cut the maximum number of balls possible with the minimum communication cost (just one index at each time).

Using this technique, we can save a significant amount of communication in the single round, as shown later in the paper. In fact, we exchange a single index (the one of the bin with the max load) instead of the whole vector (pairs index, load), as done by KM. The simple fact that the cost of a single round is less than the one of a single round of KM does not directly implies that the overall cost of CED^2 is less than the one of KM. In fact, the overall cost depends also on the number of steps, that it is different for the two considered protocols. Analysis and the experimental results show that CED^2 outperforms KM.

3.3 Protocol Description

CED^2 can be described via the bins and balls model. At every round i, we assign (throw) the n elements (balls) of each set in n indexes of a vector (bins). The launches are simulated by a hash function H_i, mapping elements of the sets into the vector indexes. In the following, the two vectors $U_{\mathcal{A}}$ and $U_{\mathcal{B}}$ denote bins, while $U^i_{\mathcal{A}}[j]$ ($U^i_{\mathcal{B}}[j]$) denotes the set of values $a_i \in \mathcal{A}$ ($b_i \in \mathcal{B}$) mapped to the j-th position at round i. At each round, the parties agree on the hash function H_i—chosen uniformly at random from a family $\mathcal{H}$ of hash functions. In particular, we consider $\mathcal{H} = \{H \equiv \lceil ax + b(modp) \rceil (modn)\}$, where $a, b < p$ ($a \neq 0$) are chosen at random, p is a prime $> 2^k$, and n is the sets cardinality—we remind that we assume that the two sets have the same cardinality. We denote with H_i the hash function randomly selected at round i. Furthermore, $\mathcal{A}^i$ and $\mathcal{B}^i$ denote the elements not eliminated after round i, from $\mathcal{A}$ and $\mathcal{B}$ respectively. Notation $H_i(a_k) = j$ indicates that the item $a_k \in \mathcal{A}$ has been assigned to the bin j, for the round i. Each of the parties involved in the protocol computes the assignment independently—without requiring any communication. However, using both parties the same hash function (even if different for each round) guarantees that elements belonging to $\mathcal{A} \cap \mathcal{B}$ map to the same position. Let us assume that for a given j, $U_{\mathcal{A}}[j]$ contains v elements (of $\mathcal{A}$) and the corresponding bin (same vector's index j) $U_{\mathcal{B}}[j]$ is empty. This assure that the v elements of $\mathcal{A}$ mapped into $U_{\mathcal{A}}[j]$ do not belong to the intersection $\mathcal{A} \cap \mathcal{B}$. From the hash function definition we have the following two properties:

$$a_s = b_t \in \mathcal{A} \cap \mathcal{B} \Longrightarrow H_i(a_s) = H_i(b_t) = j \tag{2}$$

$$H_i(a_s) = H_i(b_t) = j \not\Longrightarrow a_s = b_t \in \mathcal{A} \cap \mathcal{B}. \tag{3}$$

Equation 3 justifies the need for using different hash functions in the subsequent protocol rounds. In fact, let us assume that, at a given round, the randomly selected hash function induces a configuration of the bins such that bin $U_{\mathcal{A}}[j]$ contains elements of $\mathcal{A}$ that are not in $\mathcal{A} \cap \mathcal{B}$, and bin $U_{\mathcal{B}}[j]$ contains elements of $\mathcal{B}$ that are not in $\mathcal{A} \cap \mathcal{B}$ (hash collisions). Intuitively, changing the hash function at the next round gives a different distribution for the balls in the bins. Thus, we have a chance to eliminate the balls that do not belong to the intersection—using randomly chosen functions at each round permits to have launches behave differently at each round.

The behaviour of CED^2 is described in Algorithm 1. First, A randomly select the hash function used in the current iteration i (line 1). Hence, A maps all the elements $\mathcal{A}$ into the n elements vector, using the hash function (line 2). Then, A

sends to B the vector index, j_M, with the maximum number of elements mapped into (line 3). B sends back to A the information whether its own vector is empty at position j_M. If this is the case (line 5), A cuts all the elements mapped in j_M (line 6). Then, it checks whether there are remaining elements (line 7). In the negative case, A can conclude that the intersection is empty, and terminate with output 1 (line 8). Otherwise, if the B vector is not empty at position j_M (line 10), A checks for how many are the consecutive rounds it was not able to eliminate elements (line 11). If these consecutive rounds are more than a pre-determined constant q (its value is discussed in Section 5), A terminates with output 0 (line 12). If neither of the two termination conditions are verified (lines 8, 12), A iterates the procedure (line 14).

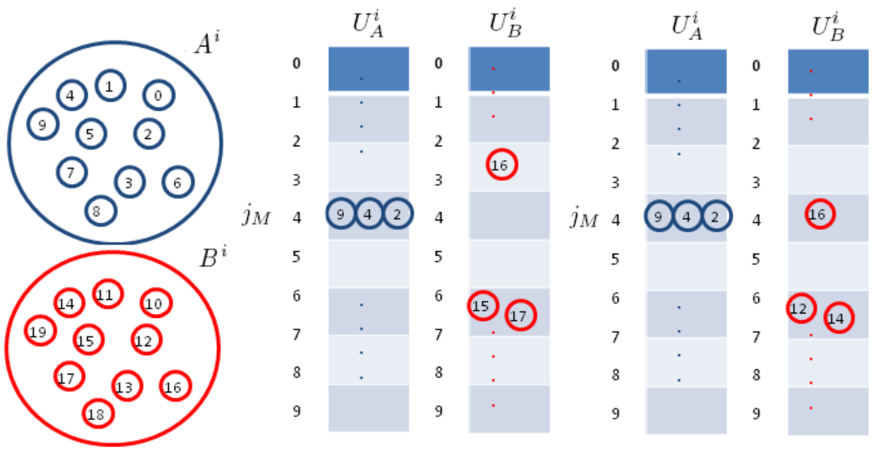
Algorithm 1. CED^2

Round i; computation made by A

```
 1: $A$ chooses a random $H_i$ from $\mathcal{H}$ and sends a description of $H_i$ (i.e. function
    parameters $a$, $b$, $p$) to $B$
 2: $A$ computes $H_i(a) = j$ for each $a \in \mathcal{A}^{i-1}$ and stores $a$ in $U^i_{\mathcal{A}}[j]$
 3: $A$ sends to $B$ the $j_M$ index, the maximum loaded bin
 4: $A$ receives from $B$ the information whether the $U^i_{\mathcal{B}}[j_M]$ is empty or not
 5: if $U^i_{\mathcal{B}}[j_M] = \emptyset$ then
 6:   $\mathcal{A}^i = \mathcal{A}^{i-1} \setminus \{a_s \mid a_s \in \mathcal{A}$ and $H_i(a_s) = j_M\}$
 7:   if ($\mathcal{A}^i = \emptyset$) then
 8:     output 1: Disjoint
 9:   end if
10: else
11:   if $i$ satisfy the condition $\mathcal{A}^{i-q} = \mathcal{A}^{i-q+1} = ... = \mathcal{A}^i$ then
12:     output 0: Not-Disjoint
13:   else
14:     $i = i + 1$; throwing again
15:   end if
16: end if
```

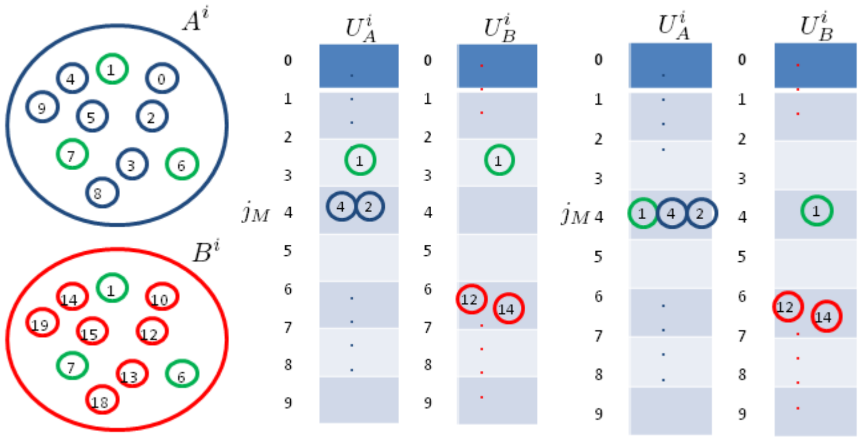
In figures 1 and 2, we depict the two possible scenarios for CED^2: disjoint sets (Figure 1) and not-disjoint sets (Figure 2). For the sake of clarity we do not show the configuration of the entire bins vectors $U^i_{\mathcal{A}}$ and $U^i_{\mathcal{B}}$ but just a sample of them.

Figure 1 shows an example of disjoint sets instances, presented in Figure 1a. Considering these sets, an example of a cutting round is shown in Figure 1b and a not cutting round in Figure 1c. Looking at Figure 1b, let us suppose that position 4 of the A vector is the maximum loaded bin j_M. We can observe that the corresponding position in the vector of B ($U^i_{\mathcal{B}}[4]$) is empty. Thus, at the subsequent round $i+1$, $\mathcal{A}^{i+1} = \{0, 1, 3, 5, 6, 7, 8\}$. Instead, in the case depicted in Figure 1c the corresponding position in the vector of $\mathcal{B}$ is not empty. In fact, $U^i_{\mathcal{B}}[4]$ contains the element labeled 16. Hence, at the subsequent round $i+1$, $\mathcal{A}^{i+1} = \mathcal{A}^i = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.



(a) Sets $\mathcal{A}$ and $\mathcal{B}$ (b) a cutting round (c) a not cutting round

Fig. 1. Example of Disjoint Instances



(a) Sets $\mathcal{A}$and $\mathcal{B}$ (b) a cutting round (c) a not cutting round

Fig. 2. Example of Not-Disjoint Instances

Similarly to Figure 1, Figure 2 shows an example of two not-disjoint sets, presented in Figure 2a; the green balls indicating elements belonging to the intersection. Considering this case, a cutting launch is shown in Figure 2b and a not cutting launch in Figure 2c. In Figure 2b, the position 4 is the maximum loaded bin j_M of $\mathcal{A}$, it does not contain any intersection element and the corresponding bin $U^i_{\mathcal{B}}[4]$ is empty. Hence, at the subsequent round $i+1$, $\mathcal{A}^{i+1} = \{0, 1, 3, 5, 6, 7, 8, 9\}$. Instead, in the case depicted in Figure 2c the green ball labeled 1 makes $U^i_{\mathcal{B}}[4]$ not empty. This produces, at the subsequent round $i+1$, $\mathcal{A}^{i+1} = \mathcal{A}^i = \{0,1,2,3,4,5,6,7,8,9\}$. In fact, the green ball belongs to the intersection and falls in the bin $j_M = 4$.

4 Analysis

In this section, we analyze the communication cost of CED^2 and KM. The analysis given in this section only considers the scenario where the two sets are actually disjoint.

We start analyzing the cost for CED^2. The overall result is given in Lemma 4. However, we need the following intermediate results: Lemma 1, Lemma 2 and Lemma 3. Lemma 1 is a result from [12].

Lemma 1. *When n balls are thrown into n bins, the maximum number of balls in any bin is $O(\frac{\log n}{\log\log n})$ with high probability, i.e., $1-\frac{1}{n}$.*

Lemma 2. *When n balls are thrown into n bins the probability of a particular bin being empty is $\frac{1}{e}$ for large n.*

Proof. The probability that a ball does not fall into a particular bin is $1-\frac{1}{n}$. Therefore: $Pr[bin\ j\ is\ empty] = \left(1-\frac{1}{n}\right)^n \approx \frac{1}{e}$. □

Lemma 3. *Given two sets $\mathcal{A}$ and $\mathcal{B}$ with $|\mathcal{A}|=|\mathcal{B}|=n$, if $\mathcal{A}\cap\mathcal{B}=\emptyset$ then the average number of messages exchanged for CED^2 (Algorithm 1) is $O(n\cdot\frac{\log\log n}{\log n})$.*

Proof. From Lemma 1, we know that if at each round it is possible to cut $\frac{\log n}{\log\log n}$ elements. Hence, the expected number of rounds to cut all the elements is $n\cdot\frac{\log\log n}{\log n}$. The expected trials to find empty a given box j is e (Lemma 2). Thus, the total expected number of rounds needed to cut all the elements is $e\cdot n\cdot\frac{\log\log n}{\log n}$. □

We can now give the following Lemma for CED^2.

Lemma 4. *Given two sets $\mathcal{A}$ and $\mathcal{B}$ with $|\mathcal{A}|=|\mathcal{B}|=n$, if $\mathcal{A}\cap\mathcal{B}=\emptyset$ then the average number of bits exchanged for CED^2 is $O(n\cdot\log\log n)$.*

Proof. From Lemma 3, we know that $e\cdot n\cdot\frac{\log\log n}{\log n}$ is the expected number of rounds required by CED^2 to cut all the elements. At each round CED^2 exchanges a message of $O(\log_2 n)$ bits. Indeed, each party sends one single index using $\log_2 n$ bits for its representation. Hence, the expected CED^2 total bits expenditure is:

$$e\cdot n\cdot\frac{\log\log n}{\log n}\cdot\log_2 n. \tag{4}$$

Substituting $\log_2 n = \frac{\log n}{\log 2}$ in (4), the claim follows. □

The communication complexity of KM is given by the following Lemma, provided in [10].

Lemma 5. *Given two sets $\mathcal{A}$ and $\mathcal{B}$ with $|\mathcal{A}| = |\mathcal{B}|=n$, if $\mathcal{A}\cap\mathcal{B}=\emptyset$ then the average number of bits exchanged for KM is $O(n(\log_2 n)(\log_2\log_2 n))$.*

Comparing Lemma 4 to Lemma 5, we conclude that CED^2 communication complexity (Lemma 4) is lower than KM communication complexity (Lemma 5) by a factor $O(\log_2 n)$.

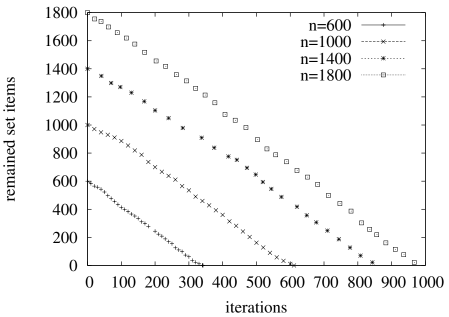
5 Protocol Evaluation

In the previous section we have provided the analysis of both CED^2 and KM for the disjointness case. In this section, we evaluate our solution leveraging both the previous analysis and the results of the simulations we run. We consider both disjointness and not-disjointness cases—and discuss them separately. We compare our solution to the one of Kurtz and Manber that, to the best of our knowledge, is the most efficient solution in the literature. In order to run the simulations shown in this section, we implemented a simulator using Python. The inputs sets are generated as random integers using the Python libraries for randomness. The same libraries have been also employed to implement the family of hash functions described both for CED^2 and KM. Each point plotted in the graphs shown in this section represents the average computed over 500 run of the algorithms.

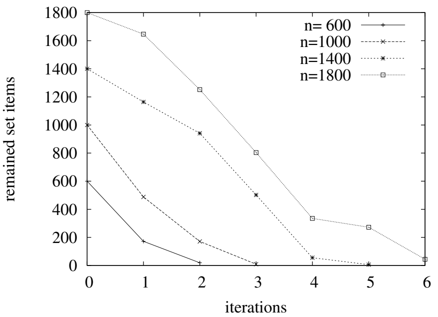
5.1 Disjoint Sets Instances

We first consider disjoint sets instances as input for the algorithms. Both algorithms work over subsequent rounds by removing elements ("cutting") from the starting sets. The two algorithms both terminate when all elements are removed. Hence, we start investigating how the size of the remaining sets vary with the number of rounds. The results are shown in figures 3a and 3b for CED^2 and KM, respectively. We specify that data for Figure 3b has been obtained fixing a value for the number of rounds, and calculating the average number of elements cut (y-axis value) for that value. Comparing the results for the two protocols, we observe that CED^2 requires a significant higher number of rounds to complete with respect to KM. As an example, for sets of $n = 1000$ elements, CED^2 requires some 600 protocol rounds to complete, while KM terminates in 3 iterations.

From just these results, one might conclude that the communication cost of CED^2 is higher than the one of KM. However, we still need to investigate what



(a) CED^2



(b) KM

Fig. 3. Disjoint Sets: Cuts Trend

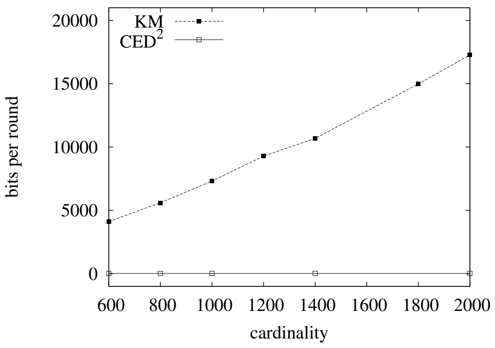


Fig. 4. Disjoint Sets: Average Bits per Round

is the communication cost of each single protocol iteration. Hence, we run our simulator to collect the number of bits exchanged in each single protocol round and we considered the average of all the rounds. The results are shown in Figure 4.

From Figure 4, we observe that the key difference between the two algorithms is the communication bits payed in each single iteration. In fact, while CED^2 requires more rounds than KM (figures 3a and 3b), each round has a very small communication cost—almost negligible compared to a round of KM. In particular, for the sets cardinality considered in the simulation (x-axis of Figure 4), the communication cost of CED^2 varies from 12 to 15 bits per round. Whereas, the cost of KM varies from 4113 bits (for sets of 600 elements), up to some 17276 bits (for sets of size 2000).

We note that "cutting" all the elements provides an answer to the question whether the sets intersect. Hence, this is a zero-error termination criteria for both algorithms. Unfortunately, we do not have the same characterization for the not-disjoint sets, as discussed in Section 5.2.

In the following graph (Figure 5) we report the overall communication complexity for the two protocols; that is, the number of bits required for the two protocols to complete. More specifically, in Figure 5, we give a global view of analysis and simulations for both algorithms showing:

- the expected behaviour from analysis, respectively from Lemma 4 for CED^2 and from Lemma 5 for KM;
- the results of simulations of the two algorithms.

Observing the results shown in Figure 5, we can draw the conclusion that CED^2 (Algorithm 1) performs better than KM algorithm. We underline that the aim of these results is to show a qualitative behaviour without taking into consideration any specific communication protocol. In fact, one might argue that when our solution is used in a practical scenario, the size of the exchanged message might be bigger than the one shown in the figure—due to the message header of the communication protocol used. However, we observe that the advantage provided

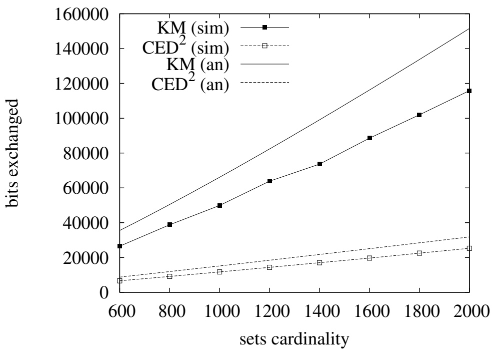


Fig. 5. Disjoint Sets: Analysis (an) and Simulations (sim) of Communication Complexity

by our solution is preserved even when a practical message header is considered. For example, let us consider sets of 1400 elements. Our solution would require an average of 843 messages (see Figure 3a) each of 15 bits of payload (Figure 4). The KM solution would require an average of 5 messages (see Figure 3b), each of 73654 bits of payload (Figure 4). Let us consider a message header of 10 bits—that is a practical choice for setting like WSN, where the header can have a small size that includes the bits required to identify the receiver (e.g. among some 1000 nodes). With 10 bits of header our solution would send about 843 messages, each of 25 bits, while KM would use 5 messages of 73664 bits. Hence, the overall number of bits would be 21075 for our solution, compared to the 368320 required by KM. This example shows that the advantage of our solution compared to KM remains even in practical scenario.

The simulations confirmed that the number of bits sent in the two solutions differs of a $O(\log_2 n)$ factor, as predicted in Lemma 4 and Lemma 5. We note that KM analysis gives an upper bound on the number of exchanged bits based on the $O(\log_2(n))$ bits representation for bins indexes (see Lemma 5). Running the KM simulation produces averaged values for indexes representation expenditure, $\lceil \frac{\log_2(n)+1}{2} \rceil$. This motivates the gap between the two KM curves. Similar argument justifies the fact that indexes representation cost in bits does not affect our solution. In fact, as each party in the protocol sends just one index at a time, the upper bound from analysis (see Lemma 3) and the result of the simulation are likely to be close to each other. We also observe that in the experiment shown in Figure 5 we used an optimized implementation of the KM algorithm. In fact, we send just empty bins indexes to the other party, saving on the total transmitted bits. Actually, the KM algorithm would send the entire vector of size n. In the latter case, the difference between the two compared communication costs would be larger. The conclusion is that in case of disjoint sets instances, choosing to reduce the exchanged bits in the single round provides a global saving in the total communication complexity.

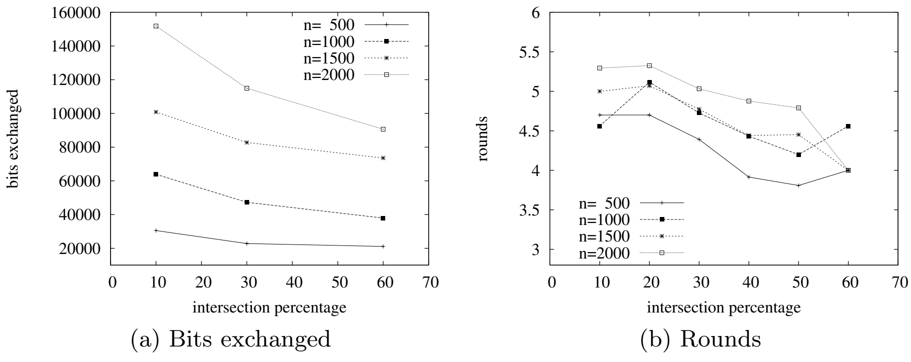


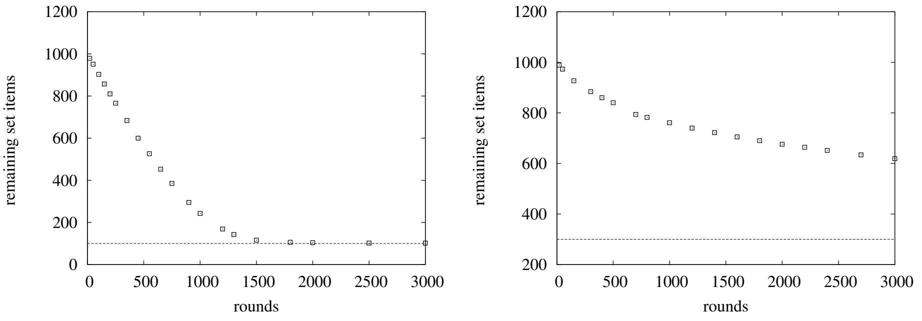
Fig. 6. Not-Disjoint Sets: KM; $q = 2$

5.2 Not-disjoint Instances

For not-disjoint instances, let us first analyze KM termination that authors left as an open issue. In KM the termination condition occurs at round r when $\mathcal{A}^{r-q} = \mathcal{A}^{r-q+1} = ... = \mathcal{A}^{r}$, for a pre-determined constant q (not specified in [10]). This condition is similar to the one used in CED^2 (see Algorithm 1, line 11), but the value of q is influenced by different underneath stochastic processes. In fact, even though the two algorithms use the same hashing strategy, the stochastic process produced by the choice of the maximum loaded bin at each round (CED^2), is not equal to the one produced by the exchange of all the empty bin indexes (KM).

We argue that $q = 2$ would be a good choice for KM to establish if two sets are disjoint or not. In fact, KM algorithm is likely to cut elements at each round. The only adverse circumstance is the configuration in which all not empty bins indexes, for one party, match all not empty bins indexes for the other party. As it is unlikely that KM algorithm does not cut elements at a given round, if for two rounds it is not possible to cut elements, this means that, with high probability, KM cut all the elements not belonging to the intersection. We report in Figure 6 both the amount of exchanged bits (Figure 6a) and the total rounds employed (Figure 6b) with this termination criteria. From Figure 6b, we can observe that the numbers of rounds employed to end, for all the cardinalities considered, vary in a short range from some 3.8 for $n = 500$ and 50% of common elements between the input sets, up to some 5.4 for $n = 2000$ and 20% of common elements. From Figure 6a, we can observe that the bits expenditure is ranged between some 22000 bits, obtained for $n = 500$ up to nearly 160000 for $n = 2000$. The second hypothesis from KM is to run the algorithm for $c \cdot (\log_2 \log_2 n)$ rounds, with constant $c > 1$. If not all the elements are eliminated, then the sets are not disjoint with an error probability depending on c.

For comparisons with CED^2 we choose this latter hypothesis and set $c = 1.1$ for the simulations. The reason is twofold: on the one hand, as we are focusing



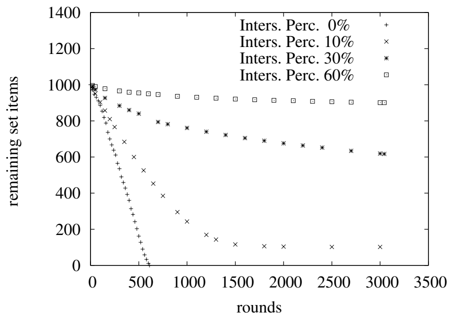
(a) Intersection Percentage 10% (b) Intersection Percentage 30%

Fig. 7. Not-Disjoint Sets: CED^2; Cuts Trend with respect to Intersection Cardinality; $n = 1000$

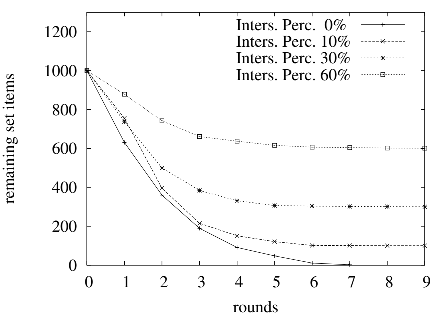
on the amount of exchanged bits, we are interested in the scenario in which KM saves more—setting $c = 1.1$ means to have the minimum rounds and thus the minimum bits expenditure for KM. On the other hand, the approach to cut all the elements not belonging to the intersection (like in the first KM hypothesis) is not applicable to CED^2. In fact, for not-disjoint instances, if CED^2 reaches, at some round, a configuration where elements not in the intersection are less than the bin maximum load (see Lemma 1), it will never be able to cut those elements (it would be impossible to find them in the maximum loaded bin). Even if CED^2 may not be able to cut all the elements not belonging to the intersection, still the cuts trend shows meaningful information.

We start observing that the convergence rate to the intersection cardinality is slow (see Figure 7) and goes slower as the intersection cardinality increases. The phenomenon is captured in figures 7a and 7b where we compare the cuts trend—using different intersection percentage—to the real intersection size.

The behaviour of the curves can be explained by observing that the possibility to cut elements, at a certain round, depends on the probability that: no intersection elements fall in the max loaded bin and the corresponding bin of the other party is empty. It is possible to check that this probability is $\leq e^{-\left(\frac{|\mathcal{X}|}{n}+1\right)}$ and that this is congruent with the simulations results depicted in Figure 7. We also observe that, for all the curves in Figure 7, increasing the protocol round number (x-axis), the cardinality of the remaining set (y-axis) decreases slowly than the case for the empty intersection (or disjointness) curve—this is shown in Figure 3a (see the curve for $n = 1000$). This behaviour can be observed even for a small intersections size (e.g. 10% of common elements; see Figure 7a). A direct comparison of these curves can be found in Figure 8a. Looking at Figure 8b, we can check that a similar phenomenon can be recognized in KM, even if it appears a bit less marked. We test CED^2 termination simulating the algorithm with $q = 11$. That is, if for 11 consecutive iterations it is not possible to cut elements, then we conclude that the intersection is not empty and terminate the execution. From simulations results, the choice of the value q resulted to be a good one. In fact (see Figure 10), setting $q = 11$ produces an error rate less than 20%. That is, we obtain the correct answer with a probability $> 4/5$. The optimal value for q and the related error rate appears to be an interesting matter for further investigations.



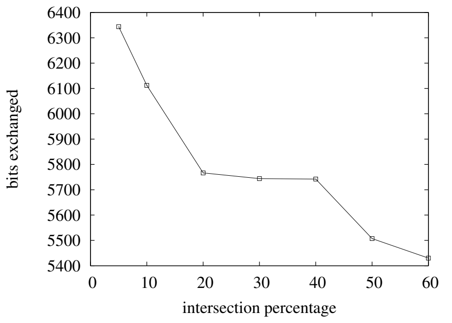
(a) CED^2



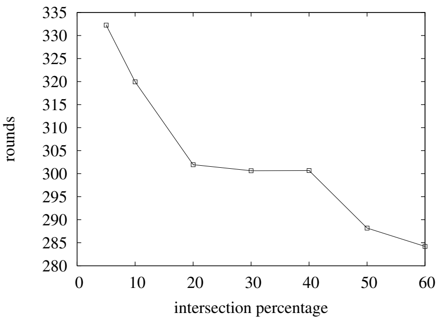
(b) KM

Fig. 8. Not-Disjoint Sets: Cuts Trend; $n = 1000$

Figure 9 shows iterations (see Figure 9b) and communication bits (see Figure 9a) reported by CED^2 simulations to decide if the two sets in input are not disjoint. Since, we want also to be able to avoid errors when disjoint instances are provided as input, we tested the two protocols providing in input to our simulator 50% of disjoint sets input pairs and 50% of not disjoint sets. In such a testing environment, CED^2 terminated with a wrong decision on 4.21% of the input pairs. This value is not plotted in Figure 9.



(a) Bits exchanged



(b) Rounds

Fig. 9. Not-Disjoint Sets: CED^2; $n = 1000$; $q = 11$

Figure 10 compares CED^2 and KM simulations results using the termination criteria discussed above. In both cases, fixed a protocol parameter (q for CED^2

and c for KM) we can observe that the error rate increases as the cardinality of the sets increases. On the KM side, it is possible to check that setting $c = 1.1$ is equivalent to set the number of rounds for termination to 4. This can be verified substituting the values of the cardinality tested in the termination condition. As we can see in Figure 3b, the average number of rounds required to cut all the elements is at least 6 for $n = 2000$, while it is just 3 considering $n = 1000$. This justifies why, with 4 rounds, KM makes more errors for $n = 2000$ than for $n = 1000$ on disjoint sets instances. On the CED^2 side, the probability to find empty the bin corresponding to the max loaded one at round i, is $e^{-\frac{|\mathcal{A}^i|}{n}}$. This value decreases as n increases. This justifies why, as the sets cardinality increases, the number of consecutive launches before cutting elements also increases. As a consequence, also CED^2 makes more errors for $n = 2000$ than for $n = 1000$.

From the performances perspective, Figure 10 shows that even if we consider the scenario with the minimum bits expenditure for KM (i.e. $c = 1.1$), still CED^2 obtains a lower error rate. In fact, in the majority of the cardinality considered, with a single exception, CED^2 requires much less communication bits. This allows us to conclude that the CED^2 strategy to consider a single bin at each round produces a global saving in the communication bits expenditure for both disjoint and not-disjoint input instances.

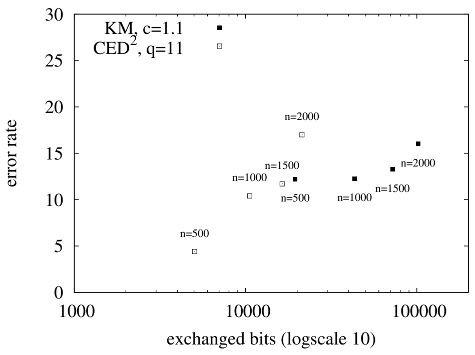


Fig. 10. Not-Disjoint Sets: KM and CED^2 Relation Between Communication Cost and Error Rate

6 Conclusions

In this paper we presented CED^2 (Communication Efficient Disjointness Decision), a probabilistic and distributed protocol that allows two parties to decide about whether they share a secret. CED^2 has been showed to be particularly suitable for devices having constraints on energy, communication, storage, and bandwidth. In particular, CED^2 significantly improves the communication cost compared to the work in the literature, having a communication complexity of $O(n \log\log n)$—improving by $O(\log_2 n)$ the state of the art. While in this paper we focused on the (probabilistic) discovery of shared secrets, our results can be applied to any scenario where two parties need to determine the disjnointness of their sets. Finally, this improvement has been achieved providing the same level of privacy and security of the state of the art solution.

Further ongoing work focus on relaxing the termination criteria—introducing probabilistic termination—and providing probabilistic assurance on the intersection size.

References

1. Agrawal, R., Evfimievski, A., Srikant, R.: Information sharing across private databases. In: Proceedings of the 22th ACM SIGMOD international conference on Management of data (SIGMOD 2003), pp. 86–97 (2003)
2. Barbay, J., López-Ortiz, A., Lu, T., Salinger, A.: An experimental investigation of set intersection algorithms for text searching. Journal of Experimental Algorithmics 14, 3.7–3.24 (2009)
3. Demaine, E.D., López-Ortiz, A., Ian Munro, J.: Adaptive set intersections, unions, and differences. In: Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2000), pp. 743–752 (2000)
4. Eschenauer, L., Gligor, V.: A key-management scheme for distributed sensor networks. In: Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002), pp. 267–282 (2002)
5. 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)
6. Håstad, J., Wigderson, A.: The randomized communication complexity of set disjointness. Journal Theory of Computing 3(1), 211–219 (2007)
7. Kalyanasundaram, B., Schnitger, G.: The probabilistic communication complexity of set intersection. SIAM Journal on Discrete Mathematics 5(4), 545–557 (1992)
8. Kiayias, A., Mitrofanova, A.: Testing disjointness of private datasets. In: S. Patrick, A., Yung, M. (eds.) FC 2005. LNCS, vol. 3570, pp. 109–124. Springer, Heidelberg (2005)
9. Kissner, L., Song, D.X.: Privacy-preserving set operations. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 241–257. Springer, Heidelberg (2005)
10. Kurtz, T.G., Manber, U.: A probabilistic distributed algorithm for set intersection and its analysis. Journal of Theoretical Computer Science 49(2-3), 267–282 (1987)
11. Kushilevitz, E., Nisan, N.: Communication complexity. Cambridge University Press, New York (1997)
12. Mitzenmacher, M., Upfal, E.: Probability and Computing: Randomized Algorithms and Probabilistic Analysis. Cambridge University Press, New York (2005)
13. Motwani, R., Raghavan, P.: Randomized Algorithms. Cambridge University Press, Cambridge (1997)
14. Yao, A.C.-C.: Some complexity questions related to distributive computing. In: Proceedings of the eleventh annual ACM symposium on Theory of computing (STOC 1979), pp. 209–213 (1979)
15. Ye, Q., Wang, H., Pieprzyk, J., Mo Zhang, X.: Unconditionally secure disjointness tests for private datasets. International Journal of Applied Cryptography 1(3), 225–235 (2009)