Broker-Based Private Matching

Abdullatif Shikfa[1], Melek Önen[2], and Refik Molva[2]

[1] Alcatel-Lucent Bell Labs,
Route de Villejust, 91620 Nozay, France,
[2] EURECOM,
2229, route des crêtes, 06560 Sophia Antipolis cedex, France

Abstract. Private matching solutions allow two parties to find common data elements over their own datasets without revealing any additional private information. We propose a new concept involving an intermediate entity in the private matching process: we consider the problem of broker-based private matching where end-entities do not interact with each other but communicate through a third entity, namely the Broker, which only discovers the number of matching elements. Although introducing this third entity enables a complete decoupling between end-entities (which may even not know each other), this advantage comes at the cost of higher exposure in terms of privacy and security. After defining the security requirements dedicated to this new concept, we propose a complete solution which combines searchable encryption techniques together with counting Bloom filters to preserve the privacy of end-entities and provide the proof of the matching correctness, respectively.

1 Introduction

Imagine that a company has an opening for a new position. The posting for new position consists mainly of requirements in terms of education, professional experience and skills. So the company has many selection criteria and is looking for the best suited candidate. Since the company does not want to take care of all the recruitment process itself, it delegates the search phase to a recruitment agency, which is more capable in terms of publishing the posting for new position on a large scale. Candidates are characterized first by their resume and they apply through the recruiting agency if they think they are fit for the job. The recruitment agency upon receiving a resume, looks at the matching ratio between the candidate characteristics and the posting's criteria and calls the best suited candidates for an interview at the company. The best suited candidates are either all candidates above a certain matching ratio threshold, or the top ten candidates for example. In order to prevent resume fraud, candidates should be able to prove the correctness of their resume, with diplomas from a university or validation of experience from a governmental agency.

This interesting scenario raises many security issues. First of all, both company and candidates' privacy should be preserved. The company does indeed not want that competitors learn about the posting, especially if it concerns an

important position because that would give a hint about the company's strategy. So the posting and more specifically the criteria expressed by the company should remain secret from other companies, including the recruitment agency. Candidates' privacy should also be preserved, to enforce equal opportunities among candidates. Therefore resumes should be confidential and anonymous to prevent the recruiting agency from discriminating between candidates on a non-professional basis. The problem is therefore to be able to compute the matching ratio between the posting's criteria and the candidates' resumes while both are encrypted. Furthermore it is important that candidates cannot forge their resume to obtain a higher matching ratio. This problem is especially hard since resumes cannot be checked directly in the case where they are encrypted: privacy and verification present conflicting requirements.

At first glance this problem has a flavor of private matching or private set intersection, whereby two parties want to learn only shared attributes without learning any information about the remaining ones. There is yet an important difference in the presented scenario which makes the problem more complex: the parties owning the private data (the company and the candidates) do not directly interact with each other, but they forward their secret data to a third party. This third party has to take a decision on the matching ratio without having any control or knowledge on the private data it received, and it should not be able to learn anything about the private data of either party in the process: it should just be able to securely compute the matching ratio (it should not even be able to tell which of the encrypted data matched or not). This paper therefore tackles with a new requirement for parties not to interact directly to achieve the matching result thus calling for a non-interactive solution.

In this paper, we analyze the requirements for the non-interactive and private computation of matching ratio and present a complete solution to address this issue. The solution is based on a searchable encryption scheme introduced by Boneh et al. in [3] used in a new mode of operation to allow the company to issue a unique query for all potential (and unknown) candidates. The solution further makes use of counting Bloom filters introduced by Fan et al. in [11], but in a radically new approach: those counting Bloom filters are not used as usual to prove the belonging of an element to a set but to compute the matching ratio without leaking privacy and to provide evidence of the correctness of the matching ratio computation. This solution presents the following advantages:

- it addresses the non-interactive scenario as it does not require the parties owning the private data to interact with each other (such as setting up keys prior to the matching process for example) or even to know each other,
- it allows a third party to compute the matching ratio and to get evidence of its correctness,
- it preserves the privacy of data, because the third party processes encrypted data blindly (in the sense that it handles encrypted data and does not learn any information about it),
- it is efficient, because the third party, which has to process a lot of data from several users, only needs to perform few and non-costly operations for the computation of each matching ratio.

The rest of the paper is structured as follows. Section 2 motivates the need for a broker-based private matching protocol comparing it with the classical two-party mechanisms, defines the security requirements and describes the underlying mechanisms. In section 3, the overall protocol and its security primitives are described in detail. The security and performance of the proposed protocol are evaluated in section 4. Finally, section 5 discusses relevant related work.

2 Problem Statement

2.1 Private matching: introducing a third party

The classical private matching scheme is a two-party protocol that enables both parties P_1 and P_2 to discover common data elements over their own datasets without revealing any additional private information. Assuming that P_1 and P_2 respectively own datasets X_1 and X_2, at the end of the private matching protocol P_1 and P_2 only learn $X_1 \cap X_2$.

In this paper, we propose a complete decoupling between these two parties in order to perform the same operation when the two parties do not interact and are even not aware of each other. The new protocol involves a third party, the Broker, which is in charge of computing the cardinality of the matching set without discovering any of its elements. Private and correct evaluation of the cardinality of the matching set by a third party has many applications, in particular for ranking, or finding friends in social networks or simply in dating sites, and compelling new applications are envisioned in the broad field of cloud computing. All these applications require a third party to take decisions while remaining oblivious to the matched information. This new broker-based private matching protocol consists of three entities, namely the **Query Issuer**, the **Subject** and the **Broker**, where the latter's main role is to discover the cardinality of the matching set originating from the other two entities' datasets. Each entity's role in the new protocol is formally defined as follows:

- the **Query Issuer** $\mathcal{QI}$ issues a query $Q_i = \langle q_{i,1}, ..., q_{i,n} \rangle$ consisting of n selection criteria which are elements of D, the global dataset. In the recruitment example, the company is the Query Issuer and an example of selection criterion could be "Degree = MSc".
- **Subjects** $\mathcal{S}^l$ $(1 \leq l \leq c)$, answers a query Q_i with a matching proof $mp_{i,l}$ based on its profile. Each Subject is indeed characterized by a profile $P^l = \langle p_1^l, ..., p_m^l \rangle$ composed of m attributes which are elements of the same dataset D. These attributes are evaluated with respect to the query defined by the Query Issuer. In the aforementioned scenario, Subjects correspond to the candidates in the recruitment process.
- the additional party, namely the **Broker** $\mathcal{B}$, first publishes the query of $\mathcal{QI}$ to Subjects and collects their answers. The Broker then selects the best suited Subjects: $\mathcal{B}$ computes a matching ratio $\rho_{i,l}$ between a query Q_i and the Subject's answer $mp_{i,l}$ defined as the cardinality of the matching set between the selection criteria and the attributes over the cardinality of the selection criteria. In the example, the Broker is the recruiting agency.

In summary, the major difference between classical private matching and the broker-based private matching protocol is the fact that there is no direct interaction between the Query Issuer $\mathcal{QI}$ and Subjects $\mathcal{S}^l$. All messages go through the Broker $\mathcal{B}$, which is an active entity in the protocol and not a simple relay: the query Q_i of $\mathcal{QI}$ is sent to $\mathcal{B}$ which then publishes it to $\{\mathcal{S}^l\}_{1\leq l\leq c}$; each Subject $\mathcal{S}^l$ sends its answer $mp_{i,l}$ to $\mathcal{B}$ which decides which Subjects correspond to the query the best. Therefore, $\mathcal{QI}$ should be able to send a query without even knowing the Subjects $\{\mathcal{S}^l\}_{1\leq l\leq c}$: there is a complete decoupling between these two entities, and $\mathcal{B}$ is in charge of gathering the necessary data and taking the appropriate decision. Finally $\mathcal{QI}$ should be able to send a query with any selection criteria and is not limited to a set that it owns.

2.2 Security requirements

The introduction of a third party in the private matching protocol requires to revisit all security requirements defined for the two-party protocol.

First of all, we assume that the Query Issuer is interested in getting the best suited Subjects; therefore $\mathcal{QI}$ is assumed to be honest. On the contrary, Subjects are considered to be potentially malicious, because it is in their interest to exhibit a high matching ratio in order to be selected by the Broker. Therefore Subjects might attempt to cheat on their attributes or more generally in the answer they send to $\mathcal{B}$ in order to lure $\mathcal{B}$ into computing a matching ratio higher than their real matching ratio. However we consider that nodes are selfish and that they do not collude with each other.

Concerning the Broker $\mathcal{B}$, we assume it to be honest but curious: $\mathcal{B}$ correctly executes the protocol and computes the matching ratio according to the data it receives, and finally sends to $\mathcal{QI}$ the truly best suited Subjects according to the matching ratio rankings. Yet, $\mathcal{B}$ is curious in the sense that it is interested in unveiling information from the private data it receives, whether being the selection criteria of the query of $\mathcal{QI}$ or the attributes of Subjects.

There are thus two main attacks to be considered:

- attacks by the Broker in an attempt to break the privacy of the other two entities: $\mathcal{B}$ tries to discover and reveal the content of the query of $\mathcal{QI}$, or to discover the attributes of one or many Subjects,
- attacks by Subjects aiming at illegitimately increasing their matching ratio with a given query.

This leads to the following two security requirements:

- **Preserving the privacy of the end entities $\mathcal{QI}$ and $\mathcal{S}^l$**: queries issued by $\mathcal{QI}$ and answers of Subjects are confidential and therefore encrypted. The Broker should be able to compute the matching ratio using these two encrypted values without discovering any information about either the criteria of $\mathcal{QI}$ or the Subject's attributes: the protocol should be semantically secure. Furthermore, as for classical private matching protocol, since the query is forwarded by $\mathcal{B}$ to Subjects, these entities should not be able to derive information about non-matching criteria. These privacy properties can also be formally defined by comparing the real situation in our protocol with an ideal situation where the protocol is run by a trusted external entity, but we do not add this formalization in this article for the sake of clarity.

– **Guaranteeing the correctness of the matching ratio**: the answer $mp_{i,l}$ of a Subject $\mathcal{S}^l$ should enable the Broker to correctly compute the matching ratio between the query Q_i and the attributes of $\mathcal{S}^l$. This requirement is very different from the privacy one, but the latter hardens the task of verifying the correctness of the matching ratio. Indeed, a simple solution to this provably correct matching ratio computation would consist in the Subjects sending their attributes to the Broker, but this solution blatantly exposes the privacy of the Subjects. The challenge for the Broker is to be able to compute the matching ratio corresponding to a set of attributes while verifying their correctness without having access to their content.

2.3 Security primitives

Based on the security requirements of the broker-based private matching protocol, we define the following security primitives:

– `SQE` (Secure Query Encoding): in order to ensure the confidentiality of the query Q_i, this primitive, used by the Query Issuer $\mathcal{QI}$, securely encodes Q_i and returns Q'_i. $\mathcal{QI}$ can express its queries on any selection criteria in the global dataset D, hence `SQE` should be a public function in that it should not require any secret information on input. Furthermore, this function should be randomized to prevent dictionary attacks.
– `SLU` (Secure Look-up): A Subject $\mathcal{S}^l$ uses this primitive to look-up its attributes against an encoded query, and outputs the corresponding answer $mp_{i,l}$. This function should be public but requires secret information (credentials) to be processed.
– `SMRC` (Secure Matching Ratio Computation): on input of a Q'_i and a corresponding $mp_{i,l}$, this primitive first verifies the correctness of $mp_{i,l}$:
  1. if $mp_{i,l}$ is invalid ($\mathcal{S}^l$ attempted to cheat), the process breaks;
  2. otherwise, the primitive outputs the correct matching ratio $\rho_{i,l}$.

In the next section, these three primitives are formally described based on a combination of different cryptographic mechanisms: searchable encryption and counting Bloom filters.

3 Solution

We now present our solution by first introducing the underlying mechanisms and further by formally describing the overall protocol divided into two phases.

3.1 Overview

In order to allow the correct execution of the new protocol, Subjects first need to retrieve their credentials (private information corresponding to their profile) from a certain authority that approves their validity. Therefore, a trusted authority is initially available during a setup phase. This authority does not play any role during the execution of the matching protocol, namely the runtime phase.

In this second phase, the broker-based private matching protocol actually takes place, and it features four main steps:

1. Query: The Query Issuer $\mathcal{QI}$ issues a query Q_i. It encodes this query using the `SQE` primitive and sends the result Q'_i to the Broker. Based on the query Q_i, $\mathcal{QI}$ also constructs a counting Bloom filter CBF_i, called a matching reference and sends it to the Broker along with the encoded query.

2. Publish: The Broker publishes the encoded query $Q'i$ to all Subjects. The matching reference is not forwarded.
3. Look-up: Each subject $\mathcal{S}^l$ looks-up its credentials in the encoded query Q'_i to determine which conditions $\mathcal{S}^l$ matches. Based on these matched conditions, $\mathcal{S}^l$ constructs another counting Bloom filter $CBF_{i,l}$, called a matching proof. This matching proof $mp_{i,l}$ is sent to the Broker.
4. Verify: The Broker compares the matching reference and the matching proof to assess first whether the matching proof is valid or not, and then to compute the matching ratio $\rho_{i,l}$. Finally, the Broker informs $\mathcal{QI}$ about the Subjects best suited to its query Q_i.

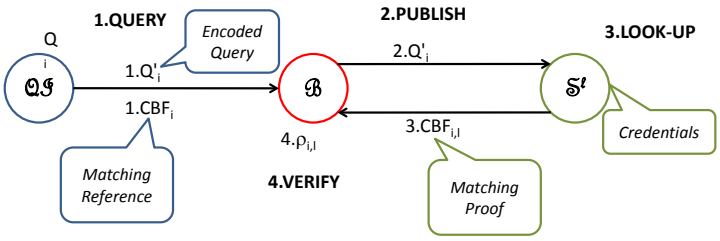

Fig. 1. High level description of the protocol

The protocol is summarized in figure 1. A major advantage of our solution is that it enables some computation on encrypted data to preserve end-entities privacy: the Broker is able to compute the matching ratio based on two encrypted data structures, the matching reference and the matching proof. This computation on encrypted data is achieved thanks to an extension of a searchable encryption mechanism that allows a third node to verify whether an encrypted keyword is included in a database or not. This mechanism is also combined with counting Bloom filters in order to prove the correctness of the computation of the matching ratio. Before formally describing the new protocol, these two mechanisms are briefly presented in the next section.

3.2 Background-Tools

Searchable encryption Searchable encryption is a general concept which enables a third entity to store an encrypted list destined to a certain party and to look-up encrypted keywords on behalf of this party without learning additional information both on the keyword and the encrypted list.

One of the main searchable encryption approaches was proposed by Boneh et al. in [3] and it uses three main operations:

- `SE-Encrypt`: a public encryption function used to encrypt the list that is stored by the third party. This function requires the knowledge of the public key of the destination.
- `SE-Trapdoor`: a private function which gives the capability of looking-up a specific keyword, called a trapdoor. This function requires the private key of the recipient and hence can only be computed by the recipient.
- `SE-Test`: on input of a trapdoor and an encrypted keyword, the third party uses this operation to verify whether the private keyword is included in the

list or not. Hence, this function returns 1 if the trapdoor corresponds to the encrypted keyword and 0 otherwise.

Due to its non-interactivity this searchable encryption proposal looks appropriate for the new broker-based private matching scenario, where the `SE-Test` operation can be implemented by the Broker while Query Issuers may encrypt some keywords with `SE-Encrypt` and the Subjects run the `SE-Trapdoor`. Unfortunately, the use of this mechanism is not straightforward because:

- As opposed to the `SE-Test` operation, the Broker should only be able to compute the global matching ratio and not individual matching attributes;
- The Query Issuer does not know the Subjects in advance, hence it does not have knowledge of their public keys and cannot use `SE-Encrypt` easily.

To circumvent these two main constraints, we propose to introduce a Trusted Third Party which alleviates the requirement of the knowledge of the (unknown) recipient's public key in our scheme (see section 3.3).

Bloom filters A Bloom filter is a probabilistic data structure which was first introduced by Burton Bloom ([5]). The classical use of Bloom filters is to test whether an element is a member of a set in a space-efficient way. We focus on an extension of Bloom filters called counting Bloom filters that were proposed by Fan et al. in [11] to support the dynamic deletion of an element.

A counting Bloom filter CBF is an array of ϕ positions (also called buckets) used to represent a set $\mathcal{X}$ with the aid of u hash functions $\{h_1, .., h_u\}$ mapping an element of $\mathcal{X}$ to one of the ϕ array positions. Counting Bloom filters implement the following three functions:

- `Insert`(x, CBF): on input of an element x, the digest of this element is computed using each of the u hash functions. The values of the filter CBF at these positions are incremented by 1.
- `Query`(x, CBF): this function verifies with a certain probability whether x is an element of the filter or not.
- `Delete`(x, CBF): this operation consists of decrementing the value at each of the u positions resulting from the hash functions evaluated over x, by 1.

In the sequel of this article, we denote by `CBF`$(x_1, ..., x_n)$ the counting Bloom filter obtained by inserting the elements x_i for $1 \leq i \leq n$.

The weight w_{CBF} of a counting Bloom filter CBF is defined as the sum of the values of all positions: $w_{CBF} = \sum_{0 \leq i \leq \phi - 1} CBF[i]$. An important property of counting Bloom filters is that the weight w_{CBF} of a counting Bloom filter CBF is linearly dependent on the number of elements inserted in it:

$$w_{\texttt{CBF}(x_1,...,x_n)} = n \cdot u.$$

Hence, counting Bloom filters are useful for our broker-based private matching as they enable computing the cardinality of a set without revealing the elements of the set (see section 3.3).

3.3 Construction

As mentioned in section 3.1, the solution features two phases: a setup phase where Subjects retrieve their credentials, and a runtime phase where the private matching protocol is executed.

Setup phase Contrary to $\mathcal{QI}$ which can choose any selection criteria in Q_i, $\mathcal{S}^l$ should answer Q'_i correctly based on their profile. Since the correctness of private matching operations depends on the correctness of these profiles, the latter should be certified, and we refer to the certified attributes as credentials. These credentials are retrieved during a setup phase which features a fourth entity, called the **Authority** $\mathcal{A}$. This Authority is required to define general parameters of the system and to provide Subjects with their matching credentials.

The general parameters are generated according to a security parameter which impacts the size of the groups that are used, as well as the size of keys. In particular, the Authority generates a private and public key pair $sk_{\mathcal{A}}/pk_{\mathcal{A}}$. In the recruitment example, universities delivering a diploma or governmental agencies can be considered as authorities.

In addition to the three security primitives defined in section 2.3, we define a fourth one, `SCE` (Secure Credential Extraction), which is used by $\mathcal{A}$ to provide $\mathcal{S}^l$ with the credentials corresponding to its profile (this primitive is similar to the private key extraction primitive in Identity-Based Encryption). On input of a Subject's profile, `SCE` returns a set of credentials. These credentials are used as matching capabilities and correspond to trapdoors in searchable encryption.

To be more precise, Subjects $\mathcal{S}^l$ first contact the Authority $\mathcal{A}$ and show their profile $P^l = \langle p^l_1, ...p^l_m \rangle$. $\mathcal{A}$ verifies the validity of P^l (this verification step is out of the scope of this paper), and then provides $\mathcal{S}^l$ with the corresponding credentials T^l which are computed using the `SE-Trapdoor` function applied over the Subject's attributes and the secret key of $\mathcal{A}$. Hence, at the end of the setup phase, each S^l receives the following set of credentials:

$$T^l = \langle t^l_1, ..., t^l_m \rangle = \langle \texttt{SE-Trapdoor}(p^l_1, sk_{\mathcal{A}}), ..., \texttt{SE-Trapdoor}(p^l_m, sk_{\mathcal{A}}) \rangle.$$

Runtime phase As described in section 3.1, the runtime phase consists of four main steps that we describe formally in the following:

1. **Query:** During this step, $\mathcal{QI}$ expresses a query Q_i by choosing a set of selection criteria and performs a secure encoding of the query using the `SQE` primitive. The output of this primitive are the encoded query Q'_i and the matching reference mr_i: $\texttt{SQE}(Q_i, pk_{\mathcal{A}}) = (Q'_i, mr_i)$.
   As previously introduced, the `SQE` primitive should be a randomized public cryptographic function, such as `SE-Encrypt`. However, `SE-Encrypt` requires the public key of the recipient and this key is unknown to $\mathcal{QI}$, hence we propose a new configuration where the public key of $\mathcal{A}$ is used instead. Therefore, the encoded query is computed as follows:

   $$Q'_i = \langle q'_{i,1}, ..., q'_{i,n} \rangle = \langle \texttt{SE-Encrypt}(q_{i,1}, pk_{\mathcal{A}}), ..., \texttt{SE-Encrypt}(q_{i,n}, pk_{\mathcal{A}}) \rangle.$$

   On the other hand, the matching reference should help the Broker to compute the matching ratio correctly. To this extent, during the execution of the `SE-Encrypt` algorithm, $\mathcal{QI}$ also retrieves some intermediate values which can only be computed by itself or by the nodes that own the corresponding trapdoors. Indeed, the `SE-Encrypt` primitive makes use of a cryptographic

hash function H at the last step of the computation[3]. For $1 \leq j \leq n$, we denote the preimage of $q'_{i,j}$ by $x_{i,j}$:

$$q'_{i,j} = \texttt{SE-Encrypt}(q_{i,j}, pk_{\mathcal{A}}) = H(x_{i,j}).$$

Thanks to the inherent security of the hash functions with pseudorandom inputs, a malicious user cannot compute $x_{i,j}$ based on the knowledge of $q'_{i,j}$. Hence, $\mathcal{QI}$ constructs the matching reference mr_i as a counting Bloom filter CBF_i, in which it inserts the elements $x_{i,j}$ for $1 \leq j \leq n$:

$$mr_i = CBF_i = \texttt{CBF}(x_{i,1}, ..., x_{i,n}).$$

At the end of this first step, $\mathcal{QI}$ sends mr_i and Q'_i to the Broker.

2. **Publish:** The Broker forwards the encoded query Q'_i to all Subjects but keeps the matching reference mr_i.
3. **Look-up:** On input of an encoded query Q'_i and a set of credentials T^l, the SLU primitive returns a matching proof $mp_{i,l}$:

$$\texttt{SLU}(Q'_i, T^l) = mp_{i,l}.$$

By using the SE-Test function, Subjects can indeed detect selection criteria corresponding to their profile: for $1 \leq j \leq n, 1 \leq k \leq m$ $\texttt{SE-Test}(q'_{i,j}, t^l_k)$ returns 1 for matching elements and 0 for the others. Moreover, the Subject can compute the corresponding preimage $x_{i,j}$ for matching criteria. Hence $\mathcal{S}^l$ can construct a counting Bloom filter $CBF_{i,l}$ in which it includes all the preimages that it managed to compute and which is used as matching proof $mp_{i,l} = CBF_{i,l}$ and sent to the Broker.

4. **Verify:** On input of a matching reference mr_i and a matching proof $mp_{i,l}$ the primitive SMRC returns a matching ratio $\rho_{i,l}$.

The Broker first compares the counting Bloom filters CBF_i and $CBF_{i,l}$ to assess the validity of the latter. To this extent, the Broker checks whether:

- $\forall 0 \leq i_1 \leq \phi - 1, CBF_{i,l}[i_1] \prec CBF_i[i_1]$ denoted as $CBF_{i,l} \prec CBF_i$, otherwise it means that $CBF_{i,l}$ was not constructed only with (a subset of) $x_{i,1}, ...x_{i,n}$,
- the weight $w_{CBF_{i,l}}$ of $CBF_{i,l}$ is a multiple of u, because each inserted element leads to an increase of the weight by u.

If one of the verifications fails, the protocol aborts (the Subject attempted to cheat), otherwise the Broker accepts the answer of $\mathcal{S}^l$ as being valid and computes the matching ratio as follows:

$$\texttt{SMRC}(mr_i, mp_{i,l}) = \frac{w_{CBF_{i,l}}}{w_{CBF_i}}.$$

The protocol is consistent in that:

Proposition 1. *If $CBF_{i,l}$ is generated as specified in the protocol, then the matching ratio between the query and the attributes of a Subject corresponds to the output of SMRC:*

$$\rho_{i,l} = \textit{SMRC}(mr_i, mp_{i,l}).$$

[3] See [3] for the detailed construction of PEKS. We roughly have $x_{i,j} = \hat{e}(H_1(q_{i,j}), r \cdot pk_{\mathcal{A}})$, and $q'_{i,j} = \langle rP, H(x_{i,j})\rangle$, where $\hat{e}$ is a bilinear map, r a random scalar, and P a point on an elliptic curve.

This proposition is a direct consequence of the fact that the weight of a counting Bloom filter is linearly dependent with the number of its elements.

This concludes the presentation of our solution, and we now evaluate its security and performance.

4 Evaluation

The security of the new broker-based private matching protocol is analyzed based on the attacker model and the security requirements defined in section 2.2. We assume that the communication channels between $\mathcal{QI}$ and $\mathcal{B}$ and between $\mathcal{B}$ and $\mathcal{S}^l$ are secured, hence eavesdroppers cannot access the messages exchanged in the protocol in clear. They thus have less information than any of the entities running the protocol, and we do not further take them into account.

4.1 Privacy

Privacy is the most important requirement in classical private matching. In this section, we assume that entities are curious and try to discover information that they should not access. We first show that our solution preserves the privacy of end-entities and we further prove that the introduction of a third party (the Broker) does not threaten the Query Issuer's and Subjects' privacy.

First, the privacy of the $\mathcal{QI}$ is preserved with respect to $\mathcal{S}^l$. Indeed, in [3], Boneh et al. proved that their construction is semantically secure against a chosen keyword attack in the random oracle model, assuming that the Bilinear Diffie-Hellman problem is hard. It is thus unfeasible for an entity to discover the value of an encoded selection criteria unless it knows the corresponding trapdoor, in other words $\mathcal{S}^l$ can only discover the matching selection criteria. Furthermore, since only the Authority $\mathcal{A}$ knows the private key $sk_{\mathcal{A}}$, nodes cannot forge trapdoors. Recovering the private key $sk_{\mathcal{A}}$ amounts to a discrete logarithm computation which is assumed to be hard.

Second, we prove that the introduction of $\mathcal{B}$ does not threaten the privacy of end-entities. On one hand, as an intermediate node, $\mathcal{B}$ receives the same encoded queries that $\mathcal{S}^l$ receives, but $\mathcal{B}$ has no trapdoors and thus cannot discover the value of the encoded queries. Furthermore, $\mathcal{B}$ cannot link successive queries even if they correspond to the same selection criteria because the encoding mechanism is inherently randomized. On the other hand, in addition to the queries, $\mathcal{B}$ receives matching reference and matching proofs from $\mathcal{QI}$ and $\mathcal{S}^l$ respectively. As proven in the following theorem, the knowledge of a counting Bloom filter does not enable the Broker to recover the elements $x_{i,j}$ inserted in it.

Theorem 1. *Let $x_1, ..., x_n$ be n elements randomly chosen from a group G of order q. Let CBF be a counting Bloom filter of size ϕ in which the n elements $x_1, ..., x_n$ were inserted using u hash functions $h_1, ..., h_u$. Then, there are more than $\frac{q}{\phi^u}$ possible sets of elements of G^n leading to the same counting Bloom filter:*

$$|\{(x'_1, ..., x'_n) \in G^n | \texttt{CBF}(x'_1, ..., x'_n) = \texttt{CBF}(x_1, ..., x_n)\}| > \frac{q}{\phi^u}$$

The proof is given in section 7.1. This result is a lower bound on the set of preimages but the actual result can be multiplied by a factor of up to $u!$ depending on the outputs of the hash functions, and is multiplied even further if more elements are inserted. Note that this result does not even take into account the complexity required to find the corresponding set of preimages.

From the perspective of an attacker, being able to solve the equations would lead to an advantage as it reduces the size of the space of possibilities from q down to $\frac{q}{\phi^u}$. However, careful setting of the parameters q, ϕ and u, makes the size of this set large enough to prevent brute force guessing (see section 7.3).

In summary, the counting Bloom filter cannot be reversed to obtain the entries that were inserted in it, which guarantees the privacy of the Query Issuer and Subjects. We now focus on the security of the matching ratio computation.

4.2 Correctness of the matching ratio

Concerning the security of the matching ratio computation, we consider now a malicious $\mathcal{S}^l$ trying to convince $\mathcal{B}$ that its matching ratio is higher than its actual value, and we show that the probability of success of such an attack is negligible.

To be more precise, we assume that $\mathcal{S}^l$ does not know the matching reference mr_i, thus the only information known by $\mathcal{S}^l$ on CBF_i are the global parameters: the hash functions used h_1,...,h_u and the size ϕ. $\mathcal{S}^l$ also knows Q'_i and therefore the number n of elements $x_{i,j}$ inserted in CBF_i.

The goal of the malicious $\mathcal{S}^l$ is to claim a matching ratio $\rho_{i,l}^{claim}$ higher than the actual ratio $\rho_{i,l}$. To this extent, $\mathcal{S}^l$ needs to claim a corresponding counting Bloom filter $CBF_{i,l}^{claim}$. For $\mathcal{S}^l$ to be successful, $CBF_{i,l}^{claim}$ has to verify the following conditions:

1. it should be considered valid by $\mathcal{B}$, as required by the last step of the protocol described in section 3.3, which implies that:
   - $CBF_{i,l}^{claim} \prec CBF_i$,
   - the weight $w_{CBF_{i,l}^{claim}}$ of $CBF_{i,l}^{claim}$ is a multiple of u,
2. it should lead to $\rho_{i,l}^{claim} > \rho_{i,l}$, hence the weight of $CBF_{i,l}^{claim}$ needs to verify $w_{CBF_{i,l}^{claim}} > w_{CBF_i}$.

The probability of success of $\mathcal{S}^l$ is exponentially decreasing in the malicious ratio increment $\rho_{i,l}^{claim} - \rho_{i,l}$, as shown in the following theorem.

Theorem 2. *Let Q'_i be an encoded query concerning n selection criteria. Let CBF_i be the corresponding matching reference.*

The probability of success $\mathcal{P}_{adv}[\rho_{i,l} \to \rho_{i,l}^{claim}]$ of an adversary $\mathcal{S}^l$ in generating an array $CBF_{i,l}^{claim}$ which is accepted by $\mathcal{B}$ and results in an increase of the matching ratio from $\rho_{i,l}$ to $\rho_{i,l}^{claim}$ is upper bounded by:

$$\mathcal{P}_{adv}[\rho_{i,l} \to \rho_{i,l}^{claim}] \leq \left(1 - e^{-\frac{(1-\rho_{i,l})n \cdot u}{\phi}}\right)^{(\rho_{i,l}^{claim} - \rho_{i,l})n \cdot u}$$

The proof is given in section 7.2. The formula of $\mathcal{P}_{adv}[\rho_{i,l} \rightarrow \rho_{i,l}^{claim}]$ shows that the probability of success of an adversary decreases exponentially with the malicious ratio increase $(\rho_{i,l}^{claim} - \rho_{i,l})$ and, decreases also depending on the value of the legitimate matching ratio $\rho_{i,l}$.

It is possible to go further and bound $\mathcal{P}_{adv}[\rho_{i,l} \rightarrow \rho_{i,l}^{claim}]$ independently of $\rho_{i,l}$ and $\rho_{i,l}^{claim}$, by observing that:

- the function $x \mapsto \alpha^x$ decreases with x for $0 < \alpha < 1$,
- $0 < \left(1 - e^{-\frac{(1-\rho_{i,l})n \cdot u}{\phi}}\right) < \left(1 - e^{-\frac{n \cdot u}{\phi}}\right) < 1$,
- $u < (\rho_{i,l}^{claim} - \rho_{i,l})n \cdot u$.

Hence, the probability of success of the adversary is bounded by $\mathcal{P}_{adv}$:

$$\mathcal{P}_{adv} = \left(1 - e^{-\frac{n \cdot u}{\phi}}\right)^u.$$

The security of the scheme hence depends on the general parameters of the counting Bloom filter and we now show how to optimize these settings.

First of all, we assume that the maximum number of selection criteria in a query is bounded and known in advance; we designate it as n_{max}. For all queries, the probability of success of the adversary is thus bounded by

$$\mathcal{P}_{adv} \leq \left(1 - e^{-\frac{n_{max} u}{\phi}}\right)^u.$$

If we fix ϕ, then the function $p_{max} : u \mapsto \left(1 - e^{-\frac{n_{max} u}{\phi}}\right)^u$ is $\mathcal{C}^\infty$ on $[1, +\infty[$, and it reaches its minimum in $u_0 = \frac{\phi}{n_{max}} ln(2)$ and $p_{max}(u_0) = 2^{-u_0}$. Therefore, for a fixed n_{max}, increasing u and ϕ exponentially increases the security, but increasing ϕ linearly impacts on the performance of the scheme. We propose the following strategy to optimize the trade-off between security and performance:

1. Set n_{max} the maximum number of criteria per query,
2. Choose a security parameter u: $\mathcal{P}_{adv}$ is then bounded by 2^{-u},
3. Set the size ϕ of the counting Bloom filter as $\phi = \left\lceil \frac{n_{max} u}{\ln(2)} \right\rceil$.

This strategy prioritizes security over performance: it defines the desired security level ($\mathcal{P}_{adv} \leq 2^{-u}$) and then sets the minimal size ϕ to achieve this security level. Note that u does not need to be very large, because $\mathcal{P}_{adv}$ is an upper bound and is obtained with very restrictive conditions:

- $n = n_{max}$, which means that $\mathcal{QI}$ uses n_{max} selection criteria,
- $\mathcal{S}^l$ has a legitimate matching ratio of 0 ($\rho_{i,l} = 0$).

With these conditions, $\mathcal{S}^l$ has a probability less than 2^{-u} of success in making $\mathcal{B}$ believe that its matching ratio is $1/n_{max}$ instead of 0. In many cases, this would not be of any use to the attacker, because the attacker needs to claim the highest matching ratio among the Subjects in order to take advantage of its attack. The attacker does not even know the matching ratio of the other Subjects, so the only way for the malicious $\mathcal{S}^l$ to be sure to benefit from its attack is to claim a matching ratio of 1, and the probability of $\mathcal{S}^l$ succeeding falls down to $2^{-u \cdot n_{max}}$.

4.3 Performance evaluation

Following the analysis of the trade-off between security and performance in the previous section, we now evaluate the overall communication and computational overhead resulting from the proposed protocol.

Communication overhead We consider that the cost originating from the setup phase is negligible given that it takes place offline. We only evaluate the communication overhead during the runtime phase.

The size of encoded queries is linear in the number of selection criteria that it includes. Each encoded criterion is the output of the `SE-Encrypt` primitive and thus has size $2q$ bits, where q is the size of the group used in the searchable encryption scheme.

Concerning the size of counting Bloom filters, they are arrays containing ϕ buckets. According to [11], we choose $\beta = 4$ bits for the size of each bucket to keep a negligible probability of overflow, thus the communication overhead incurred by the matching reference or the matching proof is 4ϕ bits.

Computational overhead The primitives of searchable encryption rely on elliptic curve operations which cost is of the same order of magnitude as classical asymmetric cryptography [18]. The most costly operation is the pairing computation: our mechanism requires one pairing computation per encoding and one per `SE-Test` evaluation, the cost is thus linear in the number of selection criteria used in the queries. In comparison, the cost of generating the counting Bloom filters which amounts to $n \cdot u$ hash computations is negligible.

The aforementioned computations are performed by the end-entities, but the Broker only carries on simple operations to compute the matching ratio:

- $\mathcal{B}$ verifies that the matching proof is smaller than the matching reference which requires ϕ integers inequality checks,
- $\mathcal{B}$ computes the weight of the matching proof and reference (a sum of ϕ integers) and performs a division.

The overhead on $\mathcal{B}$ is thus very small which shows that our scheme is scalable and efficient to disseminate a query to multiple Subjects.

5 Related work

Several previously studied problems in the literature show similarities with broker-based private matching. We list them in two main categories and show how they differ from our problem.

5.1 Private matching and private set intersection

Private matching came up as a generalization of private equality tests. A first approach introduced a Trusted Third Party (TTP) as proposed in [2] and [15]. In theses proposals, the TTP is completely trusted, computes $X_1 \cap X_2$ and sends the result back to P_1 and P_2. This solution is not satisfying from a privacy perspective as it is fully dependent on the honesty of the TTP which has full

access to the parties' sets. This three-party protocol is thus very different from our broker-based private matching solution.

In [1], Agrawal et al. propose a protocol performing private matching without a TTP, building on a previous work by Huberman et al. [14] by using a pair of commutative encryption schemes. Building on this work, Li et al. formalize in [17] the security requirements of private matching and identify the issue of spoofing, which consists in one of the entities claiming elements that it does not own. The issue of spoofing is similar to Subjects cheating in their matching proof (however this issue is not relevant for the Query Issuer). To solve this issue, Li et al. further introduce a Trusted Third Party which provides Data Ownership Certificates (similar to the Authority providing credentials) and propose a modified version of the Agrawal protocol.

A different approach was investigated by Freedman et al. in [12]: they propose a solution derived from secret sharing protocols based on Oblivious Polynomial Evaluation. They also study some variants of private matching, among which the private cardinality matching, which is very close to our matching ratio computation. The solution for the latter is only proposed for semi-honest parties but the case of malicious entities is not considered. Kissner and Song [16] proposed multi-party protocols that apply to several set operations (including set intersection) and that are secure in the presence of honest-but-curious adversaries. They also propose a construction secure in the presence of malicious adversaries based on zero-knowledge proofs. For the same problem, Dachman-Sold et al. propose a more efficient solution in [10].

In [8], Camenisch and Zaverucha introduce the notion of certified sets: a trusted third party provides credentials to users prior to the private set intersection protocol. This trusted third party plays the same role as $\mathcal{A}$ in our solution.

Finally, we note the recent work of De Cristofario and Tsudik, who propose in [9] more efficient protocols to various flavors of private set intersection.

All these protocols cannot readily be applied to our scenario, because they are interactive protocols between two entities (a client and a server) that interact directly (possibly in several rounds), and there is no clear translation of this two-party setting to our problem. The presence of an active Broker indeed introduces different privacy threats while enabling a decoupling between Query Issuer and Subjects. Furthermore, one of the entities in our scenario, namely the Query Issuer, can express queries on any selection criteria and is not limited to a predefined set contrary to P_1 limited to X_1 in classical private matching.

5.2 Oblivious keyword search

Oblivious Keyword Search is a generalization of Oblivious Transfer [21, 4, 7, 13] where the client receives all messages related to a given private keyword instead of requesting a message at a particular position. It was proposed by Ogata and Kurosawa in [20] who showed the relationships between both notions and presented two efficient methods to achieve oblivious keyword search.

Oblivious Keyword Search is relevant to our problem because it can be used to construct private set intersection protocols [12], and more importantly they

can be combined with Public Encryption with Keyword Search (PEKS) to offer additional properties as presented in [6]. The latter scheme, that we refer to as PEOKS, enhances PEKS by introducing the notion of committed blind anonymous identity-based encryption, which allow Subjects $\mathcal{S}^l$ to privately request trapdoors for attributes without revealing the attributes to the Authority $\mathcal{A}$: $\mathcal{S}^l$ commit to their attributes which allows $\mathcal{A}$ to request proofs of statement from users later on. Furthermore, the trapdoors are unique to each subject (even for the same attribute), making the scheme robust and **secure against colluding attackers**. Those properties make PEOKS more suitable to our scenario than PEKS but it is also more difficult to expose briefly and could stray the focus from our contributions and in particular the main novelty of our scheme, which is the introduction of counting Bloom filters and their use in an original way. We keep the advanced version of our scheme based on PEOKS for the extended version of the article.

6 Conclusion

In this paper, we have presented a new private matching protocol which involves an intermediate node that performs some of the matching operations on behalf of the end-entities. Contrary to classical private matching settings, where the client and the server interact directly in the process, in our new scenario the Query Issuer and the Subjects do not interact at all, and do not even need to know each others' identity. The new protocol is based on the combination of searchable encryption mechanisms and counting Bloom filters used in a radically different mindset and allows a third entity, namely the Broker, to correctly compute the matching ratio based on encrypted information only. While introducing this third entity allows a decoupling between the end-entities, it raises new privacy and security issues. We have proved that the proposed protocol preserves the privacy of end-entities thanks to the semantic security of the underlying searchable encryption mechanisms. The security against malicious Subjects cheating on the matching ratio has been analyzed and proved by bounding the probability of the success of the malicious Subject. Finally we have identified an interesting trade-off between security and performance, and we have computed the optimal parameters for an efficient execution of the protocol under a certain security level.

As future work, we plan to implement this mechanism with a PEOKS scheme to mitigate the impact of colluding attackers. We also envision to introduce multiple authorities to reduce the importance and the capabilities of $\mathcal{A}$.

References

1. Agrawal, R., Evfimievski, A., and Srikant, R. Information sharing across private databases. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data* (2003), ACM, pp. 86–97.
2. Ajmani, S., Morris, R., and Liskov, B. A trusted third-party computation service. Tech. Rep. MIT-LCS-TR-847, MIT, may 2001. www.pmg.csail.mit.edu/ ajmani/papers/tep.pdf.

3. Boneh, D., Crescenzo, G. D., Ostrovsky, R., and Persiano, G. Public Key Encryption with keyword Search. In *Advances in Cryptology - EUROCRYPT 2004* (2004), Springer Berlin/Heidelberg, pp. 506–522.
4. Brassard, G., Crépeau, C., and Robert, J.-M. All-or-nothing disclosure of secrets. In *Proceedings on Advances in cryptology—CRYPTO '86* (1986), Springer-Verlag, pp. 234–238.
5. Broder, A., and Mitzenmacher, M. Network applications of bloom filters: A survey. In *Internet Mathematics* (2002), pp. 636–646.
6. Camenisch, J., Kohlweiss, M., Rial, A., and Sheedy, C. Blind and anonymous identity-based encryption and authorised private searches on public key encrypted data. In *Proceedings of the 12th International Conference on Practice and Theory in Public Key Cryptography: PKC '09* (2009), Springer-Verlag, pp. 196–214.
7. Camenisch, J., Neven, G., and Shelat, A. Simulatable adaptive oblivious transfer. In *Proceedings of the 26th annual international conference on Advances in Cryptology* (2007), EUROCRYPT '07, Springer-Verlag, pp. 573–590.
8. Camenisch, J., and Zaverucha, G. M. Private intersection of certified sets. In *Financial Cryptography and Data Security* (2009), Springer-Verlag, pp. 108–127.
9. Cristofaro, E. D., and Tsudik, G. Practical private set intersection protocols with linear complexity. In *Financial Cryptography'10* (2010), pp. 143–159.
10. Dachman-Soled, D., Malkin, T., Raykova, M., and Yung, M. Efficient robust private set intersection. In *Proceedings of the 7th International Conference on Applied Cryptography and Network Security* (2009), ACNS '09, Springer-Verlag, pp. 125–142.
11. Fan, L., Cao, P., Almeida, J., and Broder, A. Z. Summary cache: a scalable wide-area web cache sharing protocol. *IEEE/ACM Transactions on Networking 8*, 3 (2000), 281–293.
12. Freedman, M. J., Nissim, K., and Pinkas, B. Efficient private matching and set intersection. In *Advances in Cryptology - EUROCRYPT 2004* (2004), Springer Verlag.
13. Green, M., and Hohenberger, S. Blind identity-based encryption and simulatable oblivious transfer. In *Proceedings of the Advances in Crypotology 13th international conference on Theory and application of cryptology and information security* (2007), ASIACRYPT'07, Springer-Verlag, pp. 265–282.
14. Huberman, B. A., Franklin, M., and Hogg, T. Enhancing privacy and trust in electronic communities. In *EC '99: Proceedings of the 1st ACM conference on Electronic commerce* (1999), ACM, pp. 78–86.
15. Jefferies, N., Mitchell, C. J., and Walker, M. A proposed architecture for trusted third party services. In *Proceedings of the International Conference on Cryptography: Policy and Algorithms* (1995), Springer-Verlag, pp. 98–104.
16. Kissner, L., and Song, D. Privacy-preserving set operations. In *Proceedings of CRYPTO '05* (August 2005).
17. Li, Y., Tygar, D., and Hellerstein, J. M. Private matching. Tech. Rep. IRB-TR-04-005, Intel Research Laboratory Berkeley, 2004. http://www.eecs.berkeley.edu/~tygar/papers/Private_matching.pdf.
18. Lynn, B. The pairing-based cryptography library, 2006. http://crypto.stanford.edu/pbc/.
19. Menezes, A., Vanstone, S., and Okamoto, T. Reducing elliptic curve logarithms to logarithms in a finite field. In *STOC '91: Proceedings of the twenty-third annual ACM symposium on Theory of computing* (1991), pp. 80–89.
20. Ogata, W., and Kurosawa, K. Oblivious keyword search. *Journal of Complexity - Special issue on coding and cryptography 20* (April 2004), 356–371.

21. Rabin, M. O. How to exchange secrets with oblivious transfer, 1981. Harvard University Technical Report.

7 Appendix: proofs and example

7.1 Proof of theorem 1

Theorem. *Let $x_1, ..., x_n$ be n elements randomly chosen from a group G of order q. Let CBF be a counting Bloom filter of size ϕ in which the n elements $x_1, ..., x_n$ were inserted using u hash functions $h_1, ..., h_u$. Then, there are more than $\frac{q}{\phi^u}$ possible sets of elements of G^n leading to the same counting Bloom filter:*

$$|\{(x'_1, ..., x'_n) \in G^n | \mathtt{CBF}(x'_1, ..., x'_n) = \mathtt{CBF}(x_1, ..., x_n)\}| > \frac{q}{\phi^u}$$

Proof. Let us examine the simplest case of $n = 1$ and $CBF = \mathtt{CBF}(x_1)$. In that case the positions $h_1(x_1); ...; h_u(x_1)$ are incremented in CBF. The security argument is based on two main observations:

- The first observation is that the hash functions $h_1, ..., h_u$ are not invertible, even though they are not necessarily cryptographic hash functions. Indeed, these functions map elements of G (a group of order q) to a small set (the integers smaller than ϕ). Therefore, if the hash functions have a uniformly distributed output then each output has $\frac{q}{\phi}$ preimages. If we combine the u equations corresponding to the u hash functions, the number of inputs simultaneously verifying u conditions on their digests is $\frac{q}{\phi^u}$.
- The second observation is that there is an information loss in the construction of this structure: the order of the hash functions is lost once the element is inserted in the counting Bloom filter, and it is impossible to know which hash function resulted in the incrementation of a given position in the filter. This second fact further increases the size of the potential preimages by a factor of up to $u!$: it is possible to set many sets of equations for the same counting Bloom filter.

As a result, the set of possible preimages corresponding to a counting Bloom filter containing a single element is at least $\frac{q}{\phi^u}$. This set is even larger when considering several elements. □

7.2 Proof of theorem 2

Theorem. *Let Q'_i be an encoded query concerning n selection criteria. Let CBF_i be the corresponding matching reference.*

The probability of success $\mathcal{P}_{adv}[\rho_{i,l} \to \rho_{i,l}^{claim}]$ of an adversary $\mathcal{S}^l$ in generating an array $CBF_{i,l}^{claim}$ which is accepted by $\mathcal{B}$ and results in an increase of the matching ratio from $\rho_{i,l}$ to $\rho_{i,l}^{claim}$ is upperly bounded by:

$$\mathcal{P}_{adv}[\rho_{i,l} \to \rho_{i,l}^{claim}] \leq \left(1 - e^{-\frac{(1-\rho_{i,l})n \cdot u}{\phi}}\right)^{(\rho_{i,l}^{claim} - \rho_{i,l})n \cdot u}$$

Proof. We first observe that $\mathcal{S}^l$ cannot know whether the first property (that is $CBF_{i,l}^{claim} \prec CBF_i$) is met or not as $\mathcal{S}^l$ does not know CBF_i. $\mathcal{S}^l$ can only make guesses based on the general parameters of CBF_i. We thus first establish a probabilistic model of counting Bloom filters in order to evaluate the probability of having the three aforementioned properties validated without the knowledge of CBF_i.

We consider a counting Bloom filter CBF of length ϕ containing n unknown elements which were inserted using u hash functions. Given that the probability distribution of the values in CBF_i follows a binomial distribution at each position, the probability $\mathcal{P}'(i_2)$ that the value $CBF[i_1]$ at position i_1 is greater than a given i_2 can be computed as follows: $\forall 0 \leq i_1 \leq \phi - 1, \forall 1 \leq i_2 \leq n \cdot u$,

$$\mathcal{P}'(i_2) = \mathcal{P}[CBF[i_1] \geq i_2] = 1 - \sum_{i_3=0}^{i_2-1} \binom{n \cdot u}{i_3} \left(1 - \frac{1}{\phi}\right)^{n \cdot u - i_3} \left(\frac{1}{\phi}\right)^{i_3}.$$

Based on this result, we then prove by induction[4] that the probability $\mathcal{P}'(i_2)$ decreases faster than a geometric series of ratio $\mathcal{P}'(1)$, or to be more precise that, for $1 \leq i_2 \leq n \cdot u$,

$$\mathcal{P}'(i_2) \leq (\mathcal{P}'(1))^{i_2} \tag{1}$$

assuming that $n \cdot u \leq \phi - 1$.

We then consider ARR to be an array of size ϕ (the matching proof). The probability $\mathcal{P}[ARR \prec CBF]$ that ARR is smaller than CBF can be computed as follows:

$$\mathcal{P}[ARR \prec CBF] = \prod_{i_1=0}^{\phi-1} \mathcal{P}'(ARR[i_1]).$$

Following the result in inequation 1, this probability can be upperly bounded as follows:

$$\mathcal{P}[ARR \prec CBF] \leq \prod_{i_1=0}^{\phi-1} \mathcal{P}'(1)^{ARR[i_1]}$$

Finally, based on the approximation of the Taylor series development of $\mathcal{P}'(1)$ we obtain the following upper bound:

$$\mathcal{P}[ARR \prec CBF] \leq \left(1 - e^{-\frac{n \cdot u}{\phi}}\right)^{w_{ARR}} \tag{2}$$

The last step of the demonstration consists in applying this important result to the matching reference CBF_i and the matching proof $CBF_{i,l}$ where the parameters CBF and ARR are replaced by the challenging reference counting Bloom filter CBF_i and the malicious matching proof $CBF_{i,l}$, respectively. However, this modification is not straightforward because while CBF was assumed to contain n random elements, a malicious Subject $\mathcal{S}^l$ knows some of the elements, that are the ones corresponding to the selection criteria that $\mathcal{S}^l$ matches.

[4] The (long) details of this proof are not included due to page constraints

Thus, the following modifications have to be performed to evaluate the probability $\mathcal{P}_{adv}[\rho_{i,l} \rightarrow \rho_{i,l}^{claim}]$ of success of a Subject in increasing its matching ratio from $\rho_{i,l}$ to $\rho_{i,l}^{claim}$:

- We first define by $CBF_i^{chal} = CBF_i - CBF_{i,l}$ the challenging reference counting Bloom filter, that is the part of the counting Bloom filter unknown to $\mathcal{S}^l$. The weight of CBF_i^{chal} is $w_{CBF_i^{chal}} = n(1 - \rho_{i,l}) \cdot u$
- Moreover, $CBF_{i,l}^{mal} = CBF_{i,l}^{claim} - CBF_{i,l}$ defines the part of the matching proof which is malicious which weight is denoted by $w_{CBF_{i,l}^{mal}}$ which is computed as follows: $w_{CBF_{i,l}^{mal}} = w_{CBF_{i,l}^{claim}} - w_{CBF_{i,l}} = (\rho_{i,l}^{claim} - \rho_{i,l})n \cdot u$

We therefore obtain the following inequality:

$$\mathcal{P}[CBF_{i,l}^{mal} \prec CBF_i^{chal}] \leq (1 - e^{\frac{n(1-\rho_{i,l})\cdot u}{\phi}})^{(\rho_{i,l}^{claim} - \rho_{i,l})n \cdot u} \quad (3)$$

which corresponds to $\mathcal{P}_{adv}[\rho_{i,l} \rightarrow \rho_{i,l}^{claim}]$ if $\rho_{i,l}^{claim} - \rho_{i,l}$ is a multiple of $\frac{1}{n}$ (if $w_{CBF_{i,l}^{mal}}$ is a multiple of u) to satisfy the second of the aforementioned conditions (otherwise the claimed counting Bloom filter would be rejected). □

7.3 Typical figures

To illustrate the performance of the global solution more concretely, we provide some figures of a typical scenario.

First of all, the maximum number of selection criteria that can be used in each query should be reasonably small as it directly leads to an increase in the communication and computation complexity. We therefore set this maximum number to $n_{max} = 20$.

The level of security in groups over elliptic curves depends on a security parameter called the MOV degree [19]: by carefully choosing the elliptic curve it is possible to adjust the trade-off between key size and computation time, while maintaining a given level of security. We choose a curve with a small MOV degree of 2 and a group of order q of 512 bits length to have a security equivalent to 1024 bits RSA.

The size of an encoded query is then less than $2q \cdot n_{max} \approx 20$ Kbits. To put this size into perspective, note that in the case where there is no privacy protection (where queries and replies are sent in clear) and where each selection criteria is stored in a string with 16 8bits-characters, the size of queries is approximately 2.5 Kbits. The size of encoded queries is therefore 8 times larger than their queries in clear, but this is a deliberate choice to optimize the computation performance. If the communication overhead is considered as more important, it is possible to use curves with a higher MOV degree of 6: in that case it is possible to consider groups of smaller order and the overhead would be reduced to 2.5 times.

Concerning the parameters of counting Bloom filters, in addition to n_{max}, we need to define ϕ and u.

First of all, u is used as a security parameter, since the probability of success of an adversary can be bounded by 2^{-u}. As explained in section 4.2, it is not necessary to choose a very high value for u as it does not lead to revealing a secret but only to being able to cheat on the matching ratio. By choosing $u = 10$ for example, the probability of success of an attacker would still be bounded by 10^{-3} in the most favorable case. Other probabilities of success are presented in table 1. This table shows that the probability of success for significant attacks is very low (for reference the typical security margin for symmetric encryption is $2^{-80} \approx 10^{-24}$). It is of course possible to choose a higher value for u to make sure that even in the most favorable case the attacker would not succeed with probability more than 2^{-80} but u impacts first on the construction of counting Bloom filter (each element requires the computation of u hash values) and second and more importantly on the size of counting Bloom filters. We therefore believe that choosing a smaller value for u (as we did) is a better trade-off.

Table 1. Probability $\mathcal{P}_{adv}[\rho_{i,l} \rightarrow \rho_{i,l}^{claim}]$ of an adversary $\mathcal{S}^l$ with legitimate matching ratio $\rho_{i,l}$ to successfully claim a matching ratio of $\rho_{i,l}^{claim}$ with an encoded query Q_i' containing n selection criteria. The general parameters used for the counting Bloom filter are $n_{max} = 20$, $u = 10$, and $\phi = 289$.

n \ $\mathcal{P}_{adv}$	$0 \rightarrow \frac{1}{n}$	$0 \rightarrow \frac{2}{n}$	$0 \rightarrow \frac{1}{2}$	$0 \rightarrow 1$	$\frac{1}{2} \rightarrow \frac{1}{n} + \frac{1}{2}$	$\frac{1}{2} \rightarrow 1$	$1 - \frac{1}{n} \rightarrow 1$
6	5.10^{-8}	3.10^{-15}	1.10^{-22}	2.10^{-44}	9.10^{-11}	7.10^{-31}	2.10^{-15}
10	5.10^{-6}	2.10^{-11}	2.10^{-27}	4.10^{-54}	1.10^{-8}	1.10^{-40}	2.10^{-15}
20	1.10^{-3}	9.10^{-7}	7.10^{-31}	5.10^{-61}	5.10^{-6}	4.10^{-54}	2.10^{-15}

The number of positions ϕ of the counting Bloom filter according to the strategy explained in section 4.2 should be $\phi = \left\lceil \frac{n_{max} \cdot u}{\ln(2)} \right\rceil$ which is equal to 289 when $n_{max} = 20$ and $u = 10$. We choose to allocate 4 bits for each position in the counting Bloom filter, thus the total size of the filter is slightly more than 1 Kbit while the probability of a bucket overflow to happen would be less than 2.10^{-12}. The size of the counting Bloom filters is therefore really negligible in comparison with the size of the queries, thus the use of counting Bloom filters really offers a decisive advantage from a performance perspective on top of the advantage from a privacy point of view.

On this matter, we mentioned in section 4.1 that the size of the set of possible preimages that lead to a counting Bloom filter is around $\frac{q}{\phi^u} \approx 2^{448}$. This proves that a brute-force attack to break the privacy-preserving properties of the computation assurance solution is out of reach of current computing power.