Privacy-Preserving Patient Tracking for Phase 1 Clinical Trials

Hanna Ibrahim Farah

Thesis submitted to the Faculty of Graduate and Postdoctoral Studies in partial fulfillment of the requirements for the

Doctorate in Philosophy degree in
Electrical and Computer Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering

uOttawa

University of Ottawa
Ottawa, Ontario, Canada

Abstract

Electronic data has become the standard method of storing information in our modern age. Evolving from paper-based data to electronic data creates opportunities to share information between organizations in record speeds, especially when handling large data sets. However, sharing sensitive information creates requirements for electronic data exchange: privacy requires that the original data will not be revealed to unauthorized parties. In the healthcare sector in particular, there are two important use cases that require exchanging information in a privacy-preserving way.

1. Contract research organizations (CROs) need to verify the eligibility of a participant in a phase 1 clinical trial. One criterion is checking that an individual is not concurrently enrolled in a trial at another CRO. However, privacy laws and the maintenance of a private list of participants for competitive purposes prevent CROs from checking against that criterion.
2. A patient's medical record is usually distributed amongst several healthcare organizations. To improve healthcare services, it is important to have a patient's complete medical history: either to help diagnose an illness or to gather statistics for better disease control. However, patient medical files need to be confidential. Two healthcare organizations cannot link their large patient databases by disclosing identity revealing details (e.g., names or health card numbers).

This thesis presents the development and evaluation of protocols capable of querying and linking datasets in a privacy-preserving manner: *TRACK* for checking concurrent enrolment in phase 1 clinical trials, and *SHARE* for linking two large datasets in terms of millions of (patient medical) records. These protocols are better than existing approaches in terms of the privacy protection level they offer (e.g., against dictionary and frequency attacks), of the reliance on trusted third parties, and of performance when performing blocking. These protocols were extensively validated in simulated scenarios similar to their real-world counterparts.

The thesis presents novel identity representation schemes that offer strong privacy measures while being efficient for very large databases. These schemes may be used by other researchers to represent identity in different use cases. CROs may implement the protocols (and especially TRACK) in systems to check if an individual exists in another CRO's dataset without revealing the identity of that individual. Two healthcare organizations may use a system based on this research (and especially the SHARE protocol) to discover their common patients while protecting the identities of the other patients.

Acknowledgments

Thank you God for all Your blessings throughout this journey.

The encouragement to pursue higher education at the doctorate level was initially provided by my father, Ibrahim, who has always shown me a great example of smart and ethical working habits. My mother, Alice, complemented these traits with hard work, sacrifice, and love. Thank you for being supportive and loving parents. My sisters, Widad and Liza, also provided support and love as well as their families.

Dr. Daniel Amyot, my official supervisor, provided me with great opportunities, kept me on track and followed up with me through an excellent collaboration to complete my thesis. Thank you Daniel for being a great supervisor. The chosen project was performed at the Electronic Health Information Laboratory under the supervision of Dr. Khaled El Emam. Thank you Khaled for taking me on board, and for providing space and guidance on this project. Aleksander Essex was a postdoctoral fellow that provided feedback about the security of the protocols. Thank you Aleks for your valuable help, suggestions and assistance.

This thesis would not have been possible without the support of the National Science and Engineering Research Council of Canada through the Canada Graduate Scholarship program. Additional support from the Ontario Graduate Scholarship program and the Golden Key International Honor Society helped during the last phase of my research. Thank you to the University of Ottawa Startup Garage program for providing us temporary resources to attempt to translate this research into a business product.

A number of friends supported me along the way, encouraged me, studied with me, provided personal support, entertainment and a good push towards success. Thank you all.

Table of contents

List of figures

List of tables

List of acronyms

Acronym	Definition
ACRO	Association of Clinical Research Organizations
ACM	Association for Computing Machinery
CD	Central Database
CLNR	Common Length Name Representation
CPSO	College of Physicians and Surgeons of Ontario
CRAC	Clinical Research Association of Canada
CRO	Contract Research Organization
DoB	Date of Birth
EHIL	Electronic Health Information Laboratory
EM	Estimation-Maximization
FN	False negative
FP	False positive
FS	Fellegi-Sunter
HCN In	Health Card Number Interleaving
IEEE	Institute of Electrical and Electronics Engineers
KH	Key Holder
LSH	Locality Sensitive Hashing
LSUC	Law Society of Upper Canada
NPV	Negative Predictive Value
OCR	Optical Character Recognition
OHIP	Ontario Health Insurance Plan
PC	Pair completeness
PPRL	Privacy Preserving Record Linkage
RR	Reduction ratio
SC	Secure Comparison
SMC	Secure Multi-party Computation
SSL	Secure Socket Layer
TN	True negative
TP	True positive
TTP	Trusted Third Party

Chapter 1. Introduction

Healthcare organizations face challenges in collaborating with each other to share patient information in a privacy-preserving manner. Information sharing yields to better healthcare services and improves patient safety. This thesis investigates two practical healthcare use cases and provides collaborative solutions that meet their privacy requirements.

The primary use case is about *phase 1 clinical trials*[1] that aim to assess the safety of a new drug. *Contract research organizations* (CROs) have difficulties recruiting participants for these trials because they do not offer a potential cure for a disease at this stage. As an incentive for participation, CROs offer the participants a monetary compensation. While the participants are required to wait for a specific period of time before taking part in other trials (in order for the drugs to wash out of their bodies), some people are lured by the monetary compensation therefore ignoring the waiting period and taking part in multiple trials at the same time. This phenomenon has been documented in Appendix A and endangers the safety of the participants and the integrity of the trials. CROs require a solution for checking if a participant has been enrolled at a different CRO clinical trial site within a specific period of time in a privacy-preserving manner: not revealing the identity of the participant. There are two reasons why CROs cannot share the personal information of the participants in their trials: the first is the privacy laws (e.g., Canada's Personal Information Protection and Electronic Documents Act[2]) and the second is the competitive edge of finding phase 1 participants. *Health Canada* presents guidelines for good clinical practice in an industry guide [54] on drugs and health products. They mention the problem of vulnerable subjects that participates in trials with expectations of obtaining benefits. Moreover, they note that "the confidentiality of records that could identify subjects should be protected, respecting the privacy and confidentiality rules in accordance with

[1] According to DeMets et al. [33] (see also http://en.wikipedia.org/wiki/Phases_of_clinical_research), clinical trials involving new drugs are commonly classified into several phases (labeled 0 to 5), which can span many years. *Phase 1* clinical trials usually involve testing a drug on 20-100 healthy volunteers to determine whether the drug is safe (at certain dosages) to further check for efficacy on a larger scale.
[2] https://www.priv.gc.ca/leg_c/leg_c_p_e.asp

the applicable regulatory requirement(s)". *Clinical Trials Ontario* [30] presents a report with a definition of a technological system for clinical trials. They suggest the usage of a cloud solution because it would be common to all clinical trial sites as individual sites do not have the technical expertise to choose and to maintain independent systems. They specify that the "clinical trials cloud" should "provide a user-friendly interface with simple controls", "provide a secure and user-friendly platform", and "provide secure data/document encryption to protect patients". The system that we will present in this thesis is compliant with these requirements whether it is the user friendly aspect (screenshots available in Appendix B) or the privacy requirements. Last but not least, the *Canadian Institute for Health Research* presents a document for best practices for protecting privacy in health research [20] where they stress on ten elements including one (#7) that states "Institutions or organizations where research data are held have a responsibility to establish appropriate institutional security safeguards. Data security safeguards should include organizational, technological and physical measures" and another one (#8) partially stating "there should be strict limits on access to data and secure procedures for data linkage".

The secondary use case expands on the main one and is concerned with two healthcare organizations wanting to find out the patients they have in common in order to build more complete health records or cross-match information for statistical purposes. Our research was expanded to provide a solution to linking two entire databases with millions of records in a privacy-preserving manner. This is an emerging research area known as *privacy-preserving record linkage*. Our original technique could be used to match two records, but since the number of record pairs to compare is much larger than for a clinical trial, we also need to research private blocking and/or efficient techniques for linking very large data sets in terms of millions of records. *Blocking* splits a dataset into smaller blocks in order to compare a record only against the records in its matching block. Using blocking improves the linking efficiency in terms of time performance and allows organizations to use it in practice.

In the healthcare context, organizations that agree to share information in a privacy preserving way are both benefiting from that arrangement. In such a situation, it may be reasonable to design a protocol based on a *semi-honest adversary model* [53]. The semi-honest adversary model assumes that all parties will follow the protocol steps but could try to learn as much information as possible from the data they are given. On the other hand, a *malicious*

adversary model takes into consideration that a party may modify, spoof, or send wrong information on purpose in order to break the protocol and uncover confidential data. The remaining model is one involving a *trusted third party* (TTP). A TTP would be an organization that is trusted to operate in a professional and ethical way. It has the appropriate processes to deal with and maintain confidential information. However, trust is very hard to earn. Privacy concerns may stop many organizations from adopting a protocol based on a TTP.

In the following, we outline the motivation and problem statement for our main use case. The secondary use case is fully discussed in Chapter 4 (motivation, problem statement, additional literature review, etc.) in order to avoid confusion by jumping back and forth between the two use cases.

1.1. Motivation

Contract research organizations run multiple clinical trials on behalf of pharmaceutical companies. They are given a contract and are required to finish the trial by a given deadline. When recruiting clinical trial participants, the trial manager screens them for eligibility at the beginning of the trial based on multiple criteria including enrolment in a single trial at a time (including washout periods). If a participant is found to be taking part to another trial at any point in time during the study, he or she will be disqualified and the CRO would need to find another candidate to replace him or her, resulting in delays of the trial's final result. Monetary loss also incurs when the CRO is required to replace a participant in the trial. It is therefore very beneficial for all CROs in proximity (where the cost of traveling from one site to another is less than the compensation given for a trial) to be able to check if a participant has been or is trying to participate to multiple clinical trials at once. We present a protocol that enables clinical trial sites to verify this eligibility criterion while maintaining the privacy of the participants.

1.2. Problem statement

The main problem we aim to address in this thesis is the search for an identifiable entity in a private dataset created from multiple sources without giving information to the dataset holder that could leak the entity's identity or help discover it. We are adopting the example of

healthcare organizations wanting to check if one of their patients is also the patient of a different organization with the goal of checking eligibility in a clinical trial or building complete and accurate health records.

In the context of clinical trials, an organization wants to check if a participant has taken part in another clinical trial within a specified date range. The organization will check against a central dataset populated from several clinical trial sites. The organization wants to make sure that no identifying information about its participant will be revealed, nor any information that could lead to the discovery of their identity.

1.3. Use case

We allow multiple clinical trial sites to become part of a group that stores information about participants in their phase 1 trials in a *central database* (CD) that would be hosted by a semi-trusted third party that follows the protocol properly. The group members need to be within a small geographic distance of each other (e.g., a metropolitan area) since individuals in general will not travel long distances to participate in multiple trials as it will become more expensive for them and less profitable. When a site needs to check whether a participant has taken part in another trial and has not completed their required waiting period, it can check that with the central database in a privacy-preserving manner.

Checking against the CD proceeds in two steps that are consistent with the typical screening workflow for phase 1 clinical trials. In the first step, individuals contact the site and volunteer over the telephone: they have usually heard about the study through advertisements or friends and acquaintances, or sometimes they are contacted directly by the site because they had participated in studies there in the past and were considered to be good candidates. Those who pass the telephone screening are invited to come to the site for additional tests. These additional tests may include the completion of detailed questionnaires, and possibly laboratory tests on blood and urine collected at the site.

In the second step, volunteers are screened again when they come for the visit to the site. This ensures that they have not enrolled in other studies since the telephone interview, and they are asked to provide documentation to confirm their identity.

1.4. Research method

The research method employed for this thesis is "design and creation" described by Oates [88]. This method applies an iterative and interactive cycle and suggests five steps for performing research: awareness, suggestion, development, evaluation, and conclusion. These steps are not necessarily strictly followed one after the other. When suggesting a solution, the researcher can learn more about the problem and become aware of other influential factors. Developing the suggested solution also teaches us more about the problem, and different challenges could arise that require further research and the development of new ideas. The evaluation step might also show that the suggestion was not appropriate and requires the researcher to go back and make the necessary changes. The essence of the "design and creation" method is learning through making. This method was a natural fit to our research because we were learning about the challenges faced in phase 1 clinical trials and experimenting with different techniques to create an appropriate solution.

In the following, we will describe the elements at each stage of our research and present a diagram showing the interactions and jumps between the different stages of the design and creation research methodology. A letter-based labeling system is used to show the chronology in the text and to represent the steps in Figure 1 in order to show the methodology at a glance (the sequence of events starts at A, then continues at B, and so on). Please note that the conclusion part of the research methodology is omitted from the diagram but discussed in the text in section 1.4.5. Protocols P1, P2, and P3 correspond to the versions of the TRACK protocol discussed in Chapter 3, while P4 represents the SHARE protocol discussed in Chapter 4.

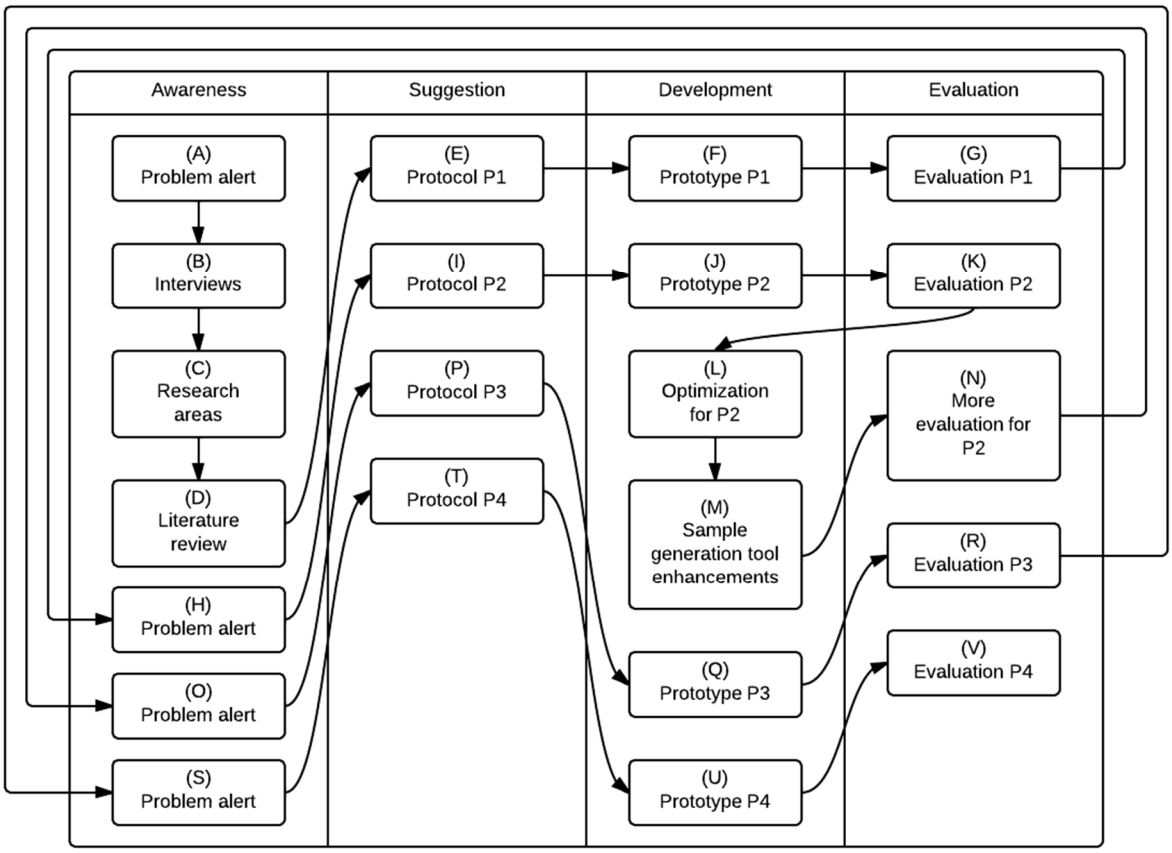


Figure 1 - Research method overview

1.4.1 Awareness

(A) The problems with phase 1 clinical trials were brought up to the attention of the principal investigator of the Electronic Health Information Laboratory [3] (EHIL), Dr. El Emam, by representatives of contract research organizations. (B) We conducted interviews with clinical trial managers to learn more about their processes, environment, and the problems they encounter in these trials. We realized that we could help them discover participants that are not eligible to take part in the trial because of dual enrolment but we needed to ensure the privacy of these participants. (C) The research areas associated with these problems were private string comparisons and private record linkage. (D) After performing a literature review, we did not find a solution that was a good fit for this problem.

[3] https://www.ehealthinformation.ca/

A protocol and a prototype based on Bloom filters were designed, developed and evaluated (steps E, F, and G discussed in section 3.2.1). (H) However, we became aware of a limitation in the privacy guarantees of Bloom filters in the way it was suggested to use them broadly in the literature and further publications questioned the appropriateness of the privacy provided by Bloom filters [78][79][86]. Another suggestion was investigated (I).

(O) A security analysis of our protocol showed the potential possibility of an attacker learning to recover names that are very lengthy and rare. Since our top priority is protecting the privacy of every individual, we had to devise techniques to guard against this potential problem.

(S) After having developed an appropriate protocol that solved the problem of phase 1 clinical trials, we were also made aware that the private record linkage problem was challenging other organizations in Canada that wanted to utilize a protocol to link very large databases in terms of millions of records. The solution we had for phase 1 clinical trials was not appropriate for very large amounts of data, so we had to think of new suggestions.

1.4.2 Suggestion

(E) We designed a protocol P1 that utilizes commutative hash functions (see section 2.1.1) and *Bloom filters* (see section 2.1.2) in order to solve the phase 1 clinical trials problem.

(I) However, given the discovered limitations of Bloom filters, we decided to try another technique that offers better privacy guarantees. We chose the *Pallier homomorphic cryptosystem* [91] in our protocol P2 in order to compare string parts instead of storing the parts in a Bloom filter. We no longer needed the commutative hash function as a method to protect the data in a central database because Pallier is known to be secure and offers randomness in the encodings so that the same value could be encoded into different cipher texts.

(P) After discovering that our protocol could be subject to an attack described in section 3.2.2, we added guards and enhanced the privacy of the final version of our protocol (named *TRACK*) discussed in section 3.4. A new name representation was also developed (discussed in section 3.3.6) to guard against frequency attacks at both the storage level and the runtime of the protocol. We also switched to using the exponential ElGamal cryptosystem instead of Paillier's

because the former offers the same properties but is ten times more efficient than the latter in terms of computation [31].

(T) The next step was to expand the protocol or develop another one suited for linking two large databases. *Blocking* is known to improve the efficiency of record linkage by splitting a large amount of data into several blocks and comparing selected blocks of data against each other instead of comparing the two entire databases. In our case though, privacy is a must so we could not just use any blocking technique. We reviewed the literature for private blocking methods and decided to explore and evaluate the idea of *hashing and bucketing* as a private blocking strategy in our protocol P4 (later named *SHARE*).

1.4.3 Development

(F) A proof of concept of the initial protocol P1 was designed, developed and deployed online on TrialGuardian.com[4]. We developed the user interface that a clinical trial site can use as well as the server functionality that could be hosted by a semi-trusted third party. We had the ability to register clinical trial sites in order for each one of them to check participants' eligibility and add them to the central database according to the protocol.

(J) After modifying the protocol to use Pallier encryption instead of Bloom filters with commutative hash functions, we implemented the changes on a local computer in order to evaluate the new protocol P2 in terms of efficiency. (L) We introduced many optimizations to the comparison algorithm in order to obtain better performance. Our optimization brought us a 90% improvement in performance.

(M) In order to perform additional evaluations, we developed a sample generation tool named *DataRealtor*, described in section 3.5, to allow the specification of prevalence and the generation of multiple versions of the data sets used in the evaluation. We also developed an evaluation tool named *ProEvalNet* to take advantage of multiple computers, which is described in section 3.8.

---

[4] Please note that this proof of concept is no longer publicly available. Screenshots and explanations are however available in Appendix B.

(Q) In order to maintain privacy as our number one concern in the protocol, we added guards and additional features to our protocol to guard against a potential attack (with the right resources at hand) that we discovered and describe in section 3.2.2. We developed a new name representation scheme that guards against frequency attacks in P3.

(V) We extended our prototype to implement P4 with a hashing and bucketing strategy, which improves the performance drastically and allows us to compare very large data sets containing millions of records if they have a unique record identifying field. P4 also introduced a new name representation scheme that is more efficient and can be taken advantage of in a scenario involving two organizations looking to find out which records they have in common.

1.4.4 Evaluation

(G) After the proof of concept for protocol P1 was developed, we designed an experiment to evaluate it. We were interested in finding out the efficacy of the Bloom filter string comparison method for different types of potential errors: insertions, substitutions, or deletions. An insertion involves the addition of a character or more to a string, while a substitution involves the replacement of a character by another one and a deletion is the omission of one or more characters. The results for all three types of modifications were similar to the ones using standard techniques for string comparison including Jaro-Winkler and Levenstein distance [57]. Another experiment with data sets containing mixed types of errors similar to the ones found in real-world data sets was also performed and produced results similar to both standard methods (Jaro-Winkler and Levenstein distance). However, we became aware of a problem with the privacy guarantees using the Bloom filter method and had to devise another methodology.

(K) The new method we were evaluating was based on the Pallier cryptosystem. We discovered the method was not as efficient as we hoped it would be so we had to optimize the protocol in order to get the desired performance. The experiments that helped us optimize the protocol included using selected fields from a record instead of all the fields and checking if we still have high accuracy (described in section 3.3.4). The experiments allowed us to reduce the number of required fields to four instead of six: last name, gender, date of birth and health card number. Other algorithmic optimizations were also performed and evaluated in section 3.3.3.

(N) Further evaluation of the protocol P2 was performed taking into account the prevalence value in the data sets in order to be evaluating a situation as close as possible to the real world.

(R) After improving the privacy measures in our protocol P3, we performed additional performance testing to determine the efficiency of our new string comparison method. The results are discussed in sections 3.3.5 and 3.3.6.

(V) The next evaluation of the protocol P4 had to deal with accuracy and performance using the hashing and bucketing private blocking strategy. The strategy showed to have the same accuracy as P3 while having a considerable performance gain.

1.4.5 Conclusion

We developed a new privacy-preserving protocol for tracking participants in phase one clinical trials. We took into consideration the limited computational power available to clinical trial sites and the strong privacy requirements for healthcare data. Our protocol was thoroughly evaluated and enhanced over several iterations in order to provide a novel approach that is secure, effective, and practical for usage in a real world scenario. Furthermore, we expanded our approach by reviewing the literature and experimenting with hashing data structures to recommend the best settings for a private blocking strategy to link datasets containing millions of records. The protocols we developed are based on a solid privacy foundation, namely the exponential ElGamal cryptosystem, which is widely used for industrial applications today.

1.5. Thesis contributions

The major contributions of this thesis are the following:

- A privacy-preserving identity representation scheme using a minimalistic set of fields and a technique called *Health Card Number Interleaving* (HCN In) for representing a last name [45].
    - Guards against dictionary and frequency attacks.

- Privacy-preserving querying protocol named *TRACK* (corresponding to protocol P3 in Figure 1).
    - Tailored towards the phase 1 clinical trials real-world scenario.
    - Enables checking the existence of an entity in a dataset created from multiple sources in a privacy-preserving manner.
    - Guards against dictionary and frequency attacks.
    - Accounts for misspellings in names and performs probabilistic matching.
- Privacy-preserving protocol named *SHARE* (corresponding to protocol P4 in Figure 1) for linking two large datasets in terms of millions of records.
    - Introduces a new *Common Length Name Representation* (*CLNR*)
    - Provides strong privacy measures.
    - Guards against dictionary and frequency attacks.
    - Provides a linear performance with the size of the dataset.

Additional minor contributions include:

- The *DataRealtor* tool [43] (Appendix C) to generate datasets from populations created in Febrl [21].
    - Allows the specification of prevalence values in datasets.
    - Allows the generation of multiple samples given a range of sizes.
- The *ProEvalNet* (Appendix C) tool to evaluate and compare various algorithms on multiple computers [44]
    - Works standalone or as a distributed application on several computers.
    - Capable of resuming the evaluations if a crash occurs.
    - Enables remote tracking of the progress of the evaluation.
- Datasets (Appendix C) for future researchers based on the 1990 US Census, the database of the College of Physicians and Surgeons of Ontario and the database of the Lawyers Society of Upper Canada, the North Carolina voter's registry and parts of the Australian phone book independently.

The following papers, based on this thesis, have been published or submitted for publication:

- Farah, H., Amyot, D., and El Emam, K.: Real-World Data Set Parameters and Synthesization for Matching Identity in Clinical Protocols. In Computer-Based Medical Systems (CBMS), 2014 IEEE 27th International Symposium on. IEEE, pp. 263-266, 2014.
- Farah, H., Amyot, D., and El Emam, K.: A Tool for Simple and Efficient Clinical Protocol Evaluation. In Computer-Based Medical Systems (CBMS), 2014 IEEE 27th International Symposium on IEEE, pp. 499-500, 2014.
- Farah, H., Amyot, D., and El Emam, K.: Efficient Privacy-Preserving Identity Scheme for Electronic Validation of Phase 1 Clinical Trials. 6th International MCETECH Conference. Lecture Notes in Business Information Processing, vol. 209. Springer, pp. 183-196, 2015.
- Farah, H., Amyot, D., and El Emam, K.: Common Length Name Representation: An Efficient Privacy-Preserving Scheme. In proceedings of ICSE 2015, 1st International workshop on technical and legal aspects of data privacy and security. Florence, Italy. 2015 (to appear).
- El Emam, K., Farah, H., Samet, S., Essex, A., Jonker, E., Kantarcioglu, M., Earle, C.: A Privacy Preserving Protocol for Tracking Participants in Phase I Clinical Trials. *Submitted* to the Journal of Biomedical Informatics. November 2014. Revised Feb. 2015.

1.6. Thesis outline

In Chapter 2, we conduct a literature review to outline what progress has been made in the privacy-preserving record linkage methodologies. We also highlight the weaknesses of the existing techniques. Chapter 3 describes the TRACK protocol, which enables clinical trial sites to query a central database in a privacy-preserving manner to verify the eligibility of a participant in a clinical trial. An evaluation of the protocol in terms of accuracy and performance is also presented. Chapter 4 discusses privacy-preserving record linkage applications and introduces the SHARE protocol for large data sets (in terms of millions of records) and compares it to recent private blocking techniques. Chapter 5 concludes the thesis and describes the limitations of this work and relevant threats to validity.

Chapter 2. Literature review

In this chapter, we conduct a literature review on the existing techniques for private string matching and private record linkage. Some of the researchers who performed reviews in these areas are Trepetin [122], Verykios et al. [133], Hall and Fienberg [53], Karakasidis and Verykios [70], Durham et al. [36][38], and Christen in his data matching book [22]. We discuss the methods that were developed to address the privacy-preserving record linkage challenge as well as their advantages and disadvantages. At the end of the chapter, we categorize the major drawbacks of the existing methods, which we address in the design of our own private record querying and linkage protocols.

In order to perform a comprehensive literature review, we reviewed all the references of the aforementioned surveys and books. We considered the references of every relevant paper thereafter. We also searched several online databases including the ACM Digital Library and IEEE Xplore. The following query alert was created on Google scholar several years ago: "privacy preserving record linkage". The alerts helped us keep up to date with new publications related to our topic. We also contacted and consulted the personal websites of the key authors in this area including Christen and Verykios to stay up to date with their work.

2.1. Private string matching

Private string matching techniques are a key piece of private record linkage. Records are compared by comparing the values of their attributes. First names, last names, or street names are just a few fields that could belong to a record that needs to be compared to another record in a privacy-preserving manner. Private string matching could either compare the strings for an exact match or for an approximate match. Approximate matching has been shown to produce better record linkage results [97][123].

There are many private string matching techniques described in the literature. Trepetin [122], Durham et al. [38], Bachteler et al. [7] have produced surveys that outline these methods, which we discuss below.

Trepetin evaluated four techniques that have been proposed for character-level privacy-preserving comparison. He argued that the available techniques fell short of providing a complete protocol for private matching. Some methods could not fully protect privacy: Churches et and Christen [29] suggested splitting strings into *bigrams* (i.e., sequences of two adjacent characters in a string) and create a power set of all bigram combinations to compare against another string, which may be subject to a frequency analysis attack. Other methods could not perform similarity comparisons in order to match more records pairs: Van Eycken et al. [127] suggested hashing a group of linkage identifiers to create a key used for matching, whereas Song et al. [114] suggested encrypting various permutations of an original identifier to account for misspellings but the typographical mistake that occurs in an identifier might not be one that was accounted for. Other methods required an unrealistic amount of computing power: Du et al. [35] suggested encrypting all permutations of an original identifier to account for misspellings.

Bachteler et al. evaluated the methods of Pang and Hansen [93] based on reference tables, of Scannapieco et al. [109] based on embedding spaces, and of Schnell et al. [113] based on Bloom filters. They noted that the methods of Bouzelat et al. [14], Atallah et al. [4], Ravikumar et al. [104], and Churches and Christen [29] either require high computing demands or produce high rates of false positives.

Durham et al. [38] only considered string fields in their analysis; they evaluated comparators that handle mainly strings and did not evaluate the techniques that were found unsuitable by other researchers, for example: guess and encrypt.

We will outline the various methods in the following subsections and discuss their advantages and disadvantages.

2.1.1 Exact matching using hashing

Hashing is a method for transforming an alphanumeric value into another alphanumeric value or number known as *hash code*, from which the original value is extremely hard to be derived. A

hash function can use a key to generate the code. A hash function, given the same parameters, will always transform the same input into the same code. However, a small variation of the input produces a completely different code.

This technique can be used to protect a participant's information: instead of storing the information itself, the codes that were generated from that information would be stored. If intruders gain access to the codes, it will be very hard for them to recover the original information. In order to compare information to the stored code, we calculate the code for that information and check if the same code exists in the database. For example, if we apply a hash function to the string "school", we get number 1562356 and store that number in a database. If an attacker is looking at "1562356", it is very difficult to find out that the string for which that number was generated is "school". However, if we know that we want to check whether the string "school" exists in the database, we apply that hash function to it and obtain the same result "1562356", whose existence is easy to check in the database.

Commutative property

Some hash functions are commutative. To illustrate the commutative property of a hash function H(), we can use the following example. If a site A uses $H_a()$ with key 'a' to hash data and then sends the result to site B that uses $H_b()$ with key 'b' to hash it again, we would obtain the same final result of the hashed data if site B hashed the initial data first and then sent it to site A for the final hashing. That is: $H_a(H_b(data)) = H_b(H_a(data))$. The benefit of this property is high when we have multiple sites that need to perform hashing operations: each site can have its own secret key independently from the rest, so each site's hash is different but the final hashed result will be the same. A central server can store that final hashed result while the data is kept private.

The fact of not having a common secret key provides better data privacy as one secret key being compromised does not allow an intruder to discover any additional information from the central database. In our scenario, multiple web sites will want to store and check information against a central database. An attacker will need the secret keys of *all* the sites in order to perform a dictionary attack on the central database.

Advantages and disadvantages

The advantage of hashing techniques is that hashing functions are fast to compute. They scale well if we have a large number of records to compare.

While this method seems to be protecting the privacy of the string being compared, there are two attacks that can be used against it.

- A *dictionary attack*: in order to discover the original string given its hash code, an attacker can use the same hash function to obtain the hash values of all the words in a dictionary, or all the names in a phone book, and then compare them to the hash code he is trying to break. Dictionary attacks can be mitigated by adding a *salt* (a secret prefix or suffix) to a string before hashing it.
- A *statistical attack* can also reveal the values of certain hash codes: because the name distributions are not the same within a population, and because a hash function given the same parameters will produce the same hash code for the same name, an attacker is able to calculate the distribution of the hash codes and compare it to the distributions of the names of the people in the area (from a phonebook using the area code), hence allowing him to deduce some relationships between the original names and the hash codes that follow a same distribution (i.e., most common or least common).

Another disadvantage of hashing methods is that there is no notion of *string similarity*: if a small typing mistake occurred while entering the name of a person to be matched (or if the mistake was already in the database), a hash function will produce a code that is very different from the one produced for the original string. If two strings are very similar, their hash codes are still very different. In order for two strings to be classified as a match using their hash codes, they need to be *exactly* the same. Hashing cannot perform relative similarity comparisons. There are only matches and non-matches. Other techniques do exist and can account for small differences in a string while still classifying it as a true match to its original one. They will be explained in the next sub-sections.

2.1.2 Bloom filters

A Bloom filter is a means of compacting the storage of multiple elements, e.g., names, addresses, and web addresses. It is used to save time and space because it does not store the actual element. Instead, a short representation of the element is created. One can check if a given element belongs to the set stored inside the Bloom filter. One can also compare the similarity of two sets stored in two distinct Bloom filters.

A Bloom filter is a data structure that consists of a series of bits that can be set to either 0 or 1. Initially, all the bits in the data structure are set to 0. The storage of an element is achieved by mapping the element to be stored to multiple indices in the Bloom filter using several hash functions to obtain these indices and setting the bits at those indices to 1. When storing multiple elements, some of the elements might share common bits at the same indices. For the interested reader, a small example of the usage of Bloom filters can be found in the example of section 3.3.1.

When the Bloom filter contains a set of elements, one can check if a given element belongs to that set by calculating the indices of the element in the Bloom filter and checking if the bits at those indices are all set to 1. Bloom filters allow false positives when checking for membership in a set: even if the bits at the calculated indices are all set to 1, the element might not have been previously stored because of potential overlaps (a bit could have been set to 1 only because it is a common bit with another element that was stored). The probability of false positives can be controlled by tuning the values of the parameters of the Bloom filter such as its size and the number of hash functions used to perform the mapping. By design, false negatives are not possible: if one of the bits is set to 0, there is no way that the element was ever stored in the Bloom filter because that bit would have had to be set to 1.

One can also compare the similarity of two Bloom filters by calculating a value called the *dice coefficient*, which will be illustrated in section 3.2.1.

Bloom filters have been widely addressed in the literature since their introduction by Burton H. Bloom in 1970 [12]. Bloom demonstrated the use of his filters in a hyphenation program: most words can be hyphenated by applying a few simple rules but some words (around 10% of them) require a lookup table. The Bloom filter would store a dictionary of all the words

that require lookup. In order to hyphenate a word, the program would check if the word belonged to the dictionary stored in the Bloom filter. If it does not find it, it applies the simple rules. Otherwise, it knows that it needs to check the lookup table. A false positive in this case means that even if a word was not in the lookup table, the program will still attempt to look it up.

Broden and Mitzenmacher [17] conducted a survey on the applications of Bloom filters. This data structure was used to store an entire dictionary of words, which was later utilized by a spell checker program in early versions of the UNIX operating system. If the word was a member of the Bloom filter, that meant that it was spelled correctly. The Bloom filter data structure allowed significant space savings (instead of storing and saving an entire dictionary) at a time when memory was still very expensive. A false positive match in this application though meant that an incorrectly spelled word might go unnoticed. Bloom filters were also used to store a set of unsuitable passwords for security purposes. That technique was also extended so that passwords with only one modification from the words in the filter would also not be accepted as suitable passwords. In that application, a false positive meant that a suitable password could be rejected. However this was not a major issue as the user can just make some modification to the password and continue his task. Another early application of Bloom filters was in databases, to speed up semi-join operations. In a distributed database environment, suppose we wanted to know how many students in a course are international students, but the database containing the students registered in a course is different from the one containing the information about students' original addresses. Instead of sending the list of all the students in the course to the address database where international addresses will be detected, the address database could send a Bloom filter containing the names of students with international addresses to the courses database where a list of only the names that were matched against the Bloom filter will be returned to be checked for false positives. We would save sending a big part of the class list if the percentage of international students was small.

Jain et al. [63] used Bloom filters to compare documents retrieved from web sites in order to refine web search results. In order to store a document in a Bloom filter, they split it into multiple chunks that composed the elements inserted in the Bloom filter. Two documents are compared for similarity by comparing their Bloom filters.

A technical report by Bellovin and Cheswick [9] in 2007 described how to use encryption and Bloom filters for privacy enhanced searches between two parties without requiring a trusted middle man. The first party can query the other party's database while maintaining search queries private.

Private record linkage with Bloom filters is described in [113]. They show that comparing encrypted data using Bloom filters for record linkage produces similar results in terms of recall and precision as using non-encrypted identifiers. In our scenario, we use a similar approach but we do not have a common encryption key that needs to be used by all parties wishing to create Bloom filters.

Durham et al. [37] used Bloom filters for private medical record linkage with approximate matching. They showed that their technique outperformed deterministic and probabilistic matching techniques; however, it is susceptible to many attacks that can breach privacy [78][79][86].

Kuzu et al. [79] studied the security of record linkage using Bloom filters. They proposed an attack based on constraint satisfaction that can lead an attacker to discover 11% of the records' real identities if they were encoded using the current recommendations in the literature (i.e., strings split into bigrams in order to store them in a Bloom filter). The authors proposed a method to tune the parameters of the Bloom filter encoding in order to be more resilient to such an attack.

Advantages and disadvantages

The advantages of Bloom filters are that they provide compact storage, they are easy to compute and to compare, and they provide accurate comparison results. Their parameters have to be well tuned. However, they are still susceptible to frequency analysis and statistical attacks and therefore cannot guarantee the privacy of the data they store [78][79][86].

2.1.3 Trigrams

This method is described by Hylton [60]. It consists of splitting a string into trigrams (groups of three characters) and of applying a hash function to these trigrams. Comparing two strings consists of computing the number of occurrences of each trigram in the two strings. Again, this

method is subject to statistical attacks. Durham describes it as highly accurate but less secure than other methods of string similarity measure [38].

2.1.4 Embedding

This method, explored by Inan et al. [61] and Pang et al. [92], uses a set of reference strings and calculates distances to that set with the theory that similar words will have similar distances from the set. It then compares distances instead of actual words to preserve privacy.

This method was criticized because its performance is highly dependable on the set of chosen reference words. Durham reports that it does not perform well with a true positive rate of 0.13 [38] which means that only 13% of the strings compared and classified as true matches actually were true matches, the others being false positives.

Bonomi et al. [13] proposed a new embedding scheme based on frequent grams mined from the original databases (that need to be linked) and provided a formal proof of privacy based on differential privacy. The usage of frequent grams instead of random strings as in [109] provided better accuracy. However, their method relied on a trusted third party to perform the linkage step. The method also produces the same embedding for the same string so it does not guard against frequency attacks.

2.1.5 Phonetic encoding and encryption

A phonetic filter produces a phonetic code given a string. This does not apply to numerical fields. The phonetic code can be the same for two strings that are not equal but similar. For example, “Sandra” and “Sandro” both have the phonetic code “S536” using Soundex [69].

A phonetic code technique also requires secret sharing because it encrypts the phonetic code using a symmetrical encryption method: both parties need to have agreed on common secret keys for encryption and decryption. If a third party discovers the shared code, they can gain information on the data.

Another issue with phonetic encoding is that it is not robust when there are errors in the initial characters or because of truncation variations [29]. For example, a unique code, C623, is

generated for three names, “Christine”, “Cristina”, and “Christopher”, while each of “Chris” and “Kristine” generate different codes, C620 and K623 respectively.

2.1.6 Edit similarity

Atallah et al. [4] present this method to compare two strings without revealing what any of these strings is. The method calculates the number of steps (i.e., insertions, deletions, or substitutions) it would take for one string to match the other. Durham describes this method as the best in privacy protection but is infeasible in real-world applications because it is computationally inefficient: it would take over 2 years to compute a one-million record pair comparison [38] (e.g., on a 2.5GHz quad-core workstation with 4GB of memory).

2.2. Private record linkage

2.2.1 Early concepts from the 1990s

Traditionally, data *linkage* required the assistance of a third party, which would handle identifying information with confidence [24]. However, better privacy protection could be established. The concept of private data matching emerged in the 1990s. The initial approaches were based on transforming strings into codes such that different strings produce different codes and that it was very hard to recover the original string given the code. This transformation was first based on polynomial coefficients [102] and was further improved by using a one-way hash function, message digest function, or public key scheme for the encoding of the original strings [39][100][101][111].

To improve the matching accuracy of data that is *dirty*, i.e., containing nicknames, name variations or other types of modifications, Quantin et al. [102] present a pre-processing approach tailored for French names where name variations are converted to a standardized format. However, their method could also produce false positives if the data was clean because names that were genuinely different would be converted to the same standardized form and classified as a true match.

After the strings have been encoded, the two parties wanting to perform data matching send their encrypted record identifiers and encoded attribute value to a third party. The third party can perform exact matching on the attributes by comparing their encoded values and deciding whether two records match based on a threshold if multiple attributes are used for matching. The usage of a third party is mandatory because the two parties can decode each other's data by comparing the encoded values to a list of encoded values from a public directory (each party can encode a public directory and compare it to the other party's encoded values). The two parties need to agree on a secret key that is used while encoding the values and that should remain unknown to the third party performing matching.

2.2.2 Two-party protocols

Agrawal et al. proposed a *secure protocol* [1] where each of the two parties has its own unencrypted dataset. They first suggest using hash functions and commutative encryption but this scheme allows the other party to launch a dictionary attack or a statistical attack because the hashing is deterministic. To solve this problem, they use a semantically secure encryption scheme. They propose a number of protocols including computing the equijoin, set intersection, and set intersection size of the two datasets. Their protocols, however, require both parties to perform computations on the other's encrypted records, which would not be suitable if the sites had limited computational power (conventional laboratory machines) because the encryption keys need to be thousands of bits long in order to provide adequate security [53].

Atallah et al. presented a two-party protocol for securely calculating the edit distance between strings [4]. Their approach has considerable communication cost [71] and is unsuitable for private record linkage where a large number of records need to be compared.

Ravikumar et al. [104] presented a two-party protocol for computing string distance measures as well as Euclidean distance. Their protocol is based on the secure set intersection protocol described by Vaidya and Clifton [125] and also requires a large amount of computation, making it unsuitable for record linkage of large databases.

Freedman et al. [50] proposed a private record linkage protocol in which a user encrypts each element in its input set using the *Pallier homomorphic cryptosystem* and sends it to a server.

For each input in the server's dataset, the server homomorphically evaluates the user's input and returns the encrypted result to the user. The user decrypts and checks the result: if the user's input is contained in the server's dataset, the result is equal to zero. Otherwise, the result is a random number. There is no third party. This protocol requires the server to perform computations on the entirety of its database. It is again not suitable for linking large databases.

Li et al. [83] criticized Agrawal's protocol [1]. They showed that it could not protect against a spoofing attack (one of the parties pretending to be someone else) if one of the parties was malicious. However, this does not mean that the protocol is not good. The choice of the protocol depends on the setting that the private linkage will occur in. Li et al. described three design criteria and two threat models. The design criteria include leaking some or no information, protection against spoofing or not, and symmetric release versus asymmetric release of information (only one party finds out the result). The threat models include semi-honest versus malicious and small versus large domains. Li mentions that large domains are safer because an attacker cannot find out the entirety of the information with a single query. Li suggests three protocols and their modifications to include a digital signature from the data sources in order to protect against spoofing. The suggested protocols include using a trusted third party, hashing, or Agrawal's protocol described in their own terms. All three protocols will have a computational overhead because of the digital signatures. In our case, we do not want to rely on a trusted third party, and hashing is not suitable because it does not protect against dictionary and statistical attacks. Agrawal's protocol has already been documented to have a heavy computational burden and with added digital signatures, Li definitely does not improve on its computational complexity.

Weber et al. [134] proposed a simple protocol where two sites can maintain a shared database. They claim that they opted out from using a Bloom filter approach because it needed highly specialized programmers, which is not the case at all. A Bloom filter is a simple data structure that can be easily manipulated. They used the first two letters from the first name and last name and the date of birth of the individual to create an anonymized database. People with short names can be very easily identified using this method. It is definitely not suited for the strong privacy guarantees we are looking to establish.

Lai et al. [81] proposed a protocol based on Bloom filters where several sites would exchange partial segments of Bloom filters in order to determine which records they have in common. Their proposal is at a very high level of abstraction, without a prototype, a substantial analysis or evidence. The privacy offerings of this protocol are highly questionable; the authors claim an adequate level of privacy, which would not be suitable for sensitive medical information. This protocol could also be subjected to the cryptanalysis attacks described by Kuzu et al. [79].

2.2.3 Three-party protocols

Bouzelat et al. [14] suggested a protocol that can overcome typographical errors and provide privacy given a trusted third party (TTP). To overcome minor typing mistakes, phonetic encoding is applied where possible. Then, the two database owners apply a common pad before hashing their data to prevent a dictionary attack. All data is sent to the TTP to be compared and matched. This protocol is still subject to frequency attacks.

Churches [28] proposed a method for improving the protection of privacy in disease registers. However, his method requires a TTP. This TTP will map identifying information for an individual in the general population to other identities in disease registers such that health events will be specific to various registers and only certain registers will be informed of relevant events such as cancer diagnosis. As Churches is aware, it may be hard to obtain public consensus for the establishment of such a system because a breach or collusion with the TTP will expose the details of all the patients.

Schadow et al. [110] proposed a protocol for querying multiple data sources and aggregating their responses via a mediator site. The protocol they describe is based on keyed *hash-based message authentication codes* that do not guard against statistical attacks. All the sites being queried need to share the key, which is described as semi-public, and the mediator needs to be trusted not to seek that key and discover the identities of the individuals to be returned to the querying party. For the sensitive nature of medical information, various other protocols provide better privacy for healthcare applications.

Churches and Christen [29] presented a protocol that requires splitting a string into *q-grams*, i.e., a group of smaller strings based on consecutively splitting the original string into smaller groups. Each string would be represented by a power set of its subgroups. Each q-gram would be hashed using a common secret key for the two database owners. These owners create collections with the group of q-grams, the number of q-grams in the group, and the total number of q-grams, and then send them to a third party for matching. The authors also recommend inserting *dummy* q-gram groups in order to prevent frequency attacks. This protocol has a large communication and computational cost [23] because it relies on power sets of q-grams [24][133] and is still susceptible to frequency attacks for the groups of q-grams of size one [25][122].

In his thesis [121], Trepetin suggests a technique for splitting a string into a series of characters and using a second identifier from a record for padding every single one of them. This method allows a third party to compute the similarity of two strings but would require the existence of a second identifier that is always correct and that is always available. Another problem might arise if the second identifier had only a couple of values to choose from (for example if the database owners decided to choose gender as the second identifier); the encodings would then become subject to a frequency attack.

O'Keefe et al. [87] presented a generalization of the protocol of Agrawal et al. using a different topology. They have two data custodians with databases holding information about overlapping individuals, and a third entity that wants to receive a linked data set without the protocol revealing personal information. This protocol can match items only if they have the exact same value (and hence is not accounting for small typing mistakes) [23].

Pang [93] suggested a protocol where two database owners agree on a set of common reference strings. Each party will compare each appropriate identifier from each record to the entire set of reference strings. The comparison produces a distance value. If that distance is less than a chosen threshold, the corresponding reference string is encrypted and added to a group containing an identifier number and the distance value. Each identifier is represented by a series of groups as defined earlier. Both parties send their groups to a third party that will compare them against each other. To classify two identifiers as a match, their distances from a common reference string should be below a specified minimum value. The performance and accuracy of this protocol is variable: it is based on the selection of the set of reference strings. Bachteler et al.

conducted an empirical study that showed this protocol has poor precision and recall and requires longer run times than other protocols in the literature [7].

Schnell et al. [113] presented a 3-party protocol for private record linkage based on Bloom filters. When comparing string fields, the authors split the string into q-grams that are then hashed using a number of different hash functions to calculate indices in the Bloom filter, which will be set to one. The third party will compare Bloom filters using the *dice coefficient* approach. The authors claim that their method produces similar results to non-encrypted identifiers and is superior to phonetic encoding. However, they do face the risk of frequency attacks especially against short names.

Durham improved on the Bloom filter technique of Schnell et al. [113]. Her technique is more secure given the random bit sampling from different attributes: instead of building a separate Bloom filter for each field in the record, Durham suggests representing the entire record in a single Bloom filter [36]. She suggests that her method makes frequency attacks less likely because the domain from which the single filter is constructed is much larger than separate domains for separate filters. She also provides guidance on which bits and which attributes to select from as well as choosing the correct size for each segment of the single Bloom filter.

Randal et al. [103] use the same Bloom filter method developed by Schnell et al., but they do not show that it satisfies the recommendation of Kuzu et al. [79] for privacy protection against cryptanalysis. They make the size of the Bloom filter smaller and reduce the number of hash functions (both by 10), using a Bloom filter with 100 bits instead of 1000 bits, and 3 hash functions instead of 30. There is no discussion of the security of their Bloom filter design. They use bigrams to populate it though they have also tried it with trigrams and it gave similar results. The blocking they use is by hashing the soundex of the surname with the first initial but the soundex mostly applies to the beginning of a name so it is questionable if adding the first initial helps. They also hash the date of birth for blocking, which can lead to statistical attacks.

Pommerening et al. [96] suggested a protocol where a trusted third party is involved. The data holders send their records without encryption to the trusted party. The trusted party hashes and encrypts the personal identifiers and sends the records to another data custodian, which will

provide linking services to different projects in different organizations. This technique does not provide privacy guarantees and is subject to frequency attacks by the linking party.

Karakasidis et al. [73] suggested a method using phonetic encoding to account for typing mistakes, coupled with encryption for privacy. The method also injects fake phonetic codes to protect against frequency attacks. Phonetic encoding yields more false positive matches than string similarity functions [51].

Mohammed et al. [85] presented a protocol for secure data integration. They considered the cases for a semi-honest model and a malicious attacker model. Their protocols however are tailored towards the financial industry and do not preserve the entire original contents of the records. The financial company can still use the data to make decisions on loans but in a medical setting, the institution requires all the details of the record to make the appropriate decision for a patient.

Kantarcioglu et al. [66][67] proposed a secure equijoin protocol for private record linkage with four parties: the two data holders, a data site, and a key holder. When a data holder submits a query to the data storage site, this site computes the join of the two data sets and sends the results to the key holder, who owns the private key of the cryptosystem. The key holder decrypts the match results to find the matches, and sends the final result to the inquirer. This protocol is not suitable for linking large databases as it offers quadratic run time. The authors suggested an optimization to their technique by means of revealing certain attributes and maintaining k-anonymity status. However, the ability to reveal such attributes will depend on the organization's privacy constraints.

El Emam et al. [41] developed a secure protocol to link medical databases of patients with the human papillomavirus and compute statistics on them. They defined the requirements for a secure protocol and outlined the drawbacks of other protocols that cannot provide all the security needs to implement such a service. Their protocol consists of multiple registries with patient information and several aggregators that compute intermediate results. When required statistics need to be computed, the registries are asked for a list of patients with specific attributes. The registries select the appropriate patients and send a hashed value of each patient identifier (to which a random number was added to prevent a dictionary attack) to a random

aggregator. The aggregator will then ask each other registry to commutatively hash the initial values it received. At the end, different aggregators will have lists of patients from different registries that match the required attributes for the desired statistic. Using a secure multi-party computation, the different aggregators can find out the patients they have in common and send the required data to a public health unit, which will have the final answer for the statistic it requested. The drawback of their protocol however is that it provides matching based on exact values of fields, which will miss some positive matches due to typographical errors. It also relies on a common identifier to be present in all the different registries.

Some federated identity management techniques [95] attempt to create an identity provider and separate health data from the original identity of the patient. The identity provider needs to be a trusted third party because it handles the linking of identities of real patients across several organizations.

2.3. Drawbacks of existing methods

2.3.1 Frequency attacks

Techniques that are based on deterministic encryption are usually susceptible to frequency attacks. The party receiving the "so thought" secure encrypted data can perform a frequency analysis on selected fields in order to uncover the original data. For example, if the attacker has access to the demographic or census data of a particular population, he could calculate the frequency of a selected field and try to map it to the encrypted data frequencies to deduce a relationship between the two.

To illustrate the feasibility of a frequency attack, consider a town where "Smith" is the most popular last name. If a database administrator owns a database with encrypted last names of patients of that town's hospital, he can calculate the frequencies of each encrypted value. The value with the highest frequency can be assumed to map to "Smith". Using the same technique, the database administrator can try to uncover the values of other fields and breaching the confidentiality of the database.

The techniques that rely on deterministic encryption are: Pommerening et al. [96], Dusserre et al. [39], Bouzelat et al. [14], Quantin et al. [100][101][102], Churches and Christen [29], Al-Lawati et al. [2], Scannapieco et al. [109], Bonomi et al. [13], Schnell et al. [113], and Karakasidis and Verykios [71].

2.3.2 Scalability issues

In private record linkage, three factors are in competition: security, performance, and accuracy. A protocol that offers the best security or best accuracy usually falls behind other protocols when it comes to performance. While the choice of the protocol will depend on the organization's priorities, some protocols require a massive amount of computing power that renders them unrealistic for practical use. Other protocols could be useful for very small data sets but impractical for linking large databases.

The techniques that are not suitable for private record linkage of large databases are: Ravikumar et al. [104], Atallah et al. [4], Freedman et al. [49], Agrawal et al. [1], O'Keefe et al [87], and Inan [61].

2.3.3 Secret sharing

Secret sharing is required when symmetric encryption is being used between two parties to encrypt their data. Both parties need to be using the same key for encryption in order for a third party to perform record linkage without knowing the original values but for the same values in distinct databases to have the same encrypted representation. Secret sharing between two database holders can be dangerous because if the secret key at one site is compromised, the attacker can also decrypt all the records of the other site. It can be argued that public key encryption using a trusted third party is also as dangerous, but the choice of the third party should be an organization that is very knowledgeable and professional with computer security whereas a regular organization would have much more vulnerabilities.

The techniques that rely on secret sharing are: Pommerening et al. [96], Al-Lawati et al. [2], Karakasidis and Verykios [71], and Schadow et al. [110].

2.3.4 Record linkage quality

Phonetic encoding

In general, phonetic encoding methods yield more false positive matches than string similarity functions [19][51][106]. Some of the techniques that rely on phonetic encoding include Karakasidis and Verykios [69][73].

Exact matching only

Real-world data sets are dirty: two records in different databases that are referring to the same individual could contain typing variations in different fields. For example, an individual named John could also be named Jon (missing the h) in another database. If only exact matching is used to perform record linkage, the process will produce additional false negative results.

The techniques that rely on exact matching are: Berman [10], O'Keefe et al. [87], Dusserre et al. [39], Bouzelat et al. [14], Quantin et al. [100][101][102], El Emam et al.[41], and Schadow et al. [110].

Reference strings

Protocols relying on embedding strings as vectors based on distances from a set of reference strings or using other methodologies based on reference strings do not have a consistent performance or accuracy measurement. The performance and accuracy of the protocol will largely depend on the chosen reference strings. It is rather risky for an organization to attempt to use such protocols unless they have been tested and proven to work with their datasets (which might be difficult if privacy needs to be maintained at all time).

The techniques that rely on reference strings are: Pang and Hansen [93], and Scannapieco et al. [109].

2.3.5 Lack of privacy

Most protocols rely on encryption or hash functions in order to transform the plain text data into an encoded version that cannot be deciphered easily. This is usually the standard method of data protection. However, other researchers suggested separating the medical data from the identifying data as a measure to protect privacy. Their technique can easily be criticized because

revealing the identity or the organization and the patients could be enough to invade their privacy. For example, if the organization is a cancer center, all the patients belonging to that organization have cancer with a very high probability.

One sample technique that relies on sending personal identifying information while keeping the medical information is Kelman et al. [75]. Federated identity management also separates health data from identity data and requires a trusted party to manage the identity information [95].

Other techniques use partial identifying information which could still easily give out the identity of an individual, e.g., Weber et al. [134].

Trusted third party

Protocols would be considered secure and maybe we would not need any privacy preserving protocols if people were trustworthy. However, from a research perspective, our aim is to design a secure protocol without betting on the unchanging trust of the data custodians.

The techniques that require a trusted third party are: Bonomi et al. [13], Karakasidis and Verykios [72], Churches [28], Schadow et al. [110], and Peyton and Hu [95].

2.4. Summary

Private record linkage has gained a lot of attention in recent years. We discussed numerous approaches and showed their drawbacks. Table 1 provides a summary of the protocols discussed in the literature review. Although some of the protocols offer methodologies that are good in certain scenarios, we have not seen a solution capable of providing privacy based on a known hard problem, that is performing in less than one minute and low computational cost to discover if an entity is present in a dataset created from multiple sites that conforms to our scenario. In the following chapter, we will consider the case of querying a dataset for a patient in a privacy-preserving way and present our new protocol (TRACK), which takes into consideration the computational constraints and high privacy demands of clinical trial sites.

Table 1 - Protocols' characteristics and drawbacks

Authors	Technique	FAP	LCPS	EH	NTP
Agrawal et al. [1]	Hashing, secure equijoin, set intersection				✓
Al-Lawati et al. [2]	Hashing, secure set intersection				✓
Atallah et al. [4]	Secure edit distance	✓		±	✓
Berman [10]	Exact matching		✓		±
Bonomi et al. [13]	Embedding		✓	✓	
Bouzelat et al. [14]	Phonetic encoding, hashing		±	±	±
Churches [28]	Based on trusted third party		✓		
Churches and Christen [29]	Q-grams, symmetric encryption, subsets	±		✓	✓
Durham [36]	Bloom filters, locality sensitive hashing	±	±	✓	±
Dusserre et al. [39]	Public key scheme		±		±
El Emam et al. [41]	Hashing with random values, SMC	±	✓		✓
Freedman et al. [50]	Paillier, set intersection				±
Inan [61]	Value generalization, SMC		✓	±	±
Karakasidis and Verykios [69]	Phonetic encoding		±	±	±
Karakasidis and Verykios [71]	Phonetic encoding, shared key, Bloom filter		±	±	±
Karakasidis and Verykios [72]	Reference table, k-anonymity	±	±	±	±
Karakasidis et al. [73]	Phonetic encoding, encryption	±	±	±	±
Kelman et al. [75]	Sharing identifying information		✓		
O'Keefe et al. [87]	Based on Agrawal [1]				✓
Pang et al. [92]	Public reference table		✓	✓	±
Pang and Hansen [93]	Reference strings, distance calculation		±	±	±
Pommerening et al. [96]	Based on trusted third party		✓		
Quantin et al. [100][101]	One-way hashing		±		±
Quantin et al. [102]	Standardisation, hashing		±	±	±
Ravikumar et al. [104]	Euclidian distance, secure set intersection	±		±	±
Scannapieco et al. [109]	Sparse map, Euclidian distance		±	±	±
Schadow et al. [110]	Hash-based		±		±
Schnell et al. [113]	Bloom filters		±	✓	±
Weber et al. [134]	Anonymization		✓	±	±

Legend: **FAP** (Frequency Attack Prevention), **LCPS** (Low Computational Power at Site), **EH** (Error Handling), **NTP** (No Trusted Party required). '✓' for well supported, '±' for partial support, and '' for no support.

Chapter 3. Clinical trial TRACK protocol

Contract research organizations run hundreds of clinical trials per year to evaluate new drugs for safety and efficacy. The phase 1 of these trials consists of testing the safety of a drug; therefore, it does not require the individual being tested to have the disease the drug is aiming to cure. However, it is important that an individual takes part of a single trial at one particular moment in time and that he refrains from participating in another trial prior to a *washout period* determined by his current trial participation. Unfortunately, people do not always follow the rules. In phase 1 trials, some people are lured by the monetary compensation offered in return for their participation. These people attempt to participate in more than one phase 1 trial at a time without the knowledge of the trial managers; we call them *concurrent* enrollers. This phenomenon introduces health risks to the concurrent enrollers caused by unexpected drug interactions and damages the integrity of the trial itself: a safe drug could be deemed unsafe in some trials that enroll a small number of participants if one of them shows adverse events. CROs can benefit from a privacy preserving protocol that allows multiple clinical trial sites to discover if a participant is enrolled or has been enrolled within a specific time period at another trial.

The literature review revealed gaps in the existing protocols that could be adopted as privacy-preserving querying protocols for phase 1 clinical trials. We researched the required properties and designed our own protocol in an iterative manner, improving it based on our findings, attacks published in the literature specifically on Bloom filters [78][79][86], and protection against frequency attacks based on knowledge of demographic information. While designing our protocol, we also actively developed the ProEvalNet tool to generate the data sets for evaluating it and to perform the evaluations on multiple machines.

In the following, we present the requirements, methods, development steps, and an evaluation of our privacy-preserving phase 1 clinical trial protocol that enables multiple contract research organizations to check if a participant meets an eligibility criterion to enter their phase 1 clinical trial. The criterion that our protocol verifies is the eligibility date: is the participant eligible to participate in our trial at a specific date? The participant should not be concurrently

enrolled in another trial at that date and should not have participated in a previous trial within a specified time period of that date for the criterion to be satisfied. We designed the protocol for a healthcare setting taking into consideration that healthcare sites have limited computational power and that participant privacy is their highest concern. The limited computational power criterion is especially important because the healthcare setting is currently adopting mobile devices to manage patients and information. We named the protocol *TRACK* because our end goal is to track the existence of an individual in different databases.

3.1. Requirements

In order to introduce the requirements for a well thought privacy-preserving solution to our problem, we take into consideration the attacks described in the literature, the advice of security experts in the EHIL laboratory and professors at the School of Electrical Engineering and Computer Science at the University of Ottawa and the considerations of real-world feasibility and practicality. We present the following requirements pertaining to tracking a participant in phase 1 clinical trials:

1) The protocol shall not reveal the identity of the participant to the database it is checking against.
2) The protocol shall encrypt or protect participants' identity information.
    a. No party shall be able to uncover the identity of the participants of the sites.
3) When tracking an individual, the sites shall only learn whether the individual's information provided in the query matches an entry in the database or not.
4) The protocol shall be resistant to frequency attacks that could lead to the discovery of the identity of some people in the databases.
5) Sites shall not be able to view or access any record belonging to another site.
6) The site queries shall allow exact matching (e.g., for health insurance card numbers) or approximate matching (e.g., for names because real health records have errors in them including typing mistakes in patient names [56]).
7) Sites shall be able to implement the protocol with limited computational power.

a. In order for a solution to be used in practice, we have to account for sites with limited computational power, i.e., containing only regular computers and not powerful servers.

8) The protocol shall be able to handle information from multiple sites.
    a. A clinical trial site will be matching against several different sites and their participants.

3.2. Iterations

As we outlined in our iterative process in Figure 1, our TRACK protocol has undergone major changes since its initial conception. However, the protocol always had three main components: the clinical trial sites, a central database (CD), and a key holder (KH) described in section 3.4. The first iteration was based on Bloom filters where a cryptanalysis revealed certain vulnerabilities. The second was based on secure multi-party computation using the Paillier [91] cryptosystem. We discovered certain vulnerabilities and required performance improvements which were done in the third and final iteration that was based on the ElGamal cryptosystem. We discuss the three iterations in the following sub-sections.

3.2.1 Iteration 1

The first iteration of the protocol (P1 in Figure 1) had at its core a method based on commutative hash functions and Bloom filters. When a site required to check or to add a participant to the central database, that site and each site in the group had to hash the information in order for it to be compared against the central database or added to it. Dictionary attacks were prevented by each site having a secret key (used by it only when hashing). Frequency attacks were prevented by monitoring how many hashing requests were asked of the site. We conducted extensive experimentation on Bloom filters and their ability to detect string similarities when inserting, deleting, or substituting characters. Our experiments allowed us to fine tune the parameters of our methods to obtain the best accuracy. However, since we are not using Bloom filters in the latest protocol, we decided to omit these experiments from the thesis. While we were evaluating more aspects of our protocol, an attack was published by Kuzu et al. [79] that allowed the recovery of some original text values by analyzing data stored in Bloom filters using the

parameters suggested in the literature. Since our primary goal is to protect privacy, we decided to research another method of computing string similarity in a privacy-preserving manner. Further attacks have been recently published in the literature on Bloom filter encodings and studies determined that there has not been enough analysis yet to determine how much privacy these encodings are capable of providing [78][86].

3.2.2 Iteration 2

By conducting a more in-depth literature review and following recommendations from colleagues at the Electronic Health Information Laboratory (EHIL), we decided to base our protocol on the Paillier *homomorphic cryptosystem* (Paillier and exponential ElGamal exhibit the same homomorphic properties that were required by our protocol, these can be reviewed in section 3.3.2) and use the *dice coefficient method* (to be seen in section 3.3.1) to compare encrypted bigrams in order to compute string similarity in a privacy-preserving way. This approach (called P2 in Figure 1) offers strong privacy measures for the encoded values as the Paillier homomorphic cryptosystem we used is semantically secure. During iteration 2, we required performance enhancements to be developed. Therefore we conducted experiments (detailed in section 3.3.4) to determine the best combination of fields from a record that can be used for matching while maintaining high accuracy. The experiments showed that the required fields for matching could be reduced to the following: last name, date of birth, gender, and health card number. Afterwards, a member of the EHIL team expressed a concern of a potential attack scenario that we evaluated in the following.

Feasibility of an attack

Iteration 2 of protocol for private record linkage gave away the following information when comparing a record against a database:

- The number of q-grams in each string
- The number of common q-grams between the string to be compared and a string from the database

Given the previous two pieces of information, the following attack can be mounted by the semi-trusted third party performing the decryptions to attempt to discover the names of the participants when a request is received to check the similarity of a string S_1:

1. Create a list of names from a public source, a phone book for example, having the same number of q-grams as S_1
2. Create a statistics table comparing that list to every other name in the public source with the following information included: length of the name being compared to, and number of common bigrams
3. Each time S_1 is compared to another element in the database, the number of common q-grams and the length of the name it is compared to allow the reduction of the possible matches using the statistics table of the public source as will be shown in the following example.

We assume that the string to be compared belongs to that list and that the goal of the attack is to reduce the number of strings in the list to only one (which would be it).

Example

Table 2 represents the statistics created using name databases (from the College of Physicians and Surgeons of Ontario and the Law Society of Upper Canada, detailed in section 3.7) for comparing a string of length equal to 5. The size of the list is equal to 3648. Every item in this list is compared to the rest of the names in the database. The statistics are organized by name length.

The analysis is performed in the following manner:

1. What are we comparing: a string of length 5
2. To what are we comparing it: to another string of length 5
3. How many bigrams are in common: 5 bigrams
4. The statistics table allows one to reduce the number of potential size 5 strings that are being compared from the full list of 3648 elements to only 59 elements given the information above.

Multiple comparisons of the same string to many elements will allow us to intersect the resulting lists of possible matches for further reduction of the space that the string can be selected from. For example, the first comparison compared a name of length 5 to another name of length 5 and discovered how many bigrams they have in common. Using the table, we obtained a set of possible names where this condition holds. When comparing the name to the second name in a dataset, we obtain another set of possible matches, and we can find the intersection of both sets and maybe reduce the space of possible matches further and so on. The real danger is when that space's size becomes equal to 1, which reveals the actual string being compared. We need to guard against such a possibility.

Table 2 - Attack feasibility on P2

Name length=5							
List size=3648							
#bigram matches>	0	1	2	3	4	5	6
Compared against list sizes V							
1	3129	512	7				
2	3648	3539	1758	89			
3	3648	3648	3538	1599	106		
4	3648	3648	3648	3327	1301	83	
5	3648	3648	3648	3606	2735	59	3612
6	3648	3648	3648	3629	2655	1041	110
7	3648	3648	3648	3626	2611	870	61
8	3648	3648	3648	3611	2514	751	77
9	3648	3648	3648	3564	2215	538	59
10	3648	3648	3647	3473	1764	403	45
11	3648	3648	3648	3247	1344	278	25
12	3648	3648	3636	2870	915	157	16
13	3648	3648	3613	2443	624	81	8
14	3648	3648	3528	1938	422	60	3
15	3648	3648	3397	1626	328	54	1
16	3648	3647	3226	1395	242	37	3
17	3648	3607	2330	657	97	2	1
18	3648	3500	1962	450	70	8	0
19	3648	3427	1721	375	47	2	2
20	3415	2252	735	154	19	0	0

3.2.3 Iteration 3

The third iteration addressed the privacy risks by modifying the representation scheme of the names in the databases and introduced performance enhancements by switching to the exponential ElGamal cryptosystem (detailed in section 3.3.2), which offered semantic security and homomorphic properties similar to Paillier while its operation was ten times faster. The changes also included using a maximum name length proven not to affect the protocol's accuracy by experimentation, the non-disclosure of individual name lengths to the party performing the decryptions, and the introduction of fake queries that would break the assumption that each query contains the information of an individual that is requesting the participation in a clinical trial.

This protocol is further described in this chapter in section 3.4 and it is the one that we adopted for checking the eligibility of participants in a phase 1 clinical trial.

3.3. Methods

In this section, we explain the methods used in our protocol for computing string similarity (section 3.3.1) and the cryptosystem in place to guarantee privacy (section 3.3.2). We also describe the optimizations we performed to speed up the computations and get rid of unnecessary computations in section 3.3.3, followed by additional optimizations that lead to the discovery of a subset of fields that can be used for matching identity in section 3.3.4. Finally, section 3.3.5 describes the techniques for choosing a limited number of bigrams from a last name and compares them experimentally to find out which one is most appropriate.

3.3.1 Dice coefficient for string matching

We initially adopted the dice coefficient method to compare the similarity of two Bloom filters. Other set similarity measures exist including the Jaccard coefficient. However, the dice coefficient was being used in the literature for Bloom filters and we decided to use it as well in our technique in order to compare results with fewer variables in the setup of the experiments. A Bloom filter is a data structure that can be represented by an array or zeros and ones. It can be populated by multiple elements by using a hash function to determine the index of the element in

the array and placing a value of one at that position. However, while we dropped the Bloom filter data structure, we were still able to leverage the concept of the dice coefficient in order to compute string similarity without the need of a Bloom filter. In fact, the information stored in the Bloom filter reflected the data stored in the bigrams of a string. Without the Bloom filter, it is still possible to compute which bigrams are similar, as we will illustrate in the following example.

Using the dice coefficient with a Bloom filter

The approach consists in splitting a string into bigrams (two characters at a time), which will create a set out of the string. Each element from the set is stored inside the Bloom filter. To add an element to the Bloom filter, we need to derive indices in the Bloom filter that correspond to that element and set the bits at those indices to 1 (the parameters used for the Bloom filter configuration are discussed in the example afterwards). An index is usually calculated using a hash function to get a numerical value for an element then using the modulus mathematical operator to get a value within the range of indices inside the Bloom filter. This allows us to compare two strings by comparing the Bloom filters that represent those strings.

The dice coefficient is: $DC = 2 * (|X \cap Y|) / (|X| + |Y|)$, where X and Y are the Bloom filters, |X| and |Y| are the numbers of bits in each Bloom filter that are set to 1, and $|X \cap Y|$ is the number of common bits in both filters that are set to 1. The higher DC's value, the more similar are X and Y.

Using the dice coefficient without a Bloom filter

The approach here consists in splitting a string into bigrams (two characters at a time), which will again create a set out of the string. Duplicate bigrams are removed to mimic the behavior of a Bloom filter (since the same bigram would hash to the same position in the Bloom filter). The bigram lists from two strings are compared in an optimized way to find out which ones are in common.

The dice coefficient here is: $DC = 2 * (|X \cap Y|) / (|X| + |Y|)$, where X and Y are now the lists of bigrams, |X| and |Y| are the numbers of unique bigrams in each list, and $|X \cap Y|$ is the number of common unique bigrams in both lists.

Example

The two names we are comparing are "John" and "Jon". We will split them into bigrams and pad the beginning and the end to have a fair bigram distribution: a fair distribution means each letter is represented the same amount of times, in this case, twice.

- John: _J, Jo, oh, hn, n_
- Jon: _J, Jo, on, n_

Computing the dice coefficient *with* the use of a Bloom filter

For a method utilizing Bloom filters with 187 bits and 6 indices per bigram (values chosen for this example based on formulas from Broder and Mitzenmacher [17] and a false positive rate of 0.001 given the relatively small amount of participants in a phase 1 clinical trial), we need to hash each bigram and derive multiple indices from the hashed value (instead of using several hash functions). To calculate an index, we use the modulus operator (%) on the partial hashed value to ensure it is in the range of the Bloom filter's size.

Let us compute the Bloom filter for John using some hash function *H()*:

- H(_J)= 5178906926994571884. Hashing bigram "_J" transforms it into a numerical value that we use in the following to derive 6 indices in the Bloom filter.
- Splitting the initial value into 6 parts turns it into: 517,890,692,699,457,1884
- Using the modulus operator (with 187 as an operand) on each value gives us the final indices: 143,142,131,138,83,14

H(Jo)= 271,820,056,996,972,8411; indices= (271%187=84),72,56,61,37,183
H(oh)= 655,497,616,190,865,3659; indices= 94,123,55,3,117,106
H(hn)= 729,626,258,230,420,1031 ; indices= 168,65,71,43,46,96
H(n_)=237,553,633,271,732,9039; indices= 50,179,72,84,171,63

The indices that will be set to 1 in the Bloom filter for John are the following (including duplicates): 3, 14, 37, 43, 46, 50, 55, 56, 61, 63, 65, 71, **72, 72**, 83, **84, 84**, 94, 96, 106, 117, 123, 131, 138, 142 ,143, 168, 171, 179, 183. They contain two pairs of duplicate values (72 and 84).

Let us compute the Bloom filter for Jon:

H(_J)= 517,890,692,699,457,1884; indices= 143,142,131,138,83,14

H(Jo)= 271,820,056,996,972,8411; indices= 84,72,56,61,37,183

H(on)= 465,704,869,890,883,8006; indices= 91,143,121,142,135,152

H(n_)= 237,553,633,271,732,9039; indices= 50,179,72,84,171,63

The indices that will be set to 1 in the Bloom filter for Jon are the following (including duplicates): 14, 37, 50, 56, 61, 63, **72, 72**, 83, **84, 84**, 91, 121, 131, 135, 138, **142, 142**, **143, 143**, 152, 171, 179, 183. They contain four pairs of duplicate values (72, 84, 142, 143).

After deriving all the indices for all the bigrams in each name, we can compute the number of unique indices that will be set to 1 in each Bloom filter.

- The Bloom filter for _John_ has 28 unique indices marked with 1.
- The Bloom filter for _Jon_ has 20 unique indices marked with 1.
- There are 16 common indices marked with 1.
- The dice coefficient is equal to: $2 * (|X \cap Y|) / (|X| + |Y|) = 2*(16/(28+20)) = 0.667$

Computing the dice coefficient *without* the use of a Bloom filter

- John and Jon have three bigrams in common: _J, Jo, n_
- John has five bigrams in total. Jon has four bigrams in total.
- The dice coefficient is equal to: $2*(3/(5+4)) = 0.667$

Please note that the two values for DC computed here with and without Bloom filters are coincidently the same, but this is not the case in general. The calibration of the DC cutoff value is dependent on the chosen method.

Dice coefficient cutoff estimation

An important question that we needed to answer was the following: what is the threshold value of the dice coefficient above (or equal to) which two strings are considered to be a match and under which they are considered to be different? We adopted the following dice coefficient estimation plan: at first, we determined an appropriate value for the dice coefficient based on the 1990 US census database [124]. Afterwards, we tested that value using the *College of Physicians*

and Surgeons of Ontario (CPSO) [117] and the *Law Society of Upper Canada* (LSUC) [118] databases by using the sensitivity, precision and negative predictive value metrics as it is typically done in the literature and shown relevant in section 3.3.4. The plan was further broken down as follows:

- US Census 1990 database:
    - Calculated the average dice coefficient for male first names, female first names, and last names.
        - We compared the names to their modified versions with the following modifications: 1 or 2 character insertions, 1 or 2 character substitutions, 1 or 2 character deletions.
    - Decided on a threshold dice coefficient value (for a true match).
        - How low can we choose the DC value?
- US Census, LSUC and CPSO:
    - Given a threshold value for the dice coefficient, we decided to evaluate how well the matching of the names in our databases is performed. We used the sensitivity metric as well as precision and negative predictive value metrics.

Table 3 shows the results of the dice coefficient values for each type of name and modification. The naming convention in the first column is as follows: [name type]_[number of modifications][type of modification]. The name type is either a female first name (*Female_First*), a male first name (*Male_First*) or a last name (*Last*). Each name has incurred one or two modifications that consist of deletions (*Del)*, insertions (*Ins*), or substitutions (*Sub*). The number of duplicate records compared is for informational purposes; it states the number of names that exist in the data set. The average dice coefficient is for all the names in a specific data set (for example, female first names with one deletion). The remaining columns show the minimum and maximum dice coefficients along with the matching names that produced them.

Table 3 - Min, Max, and Avg dice coefficient values for all names in 1990 US Census

DC for names of all lengths in 1990 US Census	**Num of duplicate records compared**	**Average DC**	**Min DC**	**Orig Name**	**Dup name**	**Max DC**	**Orig name**	**Dup name**
Female_First_1Del	4275	0.820	0.372	Ka	K	1	deedee	deede
Female_First_1Ins	4275	0.849	0.580	Ja	jau	1	annis	annnis
Female_First_1Sub	4275	0.775	0.357	Fe	fr	0.935	deeanna	deaanna
Female_First_1Del	4248	0.685	0.105	Hye	Y	0.96	jeanene	jeane
Female_First_2Ins	4275	0.778	0.347	Na	nsax	1	vada	vadada
Female_First_2Sub	4275	0.684	0.105	My	np	0.931	anastasia	anastatia
Last_1Del	88799	0.845	0.363	vo	V	1	noonon	nooon
Last_1Ins	88799	0.868	0.557	jn	jnl	1	torrez	torrrez
Last_1Sub	88799	0.803	0.301	ku	kj	0.976	kahoohalphala	kahoohslphala
Last_2Del	88698	0.723	0.036	ito	t	1	kaanana	kaana
Last_2Ins	88799	0.800	0.328	pu	pxun	1	girard	girarard
Last_2Sub	88799	0.718	0.115	cal	vao	1	farrer	farerr
Male_First_1Del	1219	0.804	0.408	al	l	0.981	tyrell	Tyrel
Male_First_1Ins	1219	0.843	0.612	bo	rbo	1	barry	barrry
Male_First_1Sub	1219	0.764	0.421	ed	ef	0.92	solomon	sololon
Male_First_1Del	1213	0.654	0	rey	e	0.962	nathanael	nathael
Male_First_2Ins	1219	0.763	0.416	wm	wumo	0.992	reinaldo	reinalaldo
Male_First_2Sub	1219	0.670	0.173	asa	jss	0.921	christopher	chrisfopher

We notice that the minimum dice coefficients are obtained for very short names. If we wanted to tolerate at least one substitution, insertion or deletion for names of all lengths, using dice coefficient values in proximity of the minimum dice coefficients will not be appropriate because they are low and will hence lead to many false positive matches for longer names. Figure 2 shows the distribution of name lengths in the US census database, which is useful to get the minimum dice coefficient for names with lengths that contribute significantly to the data set. A significant contribution is defined here as forming at least 8% of the dataset.

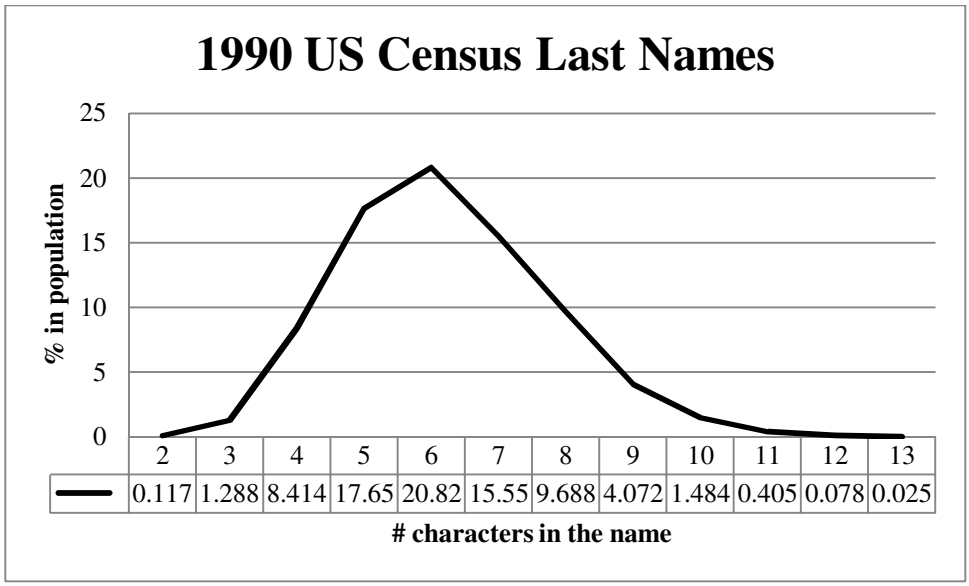


Figure 2 - Last names length distribution in 1990 US census

For last names, we notice that names with lengths between 4 and 8 characters are the most dominant.

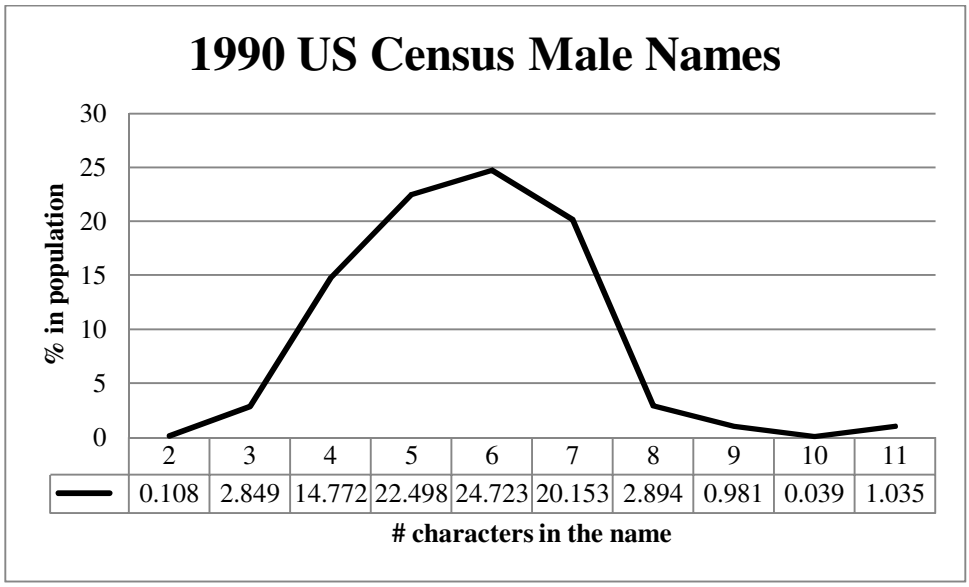


Figure 3 - Male first names length distribution in 1990 US census

For male first names (Figure 3), we notice that names with lengths between 4 and 7 characters are the most dominant.

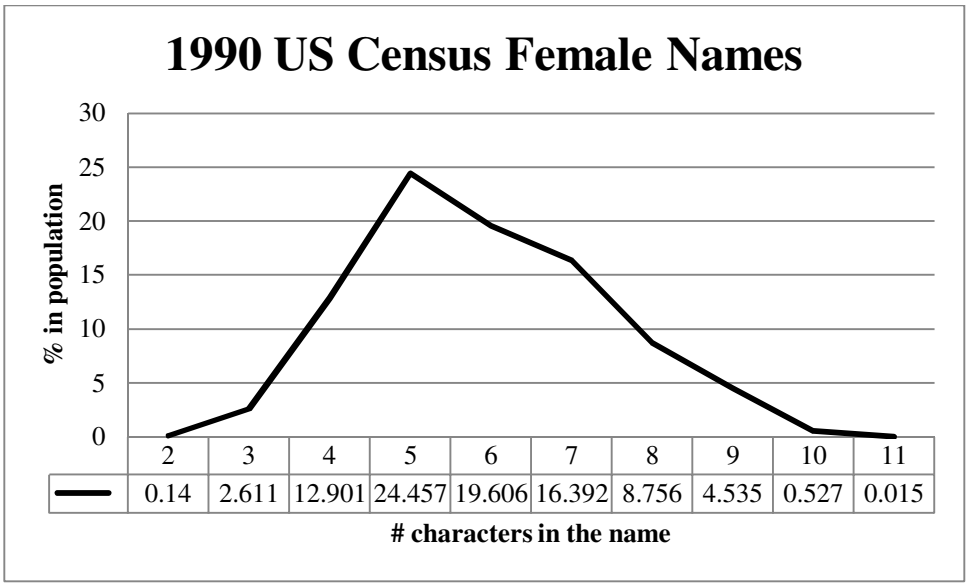


Figure 4 - Female first names length distribution in 1990 US census

For female first names (Figure 4), we notice that names with lengths between 4 and 8 characters are the most dominant.

Table 4 - Min, Max, and Avg dice coefficient values for name ranges in 1990 US Census

DC for names in a limited length range in US 1990 Census	**Num of duplicate records compared**	**Average DC**	**Min DC**	**Orig name**	**Dup name**	**Max DC**	**Orig name**	**Dup name**
Female_First_1Del	3833	0.825	0.621	Joan	joa	1	deedee	deede
Female_First_1Ins	3852	0.851	0.688	John	johnt	1	annis	annnis
Female_First_1Sub	3885	0.779	0.571	Joan	joam	0.935	deeanna	deaanna
Female_First_2Ins	3863	0.687	0.26	Kina	Kn	0.96	jeanene	jeane
Female_First_2Sub	3882	0.778	0.536	Deon	deyond	1	vada	vadada
Last_1Del	72899	0.835	0.628	Kone	kon	1	noonon	noon
Last_1Ins	72835	0.86	0.683	Bobo	boboh	1	torrez	torrrez
Last_1Sub	73010	0.792	0.567	Blue	blur	0.968	klaass	klaaas
Last_2Ins	72849	0.705	0.246	Vint	Vn	1	kaanana	kaana
Last_2Sub	73187	0.789	0.519	Sura	surxai	1	girard	girarard
Male_First_1Del	994	0.807	0.621	Joan	joa	0.981	tyrell	tyrel
Male_First_1Ins	1045	0.842	0.715	Rory	rorye	1	barry	barrry
Male_First_1Sub	1013	0.764	0.606	Loyd	liyd	0.92	solomon	sololon
Male_First_2Ins	1024	0.657	0.281	Karl	Kr	0.917	russell	rusel
Male_First_2Sub	1010	0.761	0.52	Ivan	ivadno	0.953	lloyd	lloyddd

Let us find the dice coefficient values considering only the records with name lengths in our calculated ranges above (Table 4). The *minimum* value for the dice coefficient, although now much higher than in Table 3, is still too low. The *average* dice coefficient for the most common name lengths is equal to 0.78. That value is larger than 46.6% of the other dice coefficients meaning that, on average, around 46% of the true matching pairs will be classified as non-matching. A value of 0.78 is hence too high. A small decrease of the value to 0.75 will make it larger than only 20% of the other dice coefficients, so around 80% of the true matching pairs will be classified as true matches.

A dice coefficient threshold of 0.75 will hence be our criteria for accepting a name as true match.

3.3.2 The exponential ElGamal homomorphic cryptosystem

This section outlines the foundation that we build our protocols upon. It is the fundamental building block that offers the underlying security of the data encryption. In the following, when using ElGamal, we are referring to the *exponential* variant of the algorithm [31] which we will briefly review in the following.

Key generation

Let $\mathbb{G}_q$be a multiplicative cyclic group (meaning if we multiply two elements from this group, the result also belongs to the group) of prime order q. Without loss of generality, let $\mathbb{G}_q$be a subgroup of $\mathbb{Z}_p$ (all the positive integers modulo p), in which $p = 2\alpha q + 1$ for $a \geq 1$ (a is usually equal to 1 unless a person is a security expert, then they can choose the value carefully based on the domain). The US National Institute of Standards and Technology minimally recommends $|p|$ =2048 bits and $|q| = 224$ bits, achieving a 112-bit security level [120]. Let g be a generator/primitive element of $\mathbb{G}_q$.

The key holder selects a random $x \in_R \mathbb{Z}_q$ and computes

$$y = g^x \bmod p.$$

The key holder's private key is $\langle x \rangle$ and the public key is $\langle p, q, g, y \rangle$.

Encryption

Let $m \in \mathbb{Z}_q$ be the message to be encrypted.

The sender selects a random $r \in_R \mathbb{Z}_q$ and computes

$$\langle c_1, c_2 \rangle = \langle g^r \bmod p,\ g^m y^r \bmod p \rangle.$$

Decryption

Given $\langle c_1, c_2 \rangle$ the key holder first computes

$$m' = g^m = \frac{c_2}{c_1^x} \bmod p.$$

The key holder then computes

$$m = \log_g m'.$$

This step is equivalent to solving a discrete logarithm, which is a hard problem in general, but is efficient when the message space $|\mathcal{M}|$ is sufficiently small. For our application, this condition is satisfied: we will be comparing bigrams so our space is small (in the HCN In scheme defined in this chapter, there are 260 potential bigrams) and the logarithm can be resolved using a lookup table.

Operations

In homomorphic cryptosystems, a specific algebraic operation can be applied on the plaintext by applying another algebraic operation on the ciphertext. In this cryptosystem, addition on the original data is mapped to the product of the corresponding ciphertext. Formally, for any two data values m_1 and m_2, and their encryption, $E(m_1)$ and $E(m_2)$, the following equation is satisfied:

$$D\big(E(m_1)\times E(m_2)\big) = m_1 + m_2 \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ (1)$$

where D is the decryption function. Therefore, addition of the plaintexts is mapped to multiplication of the corresponding encryptions. The cryptosystem also allows a limited form of the product of an encrypted value:

$$D\left(E\left(m_1\right)^{m_2}\right) = m_1 \times m_2 \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

which allows an encrypted value to be multiplied with a plaintext value to obtain their product. For example, if we have $m_1 = 10$ and we take the encrypted value $E(10)$ and then raise it to the power of -1, as in equation (2), and decrypt the result we would get the plaintext value of -10.

ElGamal is known to be semantically secure, which means that different ciphertexts using the same key for the same message will not be the same. This property is important to ensure that an adversary would not be able to compare an encrypted message to a dictionary and would not be able to perform a frequency attack. These properties are utilized when we outline the steps of our protocol in section 3.4.2.

We assume that all communications among the parties in this protocol occur through a secure channel (e.g., SSL), and are authenticated and digitally signed. This prevents eavesdropping attacks from individuals spying on the communications channel which runs underneath the level of our protocol.

3.3.3 Optimizing the bigrams comparison

In order to decide if two lists of bigrams match or do not match, we calculate the value of the dice coefficient between those two lists and compare it against a threshold that was determined as most suitable based on experimentation in section 3.3.1.

Consider A as the first list of bigrams and B as the second list of bigrams. The lists contain unique bigrams, and |X| represents the number of bigrams in list X. The dice coefficient's formula is the following:

$$\mathrm{DC} = 2 * \frac{|A \cap B|}{|A| + |B|}$$

The database site knows the values of |A| and |B|. The key holder knows the value of |A| + |B| and they both know the threshold for DC.

To decide if the two lists match, we need to find out if the dice coefficient is above or under the threshold. Since we know |A| + |B|, and the threshold value for DC, we can calculate a threshold value for $|A \cap B|$. For a true match:

$$\text{Threshold (DC)} <= 2 * \frac{|A \cap B|}{|A|+|B|}$$

$$|A \cap B| >= \frac{\text{Threshold(DC)} * (|A|+|B|)}{2}$$

And hence $|A \cap B| \geq K$ (a constant).

Optimization at the database site

If |A| < K or |B| < K, it is impossible that $|A \cap B| \geq K$, so we do not even need to ask the key holder to perform any decryption.

Optimizations at the key holder site

The rule $|A \cap B| \geq K$ must always hold. If the remaining number of bigrams to compare cannot meet that threshold, we can stop processing decryptions because at best, if everything else matches, we still will not be able to reach the threshold.

3.3.4 Field reduction

Quantin et al. performed a quality assessment of an anonymous record linkage procedure [100] (which includes matching a record inside a dataset) where they highlighted the results of the linkage procedure as satisfactory given a sensitivity of 95% and a specificity of 100%. Their quality assessment method included the calculation of the positive predictive value (or precision) and the negative predictive value of the algorithm.

Christen and Goiser studied the quality and complexity measures for data linkage and deduplication [26]. They recommended that the number of true negative matches should not be used because the majority of record pairs will be classified as true negatives. The metrics that use the number of true negatives include accuracy, specificity and negative predictive value. The evaluation metrics that they recommended to use were sensitivity (also known as recall or true positive rate) and precision (also known as positive predictive value).

Weber et al. based their analysis of different record linkage methods and name matching techniques on sensitivity and specificity [134]. Baxter et al. consider accuracy as sensitivity and specificity in their comparison of fast blocking methods for record linkage [8].

In the following, "condition" refers to the condition of an individual participating in more than one clinical trial at a time (or within a specified time period). "Test" indicates the application of our protocol to determine if the condition is satisfied or not.

There are four variables used to calculate the desired metrics: the number of true positive matches, the number of false positive matches, the number of true negative matches, and the number of false negative matches. We explain in the following the meaning of each of these variables in our scenario:

- True positive (TP)
    - A true positive value indicates that an individual is participating in a concurrent trial and the validation protocol detected that he is. This parameter is important for phase 1 clinical trials in order to disqualify the individuals that are not eligible for the trial as soon as they are screened. Otherwise, clinical trial sites will incur additional costs and time delays to replace that individual if they discover he is not eligible at a later time or risk the integrity and the objectiveness of their study if they do not find out that he is not eligible at all. The metrics affected by this parameter are sensitivity and positive predictive value.
- False positive (FP)
    - A false positive value indicates that an individual is not participating in a concurrent trial but the validation protocol detected that he is. It is desirable to reduce the number of false positive matches because it is hard to recruit individuals for phase 1 clinical trials. However, the true positive parameter is more crucial than this one for this scenario. The metrics affected by this parameter are specificity and positive predictive value.

- True negative (TN)
    - A true negative value indicates that an individual is not participating in a concurrent trial and the validation protocol did not detect that he is. This parameter is similar to the "do nothing" option where the operation of the clinical trial is running smoothly. The metrics affected by this parameter are specificity and negative predictive value.
- False negative (FN)
    - A false negative value indicates that an individual is participating in a concurrent trial but the validation protocol did not detect that he is. This parameter is the extremely important one for phase 1 clinical trials because the entire purpose of the protocol is to detect such individuals. The metrics affected by this parameter are sensitivity and the negative predictive value.

Table 5 summarizes the importance of the parameters in the use case of phase 1 clinical trial eligibility.

Table 5 - Importance of parameters

Parameter	Degree of importance
True positive	Important
False positive	Desirable
True negative	None
False negative	Extremely important

The metrics that were measured during our experiments are the following:

- Sensitivity (= TP/(TP+FN))
    - Sensitivity is the most important metric in our evaluation. It measures the true positive percentage given that the condition is positive and some test outcomes are negative. In our scenario, a high sensitivity value ensures that almost all the participants who are enrolled in numerous clinical trials will be properly detected and disqualified.
- Precision or Positive Predictive Value (= TP/(TP+FP))

- Precision measures the percentage of true positives given that the test outcome is positive but sometimes the condition can be negative. In our scenario, a high precision value indicates that the protocol disqualifies very few individuals incorrectly.

- Specificity (= TN/(TN+FP))
  - Specificity measures the true negative percentage given that the condition is negative and some test outcomes are positive. In our scenario, a high specificity value indicates that almost all the individuals that are not participating in another trial concurrently are identified as such. This is not a very important measure for our scenario because the focus is on finding the individuals that are participating in other trials at the same time (or within a time period) and disqualifying them. We will ignore this metric in our future evaluations.
- Negative Predictive Value (NPV = TN/(TN+FN))
  - The negative predictive value measures the percentage of true negatives given that the test outcome is negative but sometimes the condition can be positive. In our scenario, a high NPV indicates that the protocol lets through the biggest number of eligible individuals.

In order to determine the importance of each metric, we use a score calculated using the following method: for each parameter in the metric's formula the method gets a number of points added to its score. Extremely important parameters are worth 3 points, important parameters are worth 2 points, and desirable parameters are worth 1 point.

Table 6 ranks the metrics by order of importance according to their score:

Table 6 - Metrics ranking

Metric name	Score
Sensitivity	5
Precision	3
Negative predictive value	3
Specificity	1

Specificity will be excluded from our analysis because it is not pertinent to our scenario.

Private record linkage protocols typically aim to match records using a wide range of fields including names, addresses, and other demographic information. Our initial datasets were composed of six fields, typically available in healthcare: first name, last name, gender, date of birth, health card number, and postal code. These fields were used to match participant identities.

In order to compare text fields, it is known that approximate comparison techniques perform better than exact matching because they allow for small errors that can be introduced through a typing mistake. The best approximate matching techniques rely on splitting a text into a series of consecutive characters and calculating a similarity index to decide if the entire text is classified as a match or not. The implication of this comparison scheme on private protocols is the added complexity of encrypting every subset of a text instead of the entirety of the text.

In our datasets, we calculate the computational time required to match one record against the dataset. Keep in mind that our datasets do not contain a full address, which is sometimes included and used for matching. This means that we are presenting a best case scenario. The text fields being matched are first name, last name, and postal code. In our datasets, including the 1990 US census, the average name length is 7 characters. The postal code is a fixed 6 alphanumeric values in Canada. Therefore the overall text fields produce a total of 23 bigrams to be encrypted on average (typically the fields are padded with a space at the beginning and at the end so a name with length x will be represented by x+1 bigrams. In our case, first name + last name + postal code produces 7+1+7+1+6+1=23 bigrams). The maximum size of a dataset we considered for our clinical trial setup is 20000. If a protocol performed one to one matching, in a worst case scenario, it would have to perform 20000 comparisons. This means that we will have to match (8*8+8*8+7*7)*20000 = 177*20000=3540000 bigrams. We developed an optimized code implementing the exponential ElGamal cryptosystem and our experiments showed that on average, a decryption required around one millisecond. The worst case scenario to match a record in that case is roughly one hour on a machine with a single processor. This computation time is unacceptable to check the eligibility of a participant in a trial specially that the participant can be waiting for an answer on the phone or in person. To address this problem, a more powerful machine can be used, one that has multiple processors. Typically, a well-built machine today would have a processor with multiple cores. Record linkage applications are known to be easily parallelizable. Data centers on the other hand have much more powerful machines but they

also cost a lot of money. To do our part in reducing computational cost, we tried to minimize the number of bigrams that need to be compared by reducing the number of fields required to identify a participant.

Text fields are a bottleneck for privacy-preserving protocols that aim to provide higher matching accuracy. In the following, we calculate the values of the sensitivity (Figure 5), negative predictive value (Figure 6), and precision or positive predictive value (Figure 7) for the eight different text field combinations for dataset sizes of 1000, 5,000, 10,000, and 20,000 participants in a clinical trial.

The displayed values are the averaged results on 50 different sub-datasets of a same size. The legend indicates the selected fields with each letter representing a field that was included in our protocol. 'F' stands for first name, 'L' for last name, 'G' for gender, 'D' for date of birth, 'P' for postal code, and 'H' for health card number. Combinations consisting of only one or two fields are not considered because they lead to very poor results. The datasets we used for this evaluation are based on the lists of doctors from the College of Physicians and Surgeons of Ontario (CPSO) [117] and the list of lawyers from the Law Society of Upper Canada (LSUC) [118]. The results for both datasets are comparable.

We eliminated the GDH combination because it has lower sensitivity and negative predictive value than the other combinations, especially for larger datasets. We also eliminated the FGDH combination because it performed much worse (by at least 34%) in terms of precision than the other metrics for a smaller dataset size.

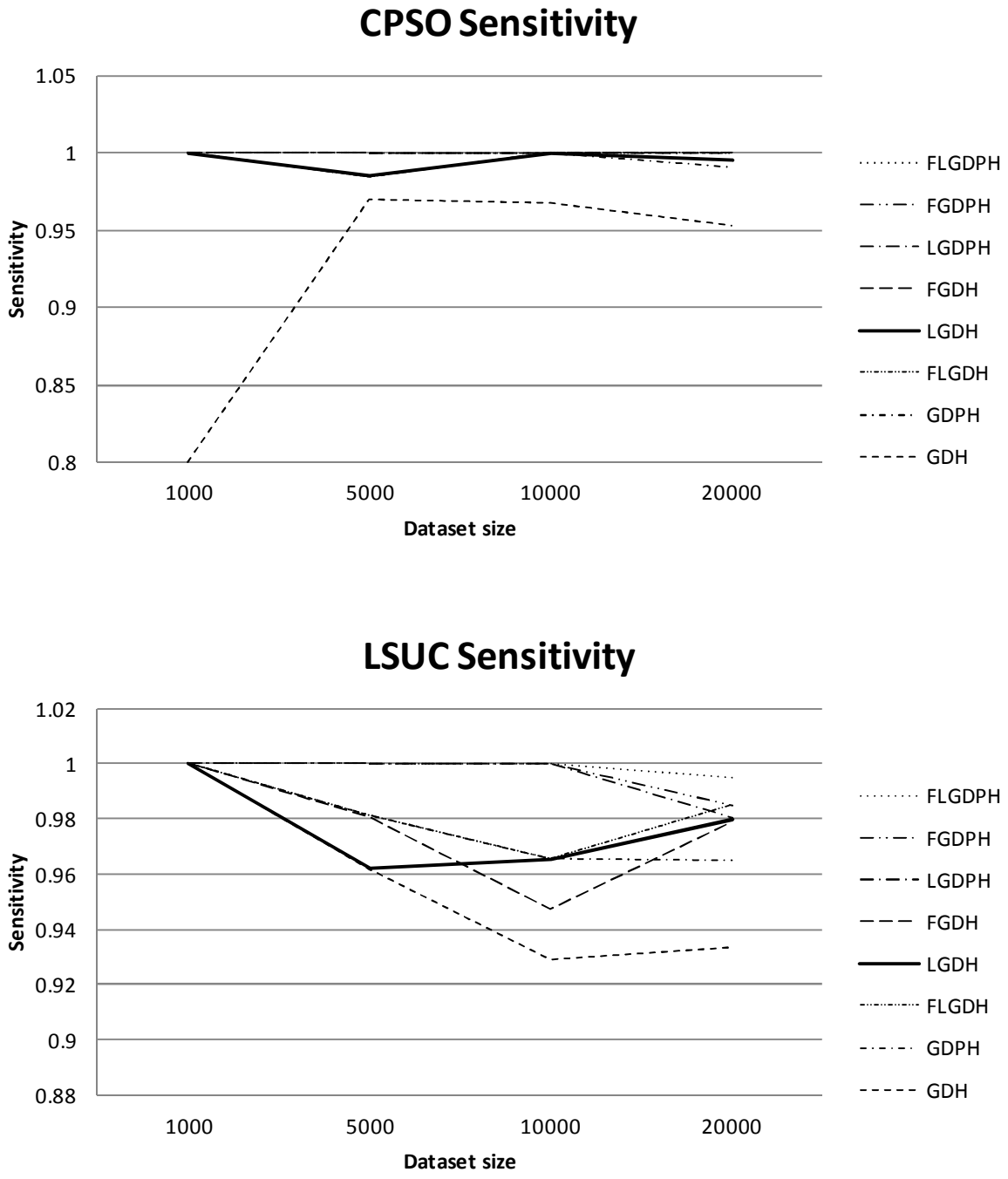
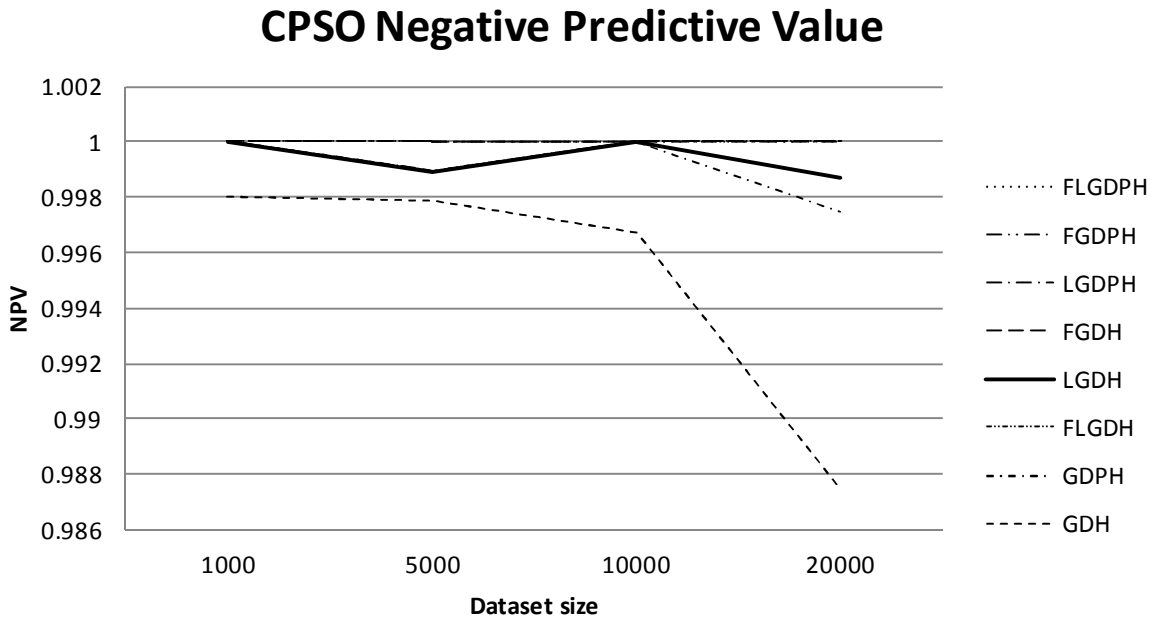


Figure 5 - Sensitivity for CPSO and LSUC for different string field combos



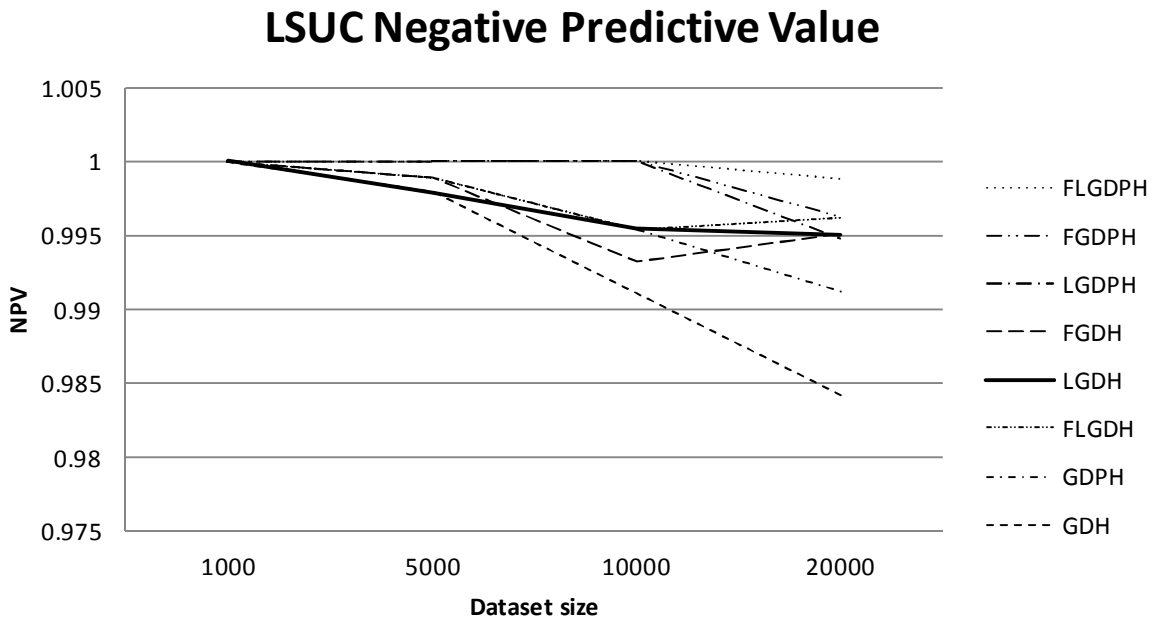


Figure 6 - Negative predictive value for CPSO and LSUC for different string field combos

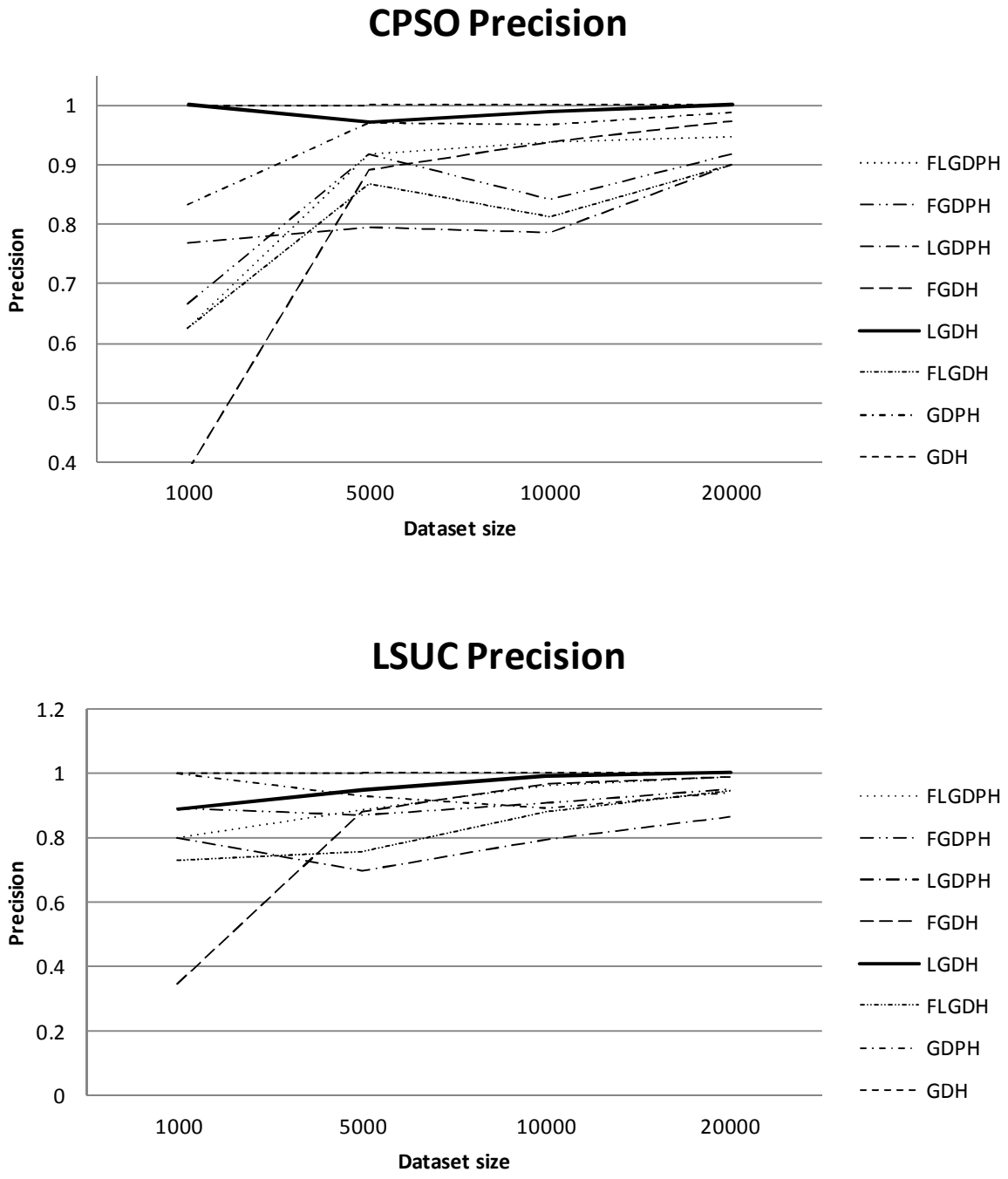


Figure 7 - Precision for CPSO and LSUC for different string field combos

The remaining combinations perform similarly for the three metrics except for precision, where the LGDH metric performs considerably better especially on smaller dataset sizes, which is typical for phase 1 clinical trials. The difference is meaningful: at least 17% between LGDH and the best one of the remaining metrics for the CPSO dataset.

Our goal was to reduce the number of text fields to the least amount possible with the least sacrifice of accuracy. The experiments above show us that the best field combination with the least amount of text fields and best accuracy is LGDH. We were hence successful in building new datasets that contained only four fields: last name, date of birth, gender, and health card number. The reduction of the text fields by a third allowed our experimental protocol to gain around 64% improvement in terms of performance. Instead of comparing 177 bigrams (first name + last name + postal code) for each record, we only compare 64 bigrams (last name). For larger datasets, the performance improvement is important because it translates into shorter wait times for participants to know whether they are eligible to participate in a trial. If a party had a machine with a 32-core configuration, they would be able to check the eligibility of a potential participant in roughly 40 seconds for a dataset size of 20,000 participants. The simple reliance on a health card number instead of the four fields can make the matching vulnerable to fake health card numbers. As stated in the 2006 annual report of the Ministry of health and long-term care [89], there were about 300,000 more health cards in Ontario than actual people. Most of these cards result from fraud (people outside Ontario who should not possess them, or possibly multiple cards per person). A match based on the four fields is efficient and also more robust in terms of proper identification of individuals.

3.3.5 Bigram selection

The graphs in Figure 8 and Figure 9 show the name length distribution in the North Carolina voters database and in parts of the Australian phone book (containing around 200,000 names). We notice that the rarest names are the ones towards the end of the curve, where for a name of length 19, there exist fewer than 10 individuals that can be identified without knowing the value of the name.

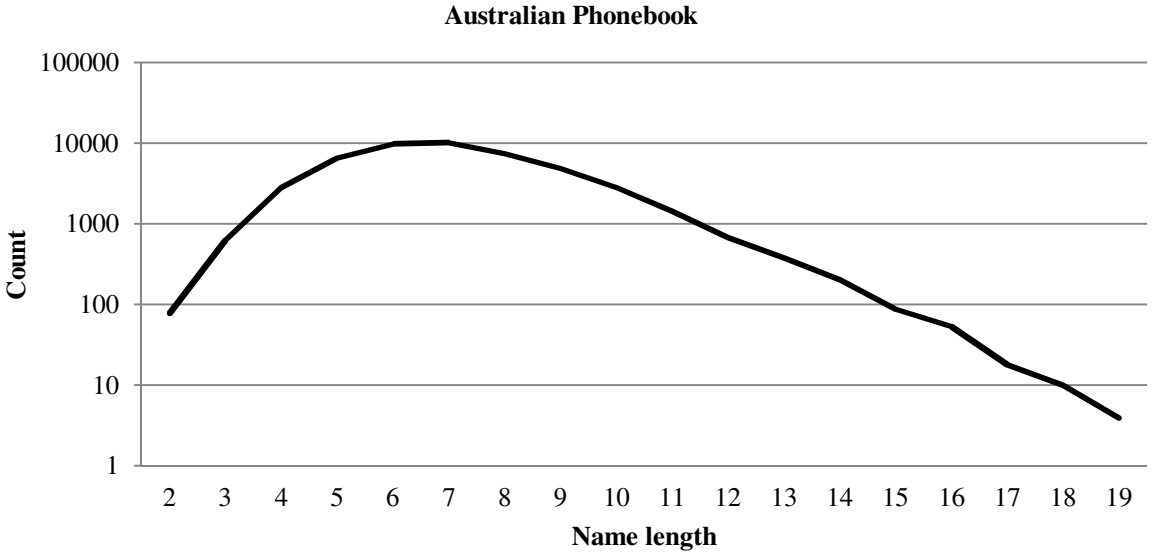


Figure 8 - Name length stats in the Australian phone book dataset

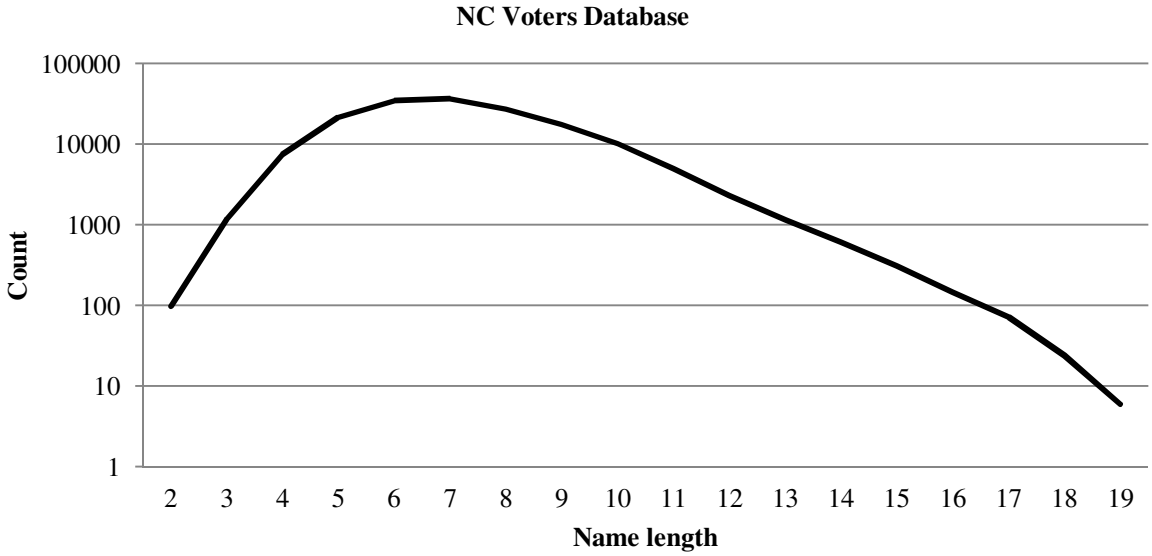


Figure 9 - Name length stats in the North Carolina dataset

We decided to limit the comparison of names using a maximum of 16 bigrams: beyond that limit, the number of names becomes fewer than 100 and poses a higher risk of identification. We conducted experiments to discover whether the 16 bigram limitation affected the accuracy of our protocol and to find out the best method of choosing these 16 bigrams. The five choices we considered for the latter part were the following: the first 16, the last 16, the most common 16,

the least common 16, and finally a random selection of 16 bigrams. The least and most common lists of bigrams were compiled from the actual list of names in the database, which provides the most accurate representation compared to compiling them from a list of names extracted from a phone book for example. Figure 10, Figure 11, and Figure 12 show that using 16 bigrams had little effect on the protocol's accuracy (as the results for the NC dataset and the Australian phone book were similar; we only show the Australian dataset in the graphs). We conclude that it is accurate to limit the number of bigrams that represent a name to only 16.

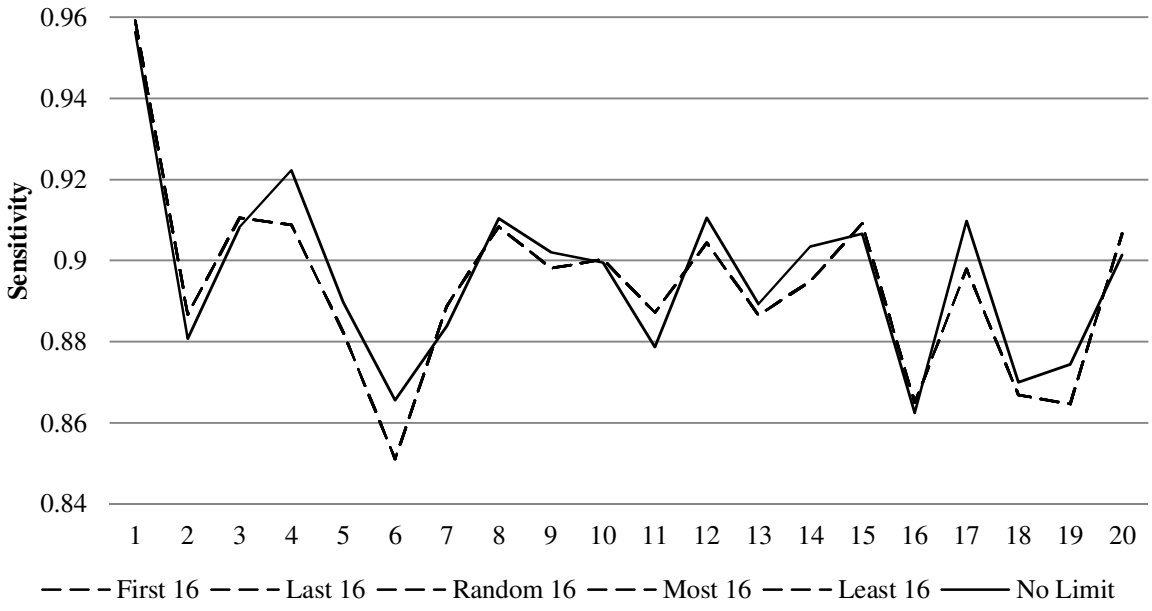


Figure 10 - Sensitivity for AustPB databases for different bigram selection techniques

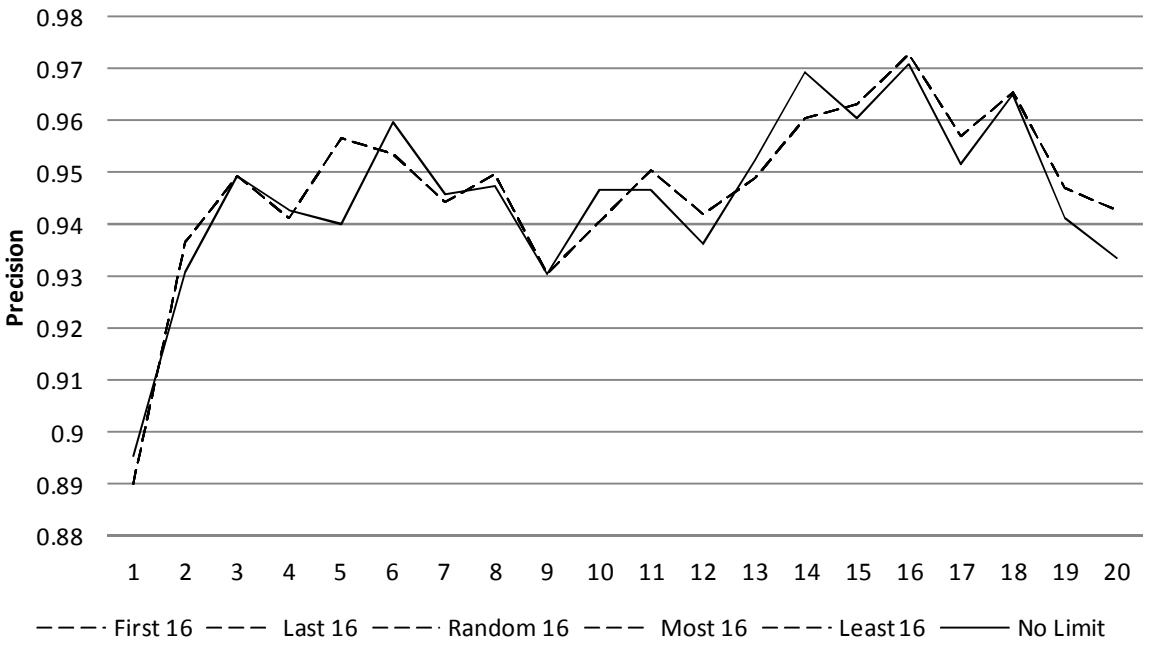


Figure 11 - Precision for AustPB databases for different bigram selection techniques

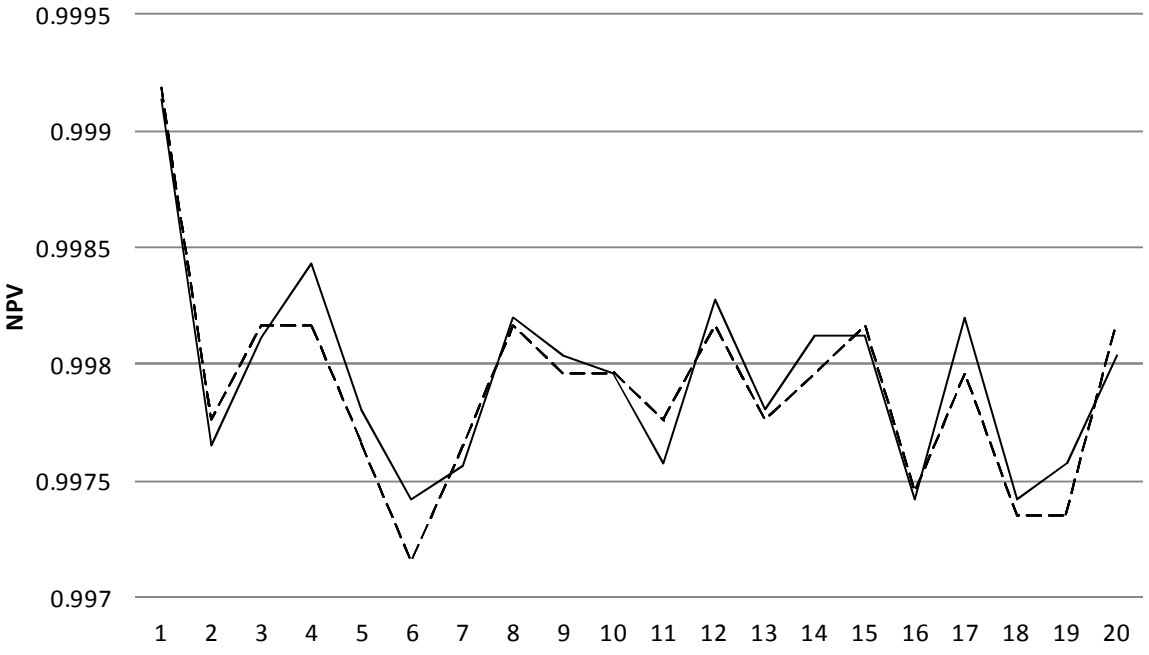


Figure 12 - Negative predictive value for AustPB databases for different bigram selection techniques

3.3.6 Name representation

Even with the best encryption and guards on the value of a field, we show other techniques in the following to discover the values of that field in a dataset. For an individual record on its own, it is enough to have a strong encryption and limit the length of a field in order to protect its privacy. However, when we are protecting the privacy of a dataset, more factors come into play. Certain fields, for example the last name, do not have the same distribution amongst a population. In the North Carolina voters database, the last name Smith is the most common (1.3%), followed by Williams (0.99%), Johnson (0.89%), Jones (0.88%), Brown (0.74%), and Davis (0.69%). In parts of the Australian phone book, Smith is also the most common last name (0.89%), followed by Williams (0.47%), Jones (0.46%), Brown (0.43%), Taylor (0.39%), and Wilson (0.37%).

If the encryption used to protect the data set did not offer properties to generate multiple encryptions for the same text, it would be very easy for a person looking at the encrypted dataset to immediately notice which fields represent the name Smith because they will be the ones that have the most common encrypted values. We performed several experiments to generate a random data set from the North Carolina database and parts of the Australian phone book. As soon as the data set size passed the 300 person limit, the name Smith immediately started to show the highest frequency in the small random dataset. As we increased the size of the generated dataset, the frequency difference for the most common names in the original dataset was more apparent.

Privacy at the storage level is also not enough to protect a dataset. If a dataset is encrypted with an encryption scheme that is able to represent the same fields in different encrypted forms, this means that the encrypted data is protected while it is stored. However, the risk is re-introduced at a later step in the process. The entity performing the decryption knows the status of the match or non-match for a field, even if the cryptosystem did not reveal the original values, the party performing the decryption has to know if there was a match or not. Therefore, there is a risk introduced for the last name, since the frequency of the matching is known, the party performing the decryption can assume a certain value based on the frequency of the matching correlated to the frequency of the name in the original population (with access to that demographic information).

We developed and tested a name representation scheme (HCN In) that offers better privacy and performance in terms of accuracy compared to the popular consecutive bigrams scheme [113] (SMITH would be represented as *_S*, *SM*, *MI*, *IT*, *TH*, *H_*). To represent SMITH with health card number 123456789, our new scheme interleaves one character from SMITH with one digit from the health card number. If the health card number is not long enough, we would restart from the beginning. The representation for SMITH would be: *S1*, *M2*, *I3*, *T4*, *H5*.

This scheme accounts for minor spelling mistakes (one mistake for names between 4 and 7 characters long, two mistakes for names between 8 and 12 characters long, three if more than 12 characters, and so on). For example, SMYTH would be represented by the same bigrams except *I3* which would be replaced by *Y3*. The dice coefficient approach to match names has a defined threshold that can account for these mistakes and still declare the two names as a match in most cases. We recommend using the exponential ElGamal cryptosystem, which is fast and provides semantic security. It also allows the use of a random number in each encryption, which will not affect the result of the decryption. Using a different random number each time, our scheme will protect against frequency attacks at the storage level and interleaving the health card numbers with the characters of the last name will allow approximate matching while protecting against frequency attacks at the decryption stage.

The graphs in Figure 13, Figure 14, and Figure 15 show the average results over 50 independent datasets based on the CPSO and LSUC resources. They compare the regular technique of splitting a string into consecutive bigrams to our new *Health Card Number Interleaving* (HCN In) technique that creates bigrams by interleaving characters from the name with digits from the health card number [45]. We notice in particular that the values of the precision are particularly better than the regular bigrams scheme, meaning that protocols using this new scheme will disqualify much less individuals incorrectly and allow contract research organizations to recruit a bigger number of participants as they are striving for.

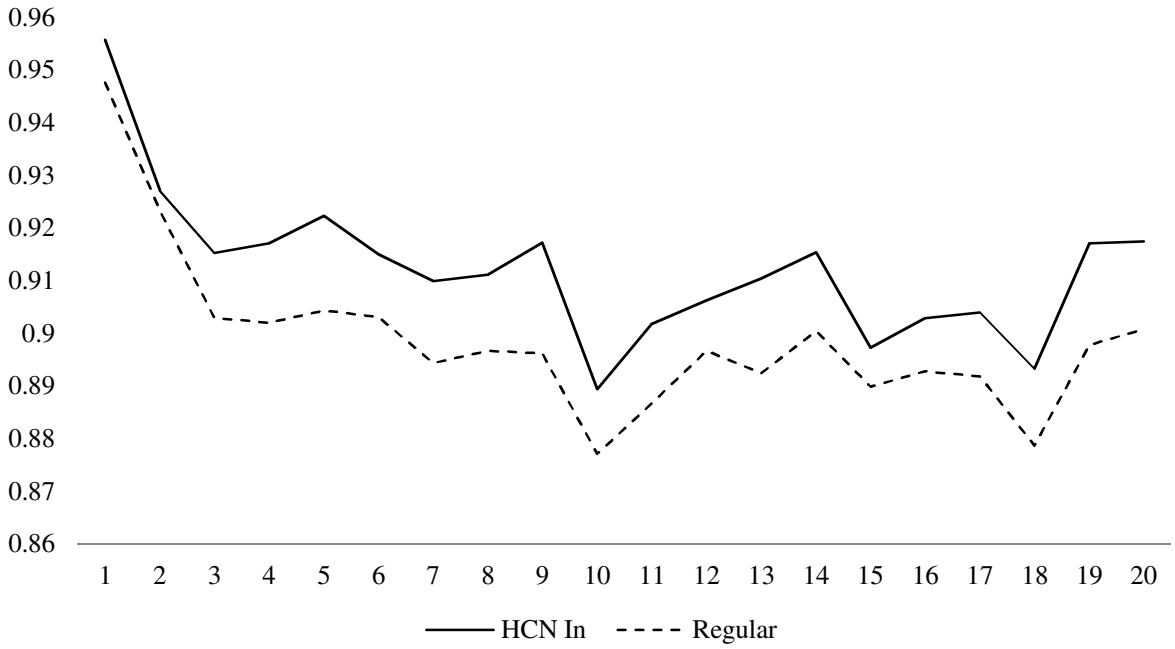


Figure 13 - Comparing the sensitivity of the regular bigrams scheme against our new scheme

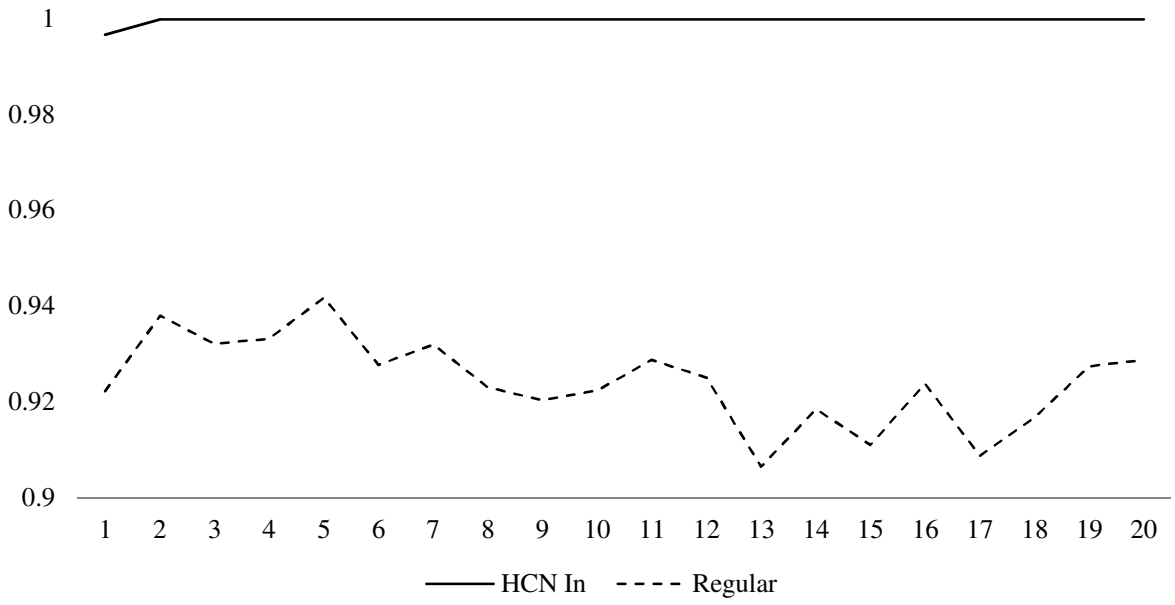


Figure 14 - Comparing the precision of the regular bigrams scheme against our new scheme

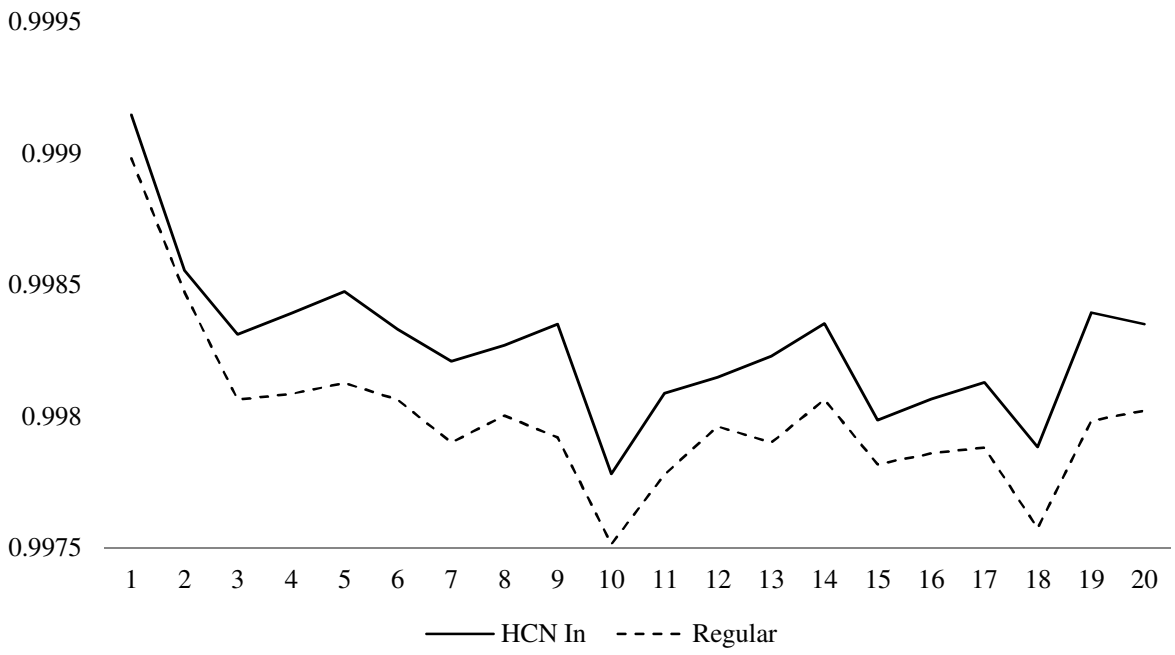


Figure 15 - Comparing the NPV of the regular bigrams scheme against our new scheme

3.4. TRACK protocol

Our TRACK protocol allows the tracking of participants to ensure that they do not enroll in multiple concurrent trials. For the purpose of our description, we assume that a concurrent trial is one where the washout period (required waiting time after completing a clinical trial for the drug to exit the system properly) has not yet been completed. The protocol can be easily extended to incorporate other criteria, such as the volume of blood drawn during the trial, without loss of generality. Our protocol supports multiple clinical trial sites that want to find out if an individual is concurrently enrolled at another trial. Several clinical trial sites become part of a group that agrees to enable all members to check if an individual has taken or is taking or will be taking part in a clinical trial, in a privacy-preserving way.

There are three roles in this protocol:

1. **Key Holder (KH):** This is a *semi*-trusted third party that generates the key pairs to be used in the cryptosystem and performs similarity computations. The semi-honest adversary model described in [53] can be applied to the KH.
2. **Central Database (CD):** The CD holds the encrypted data from all of the sites and responds to queries. The semi-honest adversary model described in [53] can also be applied to the CD.
3. **Sites:** The sites register their participants and execute queries to determine whether an individual is enrolled in a concurrent trial.

The KH role is not strictly necessary if the sites and the CD can communicate in the context of a traditional two-party protocol in which both parties perform computations on each other's encrypted data. In our setting, however, sites do not receive any encrypted data back from CD due to their limited computing capacity (section 3.1, requirement 7), making KH a necessary facilitator of the secure computation.

The protocol has three phases: *initiation*, *screening*, and *recruitment*. During the initiation phase, the keys of the cryptosystem are exchanged between the parties. Screening participants allows sites to query the CD. Recruitment allows the sites to add new individuals to the CD. Our protocol utilizes a homomorphic cryptosystem such as the one described in section 3.2.2.

3.4.1 Initiation phase

During the initiation phase, the KH generates a key pair in accordance with the settings specified in section 3.3.2 and sends the public key to the sites and the CD. Whenever a new site wishes to join the group, it is sent a copy of the public key. This phase is illustrated in Figure 16. The private key is kept confidential and is only used by the KH.

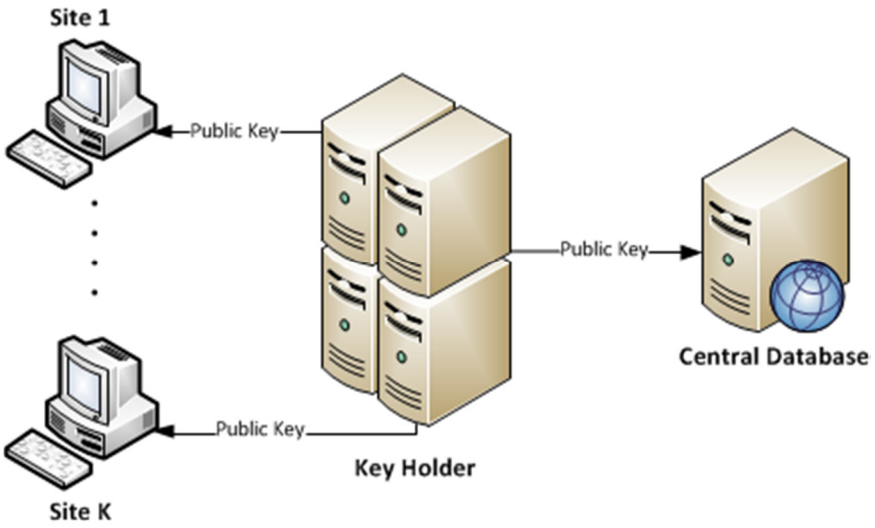


Figure 16 - Initiation phase where the KH sends out the public key to the different parties

3.4.2 Screening phase

During the screening phase, a site runs a query against the CD, as illustrated in Figure 17. The notation E(x) denotes an encrypted form of the identifier x. E(x) could represent multiple encryptions depending on the nature of the field (string, character, date, number).

Following the date matching guidelines used by the National Death Index [11], the following three matches are attempted for dates, and the dates are considered a match if any of them matches: (a) exact month and year of birth, (b) exact month and day of birth, or (c) exact month and +/- 1 year of birth. Therefore, the date of birth values are encrypted and sent as E(day), E(month), E(year -1), E(year), and E(year +1).

The last name is a string field and has to be considered differently to allow for approximate matching of strings. Therefore, we use an approximate matching approach. In this case, the string fields are converted into bigrams and these are encrypted individually. For example, if the last name has s bigrams, these are encrypted as $E(lastname_1)$, $E(lastname_2)$,…, $E(lastname_s)$. These are then transmitted, sorted lexicographically, to the CD.

The clinical trial study coordinator initially recruits individuals by telephone. During the call, the study coordinator collects basic information about the individual. The fields collected are: last name, gender, date of birth (DoB), and a unique identifier, such as the health card

number (known as the Ontario Health Insurance Plan (OHIP) number in Ontario) based on the experimentation performed in section 3.3.4. We do not rely solely on the unique identifier because it could be wrong or fake. Our technique will rely on a probabilistic matching technique capable of detecting a match even if one of the fields does not match.

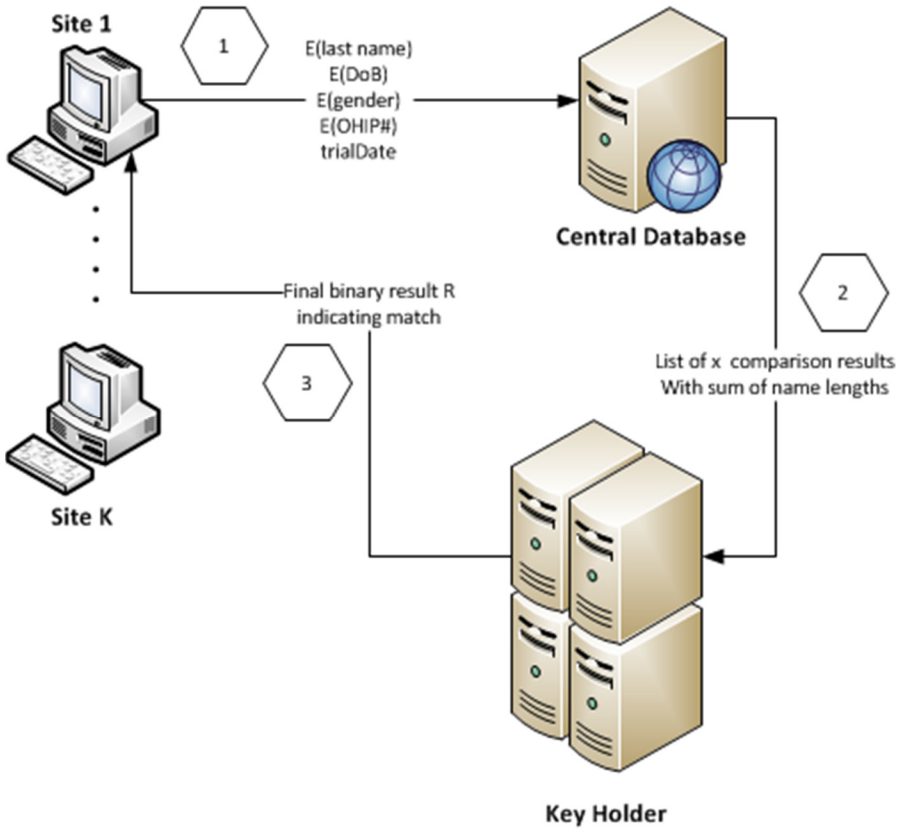


Figure 17 - Telephone screening phase where the site submits a query to the central database

After collecting the information over the telephone, the study coordinator enters the fields into our system. The system then encrypts the values using the ElGamal public key and sends these values as a query to the CD (annotation/hexagon 1 in Figure 17).

The CD performs a comparison of the values submitted by the site to the values stored in its database. The values stored in the database have been encrypted using the same public ElGamal key as the site. Let there be *N* records in the database. Out of these, let *N'* have a washout period that the date of the query is within; we assume the clinical trial starts at the date of the query but we can also supply a start date of the clinical trial instead of taking the query's

date as the default one which will allow us to accept participants whose washout dates from other trials are further in time then the date they are being checked at. If the values in the query match any of these *N'* records, then that would be considered a concurrent participant. The exact protocol for the matching depends on the type of field. To simplify the presentation, we will omit the "mod *n*" seen in section 3.3.2. We also assume that all communications among the parties in this protocol occur through a secure channel (e.g., SSL), and are authenticated and digitally signed.

For non-string fields

The non-string fields include the date of birth, gender, and health card number. We use a secure comparison protocol as follows [68]:

1. Let k index the record in the CD being matched with the values in the query, and let $E(c)$ be the value sent in the query and $E(d_k)$ be the value in the database.
2. Given the public modulus n, the CD generates a random number $r_k \in Z_n^*$
3. The CD computes $x_k = (E(c) \times E(d_k)^{-1})^{r_k} = E((c - d_k)^{r_k}) = E(\widehat{d_k})$. Note: $\widehat{d_k} = 0$ if and only if $c = d_k$. Otherwise $\widehat{d_k}$ is a random number in Z_n^*.
4. CD sorts the complete list of x_k values lexicographically, then sends them to the KH (annotation 2 in Figure 17).
5. The KH uses the private key to decrypt the x_k values. If any of these values is equal to zero, then the value in the query matches the value in the database, otherwise it is a non-match.

For string fields

We perform the following steps for matching:

1. All unique bigrams for the volunteer's last name are generated using a character from the last name and a digit from the health card number. If the health card number digits are not enough to cover the length of the name, we restart from the first digit after reaching the last one. (The documentation of the Febrl tool [21] has reviews of the types of real-world errors that happen in names and shows that the majority of these errors are single character substitutions. This scheme performs well in handling these errors and other types as shown in the empirical evaluation in section 3.9.3).

a. Let the number of bigrams in the query be *s*. Following the experimentation in section 3.3.5, the maximum value for s is 16. Any additional bigrams are truncated. This constraint guards against analysis attacks on very long names that are rare.

2. All values of $E(c_i)$ where $i \in \{1,\ldots,s\}$ are sent to the CD (annotation 1 in Figure 17).
3. Let the number of unique bigrams in record j in the database for the same field be z. Following the experimentation in section 3.3.5, the maximum value for z is 16. The CD has $E(d_{k,j})$ where $k \in \{1,\ldots,z\}$. The CD applies the secure comparison protocol above to all pairs of bigrams in the query and record j. There will be $s \times z$ encrypted comparison results x_i, where $i \in \{1,\ldots,s \times z\}$.
4. CD adds a random number of fake bigram comparisons that do not match any real bigrams (for example two underscores compared to any two letters). This does not affect the real number of common bigrams but prevents the KH from knowing the original lengths of the individual names being compared.
5. These cipher texts are sent as a list to the KH, sorted lexicographically (annotation 2 in Figure 17). The KH will not know the individual name lengths values s and z to guard against a potential attack described in section 3.2.2. The CD sends the KH the sum of the lengths of both names being compared in order for the KH to compute the dice coefficient (DC, refer to section 3.2.1).
6. The KH decrypts all x_i values and computes the DC based on the number of bigrams that matched and the sum of the length of both names. If the DC is above a certain threshold (computed to be 0.75 in section 3.3.1) then this is considered a match on the last name.

The above protocols for exact matching and string matching result in a binary match/non-match value for all of the fields that are being sent in the query. The KH has these plaintext binary values. A probabilistic score can then be computed based on each comparison result using the *Fellegi-Sunter* (FS) model [47][57]. The parameters of this model were estimated using Jaro's *Estimation-Maximization* (EM) algorithm [64]. This model looks at all the values of the matches in every record and assigns weights to each field based on the number of matches in that field.

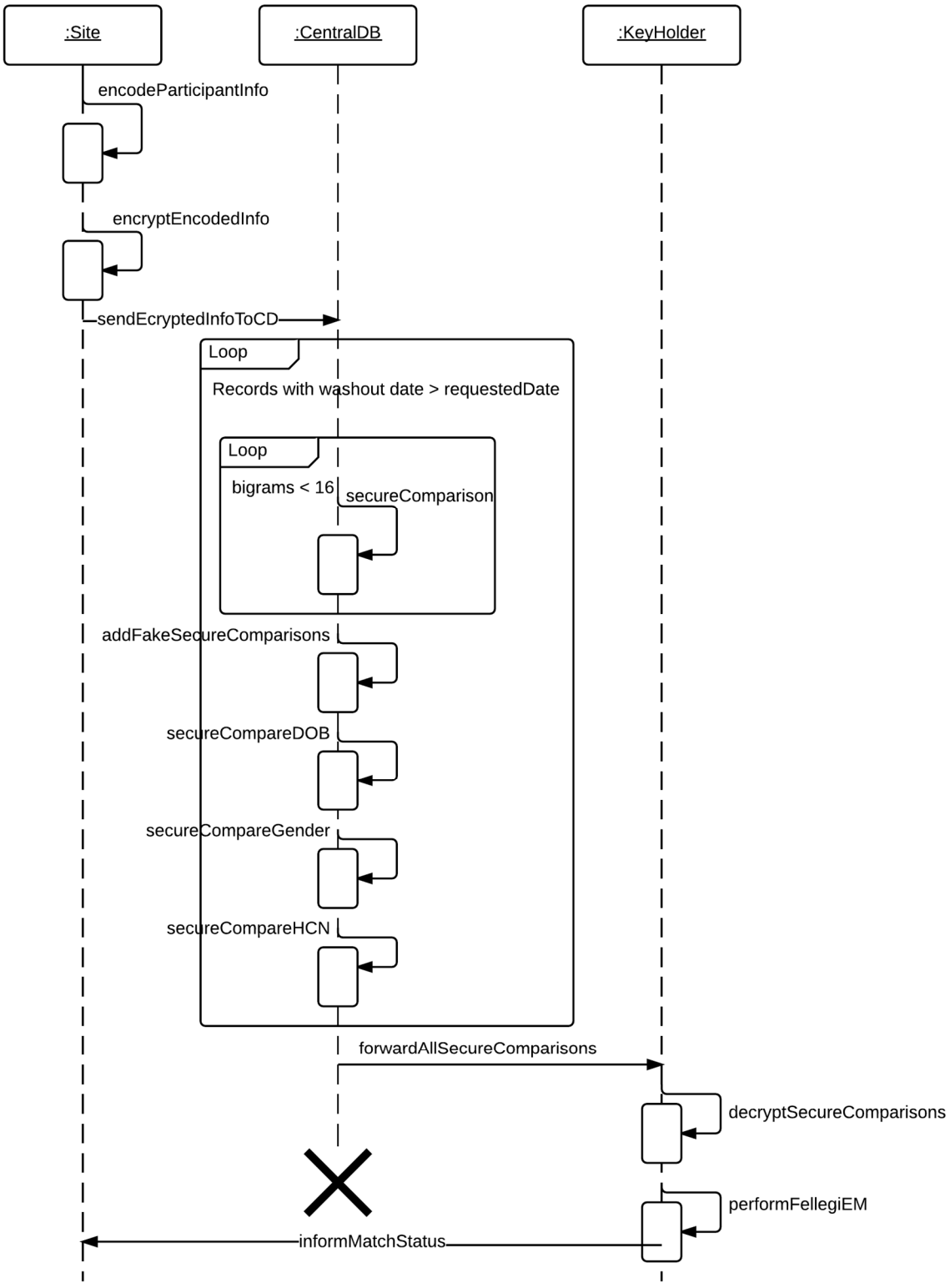


Figure 18 – Sequence diagram for TRACK, phase 2

For example, the gender field only has two possible values so there will be many records that match based on that field only; the model realizes this issue and assigns a lower weight to that field because it is not a strong differentiator. The health card number, on the other hand, is a field that should contain only one match, and hence the model assigns a stronger weight to that field because a match according to health card number should indicate a real record match. For each record, a score is computed based on the sum of the weights for each field where the field was classified as a match. If the score is higher than a cutoff value (we calculated this value following the same principle as the estimation of the DC cutoff value in section 3.3.1), then the query is considered a match, otherwise it is considered a non-match. This method presents an advantage in terms of probabilistic matching where a record is classified as a match even if some fields do not match. The KH sends the final match/non-match result to the site that initiated the query (annotation 3 in Figure 17). A summary of this phase is presented in the sequence diagram in Figure 18 on the previous page.

3.4.3 Recruitment phase

When a patient's recruitment eligibility is confirmed, he is invited on site before the beginning of the trial to present an official identification that matches the information he submitted over the phone. A final check for eligibility is performed where the participant is physically present at the site to prevent concurrency attacks: perhaps the phone verification can create an opportunity for an individual to contact several sites at once and take advantage of some lack of synchronization in the software to be accepted in multiple trials concurrently. Following the final verification, the site sends the participant's encrypted information to the CD for storage. The site must pre-compute all of the bigrams for the last name and the date of birth variants before encryption. The end of the washout period for that participant must also be sent, and this value is not encrypted because it does not represent any identity information and can be used efficiently to determine which records we need to match against. This phase of the protocol is illustrated in Figure 19.

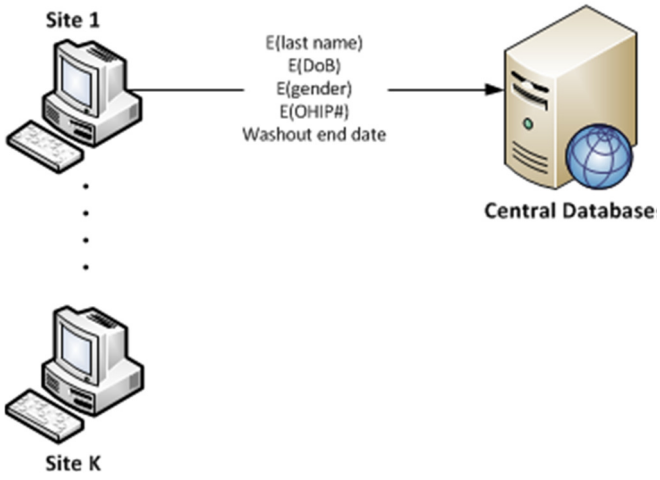


Figure 19 - Recruitment phase where the site submits the information to be stored in the CD

3.4.4 Requirements checkpoint

The literature review section outlined how the existing protocols lacked in terms of:

- Frequency attacks (section 2.3.1)
  - Breaks requirements *1)* and *4)* discussed in section 3.1
- Scalability issues (section 2.3.2)
  - Breaks requirement *7)*
- Secret sharing (section 2.3.3)
  - Breaks requirement *1)*, *2)*
- Record linkage quality (section 2.3.4)
  - Breaks requirement *6)*
- Lack of privacy (section 2.3.5)
  - Breaks requirement *3)*, *5)*

In this section, we explain how each requirement is satisfied according to the design of our protocol.

Requirement *1)* is satisfied by encrypting all of the participant's information in the query before submitting it to the central database. The CD does not know what it is comparing and does not know the comparison result (which is sent by the KH directly to the querying site).

Requirement *2)* is satisfied because the participants' identity information is always communicated and stored using the ElGamal public key encryption. The owner of the CD only has the public key (similar to each site) so he is not able of uncovering the identities of the participants. The CD contains a list of encrypted records, whenever a site sends a query, the CD only multiplies encrypted values without the possibility of knowing what they are. The KH receives a list of values that is the result of the multiplication of encryptions, so when the KH decrypts these values, the result is either zero or a random number. The KH only learns if there was a matching value or not, it does not know the original values of the records in the CD.

Requirements *3)* and *5)* are satisfied because the sites send a query to the CD and the KH replies to the site with a yes or no answer to its query for the site to know if the participant it queried for was present in the CD or not for a specific date.

Requirement *4)* is satisfied given the properties of the ElGamal cryptosystem: we are able to encrypt the same value multiple times and produce different encryptions for each one therefore making frequency attacks infeasible. We have also analyzed our protocol and provided guards against possible but not guaranteed forms of attacks.

Requirement *6)* is satisfied by allowing approximate matching using the dice coefficient method and splitting strings into a list of bigrams and the regular ElGamal cryptosystem for exact matching on numbers.

Requirement *7)* is satisfied by delegating the heavy decryption work to the KH. A site only has to perform an encryption on one record that it wants to check or add to the CD. The bulk of the work is performed by the KH while decrypting the results of the comparisons from the CD to answer the site if a match was found or not.

Finally, requirement *8)* is satisfied by the central database itself, which holds the information from multiple sites and therefore allows any site to compare against the information submitted from all collaborating sites.

3.4.5 Security analysis

We adopted the semi-honest adversary model in the context of phase 1 clinical trials. The parties are expected to follow the protocol and could try to learn as much as they can while performing their computations. We discuss in the following what each party is capable of learning when following our protocol. Section 3.4.6 will then discuss malicious behavior that does not conform to the semi-honest adversary model and under what circumstances the privacy can be breached.

By definition, the recruitment sites only learn whether the individual they are trying to recruit is concurrently enrolled in another trial or not.

The central database has the public key and the encrypted values from each site. Since exponential ElGamal is semantically secure, it cannot run statistical attacks on the collection of values it has. We also limit the number of maximum bigrams to 16 in order to guard against attacks for long names that are rare. The central database knows when a site has recruited an individual because the site asks the CD to include that individual in its database but does not know the identity of that individual. The CD also knows the number of individuals recruited by each site.

The key holder has the private key and decrypts any information sent to it using the public key. However, the single encryptions are never sent to the KH. The CD always performs a comparison by multiplying the values in the query to the values in its database and raising the result to a power of *r*, a random number. The decryption is either another random number or zero if the values match. Hence, the KH does not have access to record values; it only knows the sum of the lengths of two names and the number of common bigrams in order to compute the dice coefficient. The limitation to a maximum of 16 bigrams and the non-disclosure of individual name lengths guard against the potential length/common bigrams analysis attack (if the right resources where available) described in section 3.2.2. Our *HCN In* bigram generation strategy prevents the KH from doing statistical analysis on the last name field: since we combine the digits from the health card number with the letters of the last name, two last names that are the same will not match in their encrypted form if the health card number is different. Note though, if the health card number has a small mistake, the match will still return true since we are using the dice coefficient approach with a threshold.

3.4.6 Malicious behavior analysis

Our protocol was designed for the semi-honest adversary model. Nevertheless, we present an analysis of malicious behaviors some parties may attempt.

The recruitment sites only learn from the protocol if the individual they are trying to recruit is concurrently enrolled in another trial or not. To check for that individual, they need his last name, date of birth, gender, and health card number. Arguably, the last three pieces of information are not publicly accessible but if a site already recruited an individual for a clinical trial in the past and kept his information, they can check if that individual is participating in a trial at a future date in time. To protect against such scenarios, an individual may want to check the consent form he is signing to make sure that his information will solely be used for the clinical trial he is applying for and will not be kept for future use.

The central database does not know the identity of the individuals in the database. Even if the CD was able to discover the identity of one individual in the database (perhaps by sending an individual to spy at a clinical trial site and monitoring the date and time an individual walked in the site and the date and time the site requested to add that individual to the CD), the other individuals with the same name or date of birth are still protected because ElGamal is semantically secure and offers a different encryption value for the same plain text.

The danger that causes full disclosure of individual identities lies in the collusion between the central database and the key holder. If these two parties were to become malicious (contrary to the assumption in the semi-honest model), the CD can give the KH an encrypted record and the KH can decrypted using the private key and discover the identity of the individual.

Collusions between individual clinical trial sites does not break the privacy of the protocol as the sites only have access to the public key and only learn whether an individual is taking part in another trial or not.

Collusion between a site and the KH can lead to the discovery of the individual field values in the CD that match the values in the query. The KH and the colluding site would be able

to construct partial records that have information equal to the ones in the query. This would not conform however to the semi-honest adversary model adopted for our use case.

Collusion between a site and the CD can lead to the discovery of which participants belonging to the site's database are taking place at a trial at a different site. However, it cannot identify any other individual in the CD. This risk can be mitigated as discussed earlier by the participant signing a consent form authorizing the site to use his information only for the clinical trial he requested and for his information to be deleted afterwards.

Given the outlined risks, it is important to choose a key holder and a central database operator that will follow the protocol properly, which offers strong privacy for the participants. However, derailing from the protocol does present some risk of participant identification especially between the KH and the CD. The KH and the CD need to be chosen as entities that are professional and known to respect privacy.

To enhance the privacy measures of the protocol, we also introduce the notion of fake queries. Any site can send the central database a query for a fake individual for whom it does not care about the result. The benefit is that this would break the assumption that each query contains the identity of a participant for a trial. However, if the sites send too many queries, this could diminish the performance of the system. Since it is the key holder that has the potential of learning the most from the queries, we could delegate the task of the fake queries to the CD. The CD, handles all the query forwarding to the KH and so the CD knows the computing capability of the KH and can determine at any time if the submission of fake queries will delay the processing of normal queries or not. The CD can act as a load balancer and inject fake queries to the KH therefore the KH can no longer assume that each query has valid content and makes the task of trying to analyze queries for personal information harder. To mitigate the risk of collusion between the CD and KH, individual sites could also choose to send fake queries at random times to misinform the KH and to guard against the disclosure of their recruitment frequency.

3.5. Data sets generation

A main challenge for clinical protocol evaluations is the lack of public real-world data sets due to the private nature of patient information. Researchers require a collection of data sets of various

sizes to evaluate, verify and validate their protocols. Some of the data sets are used to calculate the appropriate threshold values of certain variables while the others are used to verify the performance of the protocol given those thresholds. In the medical sector, these data sets consist of patient records with identity-revealing information. The information available in these data sets is highly sensitive in nature and therefore it is extremely hard for researchers to use the data sets of a medical organization in order to evaluate a protocol under development.

Researchers have resorted to synthetic data sets that model the population expected to run through their protocols. Existing tools for generating synthetic data sets exist and are capable of introducing errors at rates seen in real-world data sets [99]. These tools make it feasible to test protocols under development to make them ready for their real-world usage.

Techniques for creating synthetic data sets based on prior knowledge of medical databases have been developed [18] and offer solutions for creating entire medical records and patient visits simulation. However, our focus is on zero knowledge of existing medical databases and the evaluation of medical protocols at their early stage, when collaboration with institutes is extremely difficult because of a lack of proof of concept and viability of the protocol. The data sets that we want to help researchers create aim to fill that gap at the early development of protocols in order to help achieve a proof of concept and move testing further with more sophisticated tooling. Record linkage protocols are the main target of our data set generation because they are based on patient identity. Record linkage of electronic medical records is very important to help hospitals complete their patient files and enable doctors to make better treatment decisions, especially with the current shift from institution-centered care to consumer-centered care [108].

We experimented with data set generation to evaluate our protocol for phase 1 clinical trials. In the following subsections, we describe the parameters of real-world data sets; it is very important that synthetic data sets match the parameters of their real-world counterparts in order for the validation of the protocol to be performed with confidence and integrity. We also provide an overview of the current tools and their benefits and limitations.

3.5.1 Real-World data set parameters

Researchers refer to publicly available data sets as sources for the creation of synthetic data to test their protocols. In order to avoid bias, researchers require using multiple data sets: the data set used to calibrate the parameters of their methods needs to be different from the data set on which the methods are tested. The various types of publicly available data sets include: phone books (Australian), voters lists (North Carolina, USA), professional members lists (doctors and lawyers in Ontario, Canada), public occupancy records [116] (states of California, Texas, and Florida, USA), online resources (Data.gov), and census data (USA, Ireland).

Researchers should be careful in the selection of their data set depending on the type of protocol and application they are trying to evaluate. For example, a concern for the evaluation of clinical trial protocols is the social status of the participants. A list of lawyer names might differ from the general population.

The following parameters describe some of the variations that need to be accounted for when simulating a real-world data set. The number of parameters that are accounted for may depend on the type of protocol being evaluated.

Size

This represents the number of records in a data set. For protocols handling clinical trials, the number of records may be in the range of thousands. However, for record linkage protocols, the number of records may go up to the range of millions. Researchers should also generate various data set sizes to test the performance of their protocol against the data set size.

Error types

This represents the types of errors usually found in real-world data set. For example, the Febrl tool [99] handles this parameter properly and is capable of producing data sets with error rates that match the ones found in real-world data sets including typographical, optical character recognition (OCR), and phonetic errors.

Content type

Each protocol will require specific fields to exist in the schema of a data set. Existing tools allow the generation of data sets with detailed names, addresses, bank account and credit card numbers. Certain fields have formatting restrictions that are hard to reproduce with data set generation

tools. For example, a social insurance number (SIN) in Canada has particular patterns, including a checksum. The random modification of one number could make the SIN invalid. The degree of importance of this parameter will depend on the protocol under evaluation. If the protocol compared a field for equality, it would be acceptable that a data set modification tool did not respect the format and restrictions of that field. The protocol that we evaluated only required identity information. However, specialized protocols may require the presence of specialized fields that cannot be generated automatically by existing tools unless the researcher has the option to supply the functionality that is capable of generating such fields (which will require programming skills) as an extension to the generation tool.

Prevalence

This parameter is important in the presence of more than one data set where it is important to have a percentage of commonality between the two sets in order to reproduce a real-world use case. For example, in the case of patient simulation, how many patients will exist in a data set compared to the patients that will not have any record?

Frequency

In order to simulate an existing population properly, it is important to capture the correct distribution of names and ages when generating identity information for synthetic data sets. A technique that we used is to capture the frequency of names in the original data set (for example, the 1990 US Census) and assign the same frequencies in the source file of unique names passed to a tool like Febrl [21], responsible for generating the synthetic data set.

Variability

Data set generation is seen to be a static operation most of the times: a researcher generates a data set and runs the protocol to capture the results. However, real-world scenarios have to handle the case of constantly evolving data sets, including address changes, last name changes, and the addition or removal of records. Most existing tools do not offer this functionality, i.e., the ability to selectively modify an already generated data set to account for small variability such as a name change. With the introduction of small variability to an existing data set that was tested against the protocol, a user could run several test iterations and evaluate the robustness of their protocol against these real-world situations.

3.6. Existing tools

In this section, we describe some of the known data set generation tools and their benefits and limitations based on testing (Febrl [21] and online tools) and a literature review. Febrl was the tool that we used to create our own data sets. We addressed some of its limitations by creating our own DataRealtor tool to bridge the gap between the generated data sets and our real-world parameter requirements.

3.6.1 The Febrl name modification tool

We used the *Febrl* dataset corrupter tool [21] to create populations for testing our protocols and methods. We discuss in the following some of the benefits and limitations of Febrl.

- Benefits
    - Introduces errors at the rate typically seen in real-world data sets.
    - Introduces various types of errors: insertions, deletions, substitutions, transpositions, Optical Character Recognition (OCR), phonetic, and more.
    - Allows the specification of how many errors to introduce per record and how many errors to introduce per field.
    - Allows the specification of how many duplicate records to generate for an original record and the total number of duplicates to generate.
    - Allows the specification of which fields to include for record generation.
        - For example, we can choose to create records with only a first name and a last name, or we could specify many more fields including an address.
- Limitations and improvements
    - Febrl does not allow us to specify an existing dataset to be corrupted. Instead, we supply it with data banks of first names and last names that it uses to generate both original and modified versions of a dataset.
    - For some fields such as gender, Febrl introduces random substitutions that consist of many different characters. For example, the gender could be ‘m’ or ‘f’, however Febrl can make them any other letter of the alphabet.

        - We wrote additional code so that a corrupted gender field will just have a 'u' for unknown. That step was important if we wanted to use blocking methods on gender.
    - Febrl creates a random number of health card numbers. It would have been closer to a real-world scenario if the health card numbers matched specific rules that are used in the real world.
        - This limitation does not affect the matching protocol because we check for equality of the health card numbers only.
    - The gender field in Febrl is randomly assigned. However, if we want a male gender to also have a male first name and a female gender to have an actual female name, we should split the first names into two files, one for males and one for females.
        - In order to generate a dataset with both real name and gender entries, a user can generate the male dataset separately from the female dataset and then merge the two datasets together.

We show some examples of the dataset modifications introduced by Febrl in the following:

- Insertion
    - The tool inserts one extra character in the spelling of the name.
    - Example of 1 insertion: "Nannie" becomes "Nannnie" (an extra 'n' was inserted)
- Substitution
    - The tool substitutes one character in the spelling of the name by other characters.
    - Example of 1 substitution: "Melissa" becomes "Melisaa" (the last 's' was substituted by an 'a').
- Deletion
    - The tool deletes one character in the spelling of the name.
    - Example of 1 deletion: "Flossie" becomes "Flosie" (one 's' was removed).

3.6.2 GeCo

GeCo [119] is the next generation of the Febrl tool, with a web interface to greatly facilitate the generation of data sets without the requirement of installing and configuring Febrl. However,

GeCo is limited to generating data sets of 9999 records or less in its online version. This is probably due to performance considerations and computing capacity of its hosting site. Still, it is helpful for researchers, who are also offered the option to download the tool on their own machine if required, and to use it as they need. The tool authors also offer researchers the ability to customize the tool to their needs by contributing lookup files and attribute generation functions.

3.6.3 Synthetic Occupancy Generator (SOG)

SOG [116] is a data set generator capable of producing occupancy lists. It generates personal information by preprocessing names using an industrial tool called DataFlux to separate last names, male first names, female first names, and names of unknown gender. The processed names become a source for generating records. This tool can also generate data for phone numbers and addresses. Identity information in SOG consists of a full name, a gender, a social security number, and a date of birth. The records in SOG could be modified based on marriage or death during a simulation.

However, SOG does not provide a data corrupter similar to Febrl, to simulate errors in real-world data sets.

3.6.4 Online tools

Data generation is offered by several websites that provide functionality to generate information either for a single individual or a data set of specific size. We highlight in particular http://www.generatedata.com/ for the clean interface it provides for generating a data set. It claims to be capable of generating country-specific data: for example, first names and last names would be tailored to the name frequencies in each country. However, we question that claim because the names generated for France and Italy were very close to the ones generated for the United States. These online tools allow one to specify some constraints for number and date ranges as well as gender-based name generation, phone numbers, and addresses. Some of these tools can be used to create source files for Febrl to reuse in order to generate data sets that also have corrupted versions that match the error rates found in real-world data sets.

3.6.5 Benerator

The Benerator tool described in [5] requires programming skills in order to generate reliable synthetic data sets. The authors showed how accounting for the frequency in the original data set can produce a synthetic data set that is similar in distribution to its original source (Irish census 2011). This tool, however, does not account for our other parameters, such as typographical errors.

3.6.6 Tools based on existing medical records

There are methods that use existing information from medical record databases [18] to create synthetic data sets that mimic the distribution of the symptoms while perturbing patient identity. These techniques are very rich in creating medical data sets. Another challenge of using these methods is to have an organization grant the researcher access to a real data set.

3.6.7 Summary

Table 7 summarizes the support of known tools (except the last category) according to our real-world parameters. We have classified the support in three categories: '✓' for well supported, '±' for partial support, and '×' for no support. The content column will widely be linked to the type of data set: if identity is the most important criterion then all the tools would get a '✓' for its support. However, some medical protocols require the generation of medical information that require specialized data generators.

Table 7 - Evaluation of known data set generators

	Size	Errors	Content type	Prevalence	Frequency	Variability
Febrl	✓	✓	±	×	✓	×
GeCo	✓	✓	±	×	✓	×
SOG	✓	×	±	×	×	✓
Online	✓	×	±	×	×	×
Benerator	✓	×	±	×	✓	×

While some tools like GeCo allow users to write their own rules for content generation, the content type evaluation takes into consideration users with no programming knowledge, and therefore requires the tool to offer formatting options in a user-friendly interface.

Febrl was our choice for creating the data sets for phase 1 clinical trials, topped with our DataRealtor tool that we developed to overcome some of Febrl's limitations.

3.7. Data sources

We generated data from the following sources:

- The 1990 US census database [124]
    - Containing a list of the most common:
        - male first names (1 219 unique entries)
        - female first names (4 275 unique entries)
        - last names (88 799 unique entries)
            - We eliminated the names with frequencies less than 0.001 because they would have been rarely used in the generation anyway. We were left with 18 839 last names.
- The College of Physicians and Surgeons of Ontario (CPSO) database [117]:
    - Containing a list of 38 295 full name entries:
        - male first names (4 620 unique entries)
        - female first names (2 837 unique entries)
        - last names (18 610 unique entries)
- The Law Society of Upper Canada (LSUC) database [118]:
    - Containing a list of 26 635 full name entries:
        - male first names (1 290 unique entries)
        - female first names (1 279 unique entries)
        - last names (10 554 unique entries)
- Parts of the Australian phone book (from the authors of [129])
    - Containing 172936 entries
    - We extracted 48175 unique last names and assigned their frequencies

- North Carolina voters database (obtained from EHIL)
    - Containing 6720722 entries
    - We extracted 166924 unique last names and assigned their frequencies

3.8. Data sets generation infrastructure and tools

Researchers require a collection of data sets of various sizes to evaluate, verify and validate their protocols. In the medical sector, researchers have resorted to synthetic data sets that model the population expected to run through their protocols. With a synthetic approach, researchers have to be careful in the bias introduced in the data sets. One method to reduce bias is to generate multiple versions of a data set and average the evaluation results calculated on the entirety of the data sets, which is a highly time consuming task. Moreover, as the data set size increases, the running time for an evaluation also increases.

We experimented with data set generation to evaluate a protocol for clinical trials. Our experiments highlighted the necessity of using a tool that can run evaluations on multiple computers and merge partial results into one final result to save time and have results generated faster in order to refine the protocol over several iterations.

Research in distributed systems is widely known and existing protocols are well proven to provide redundancy, task distribution, monitoring, and recovery from error. One of the systems widely used in industry is Google's MapReduce [32]. However, our goal is to provide a simple tool that can be used by the everyday researcher without the need to set up major infrastructures like MapReduce. Our infrastructure is based on everyday components, including a file synchronization system that allows several computers to share a collection of files automatically updated on all computers whenever an addition, a deletion or a change happens on any file (e.g., Dropbox[5][59]).

[5] Dropbox (dropbox.com) is a file synchronization system that allows several computers to share a collection of files that is automatically updated on all computers whenever an addition or a deletion or a change happens on any of them. The user has the option to select specific directories to be synchronized and the application will store the files on a remote file system in order to allow a machine on any network to access them. Synchronization is smart enough to determine if two hosts are on the same network, in which case the files will be accessed through the network instead of referring to the remote file system.

In the following, we highlight and explain our ProEvalNet tool characteristics and features so researchers can understand how it can benefit them in their projects. We also give a small description of how we put the tool to use while evaluating a newly developed clinical trial protocol.

3.8.1 ProEvalNet characteristics

We created an application called ProEvalNet (for Protocol Evaluation Network), which can be deployed on several computers to evaluate algorithms on multiple versions of a data set *concurrently*. Partial results are created at each location and merged by utility functions that we created to compile the final result.

Since operations on a large data set can take a long time, it is desirable to monitor their progress and send commands remotely. However, computer labs, especially those handling privacy, have firewalls that restrict connections from machines outside the local network. Using a Dropbox-based infrastructure (Figure 20) solves this problem because usually the firewall allows the local network machines to access Internet services including Dropbox. It becomes the communication link between a home computer or a mobile device and the machines on secure local network and enables ProEvalNet to drop a command for our system to run evaluations from anywhere and to monitor progress remotely, even through a smart phone. Sensitive data is not shared through Dropbox, only the command requests and progress updates are available. This type of infrastructure could be re-used by other researchers to overcome the fundamental problems we have encountered.

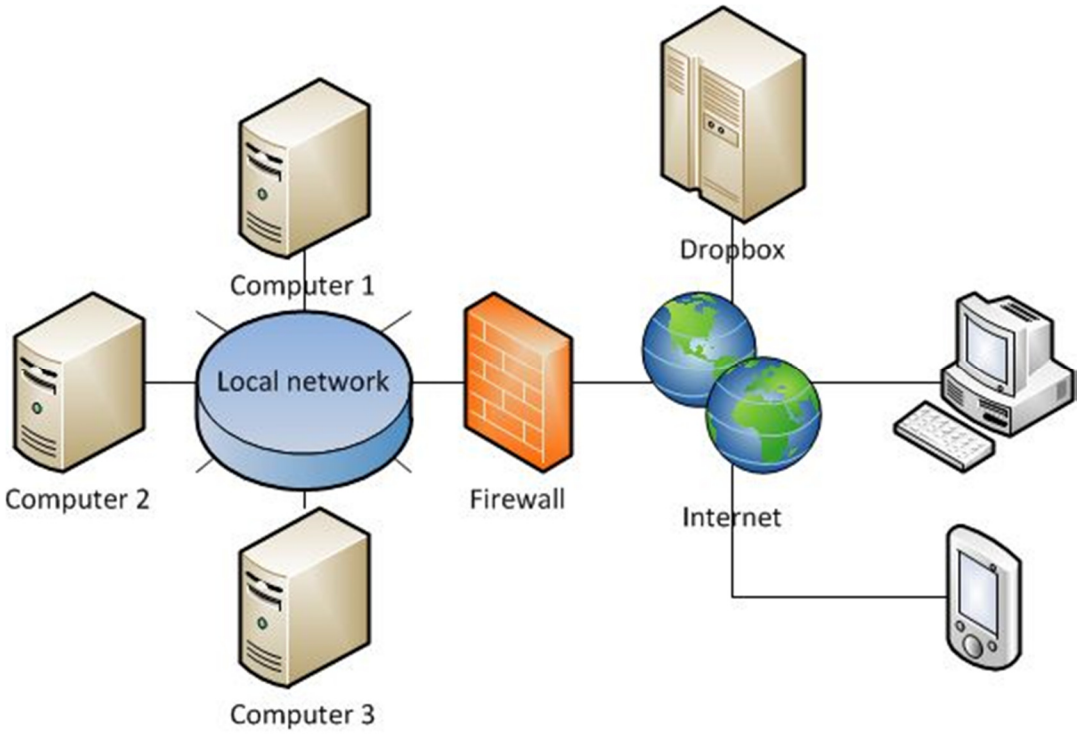


Figure 20 - Samples generation and monitoring infrastructure

3.8.2 ProEvalNet features

There are five main features in ProEvalNet, whose user interface is shown in Figure 21.

1. Field and data set selection criteria

For flexibility, we allow the choice of selected data sets from a large collection of data sets or even the selection of specific fields from these sets. Field selection is particularly useful to test performance and accuracy criteria of different algorithms.

2. Evaluation and comparison of multiple algorithms

This feature allows the side-by-side comparison of multiple algorithms on the same data sets, e.g., a newly developed algorithm and the gold standard in the field. The user can select which algorithms to include/ignore in the evaluation.

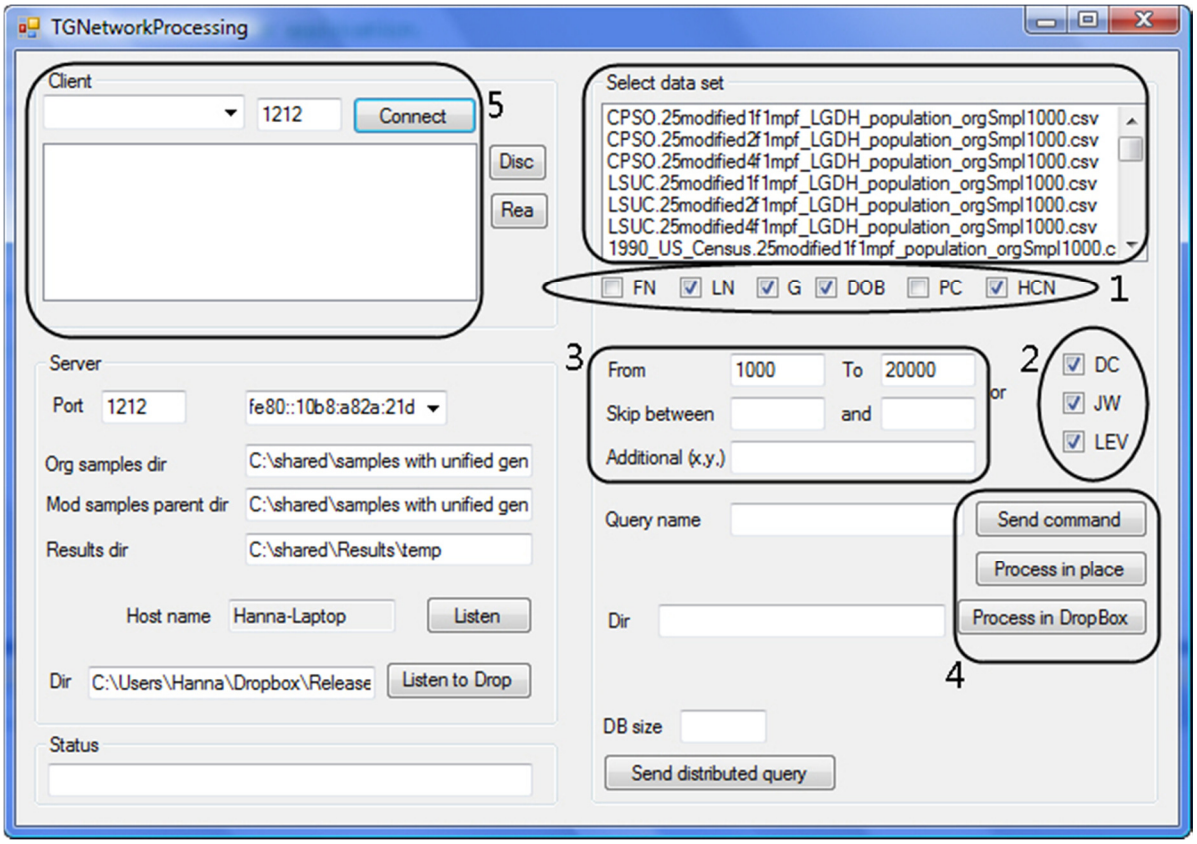


Figure 21 - ProEvalNet interface, with five areas discussed in this section

3. Flexible specification of data set intervals

This feature is key to the task distribution on multiple machines. Currently, the user specifies the interval to be executed manually but future versions will allow the usage of the 'Send distributed query' button to automatically distribute the work on the available machines. The specification of an interval is flexible, allowing the input of a straight interval, the skipping of a selected interval and a manual input of select data set sizes. This is particularly useful if a machine crashes and allows partial recovery instead of restarting the entire task.

4. Local and remote processing

The processing of commands can be done on the local computer or sent to a remote computer either through a local network or to a computer anywhere in the world if it is connected to a shared Dropbox directory. In the case of Dropbox, ProEvalNet will drop the command to be executed in the shared directory. The remote computer will read and execute it, and then delete

it. A results file is also made available in the directory to allow the user to monitor the task's progress.

5. Remote commands for local networks

This allows the operation of ProEvalNet on a local network without the need for additional Internet services if increased security was enforced by network administrators. It works by connecting to the network computers using their host names or IP addresses and sending them remote commands to execute.

3.8.3 Use case

To evaluate our phase 1 clinical trial protocol, we used Febrl to create populations (list of individuals) based on the 1990 US census, and doctors and lawyers lists of Ontario. We generated data sets ranging between 1000 and 20,000 individuals with increments of 1000 based on those populations to evaluate the scalability of our protocol. We decided to generate 50 different versions of these data sets and spread them over five computers in a secure lab environment. The Dropbox infrastructure was helpful to monitor evaluation progress and send commands remotely, with negligible communication delays. This flexibility allowed us to generate results nearly five times faster than on a single machine and shorten our protocol development iterations. Instead of waiting for 50 data sets to be evaluated on one machine, we had five machines handle ten data sets each working around the clock and easily recovered from crashes, especially the ones that occurred near task completion, by only repeating the intervals that were required. We also utilized ProEvalNet in our field reduction experiments in section 3.3.4.

3.9. Empirical evaluation

The purpose of the empirical evaluation of the TRACK protocol is to determine how the latter scales as the CD database size increases in terms of computation time, and what the accuracy of the probabilistic matching is. In order to perform the evaluation, we developed an optimized version of the protocol written in C#. For TRACK's exponential ElGamal homomorphic cryptosystem (section 3.3.2), a 1024-bit key was used for p and a 160-bit key for q, and each

processor core used for performance evaluation was running a Windows XP SP3 operating system at 2.8GHz.

3.9.1 Data sets

Phase 1 clinical trial sites enroll participants based on several criteria including the participation in a single clinical trial in a given time interval. To prevent concurrent enrolment, multiple clinical trial sites can agree to exchange data in a privacy-preserving manner in order to check if a participant is concurrently enrolled at another site at a given point in time.

Each clinical trial site can have a local data set of participants and a private central database might be required in order to have access to the participants from every site. In order to test and evaluate a protocol that satisfies the needs of clinical trial sites for checking participant eligibility, we require the generation of two data sets.

The first data set simulates individuals coming in a clinical trial site to participate in their trial. It is composed of names without errors in them as it represents individuals providing a piece of identification with their name visible on it. The second data set however represents all the individuals in a central database that participated in trials at all the collaborating sites. The second data set is allowed to have errors as data entry might have introduced these errors in the real-world environment. The prevalence value between the first and the second data set should be between 1% and 10% as this is the case in real-world clinical trials: less than 10% of individuals will be concurrent enrollers according to the references provided in Appendix A, which means that only 10% of records in the first data set should exist in the second data set.

We created our DataRealtor tool to support prevalence values between data sets. This feature ensures that two data sets have a particular number of records in common in order to simulate the real-world use case properly.

Using Febrl, we fixed the number of modifications per field to 1 as 95% of data errors are as such. There is also evidence that up to 25% of records have misspellings and typographical errors in names [51][136].

Febrl was used to create populations (lists of individuals) based on the 1990 US census, and doctors and lawyers lists of Ontario (available in Appendix C). Those data sets contained four

fields only: last name, date of birth, gender, and health card number. The choice of which fields to use is based on experimentation in section 3.3.4. Each population had an original version and a duplicate version with real-world errors. In order to simulate people participating in phase 1 clinical trials, we randomly selected 1000 people from the original population to act as participants in the phase 1 trials. To simulate a central database with errors, we randomly selected 20,000 people from the populations with 25% of them including errors in their fields (15,000 from the original population and 5,000 from the duplicate population). Our DataRealtor tool also created multiple versions of these data sets based on prevalence values between 1% and 10%.

The census data sets were used to calibrate the parameters of our protocol, which was later evaluated using the other two data sets. This step is highly recommended to avoid bias.

3.9.2 Evaluation process and metrics

We created our own DataRealtor and ProEvalNet tools [43][44] to generate and evaluate a range of datasets supporting the correct prevalence levels of dual enrollers according to real-world considerations outlined in Appendix A. The size of the database in the CD was varied from 1000 to 20,000. This is a reasonable number given that usually phase 1 trials recruit a small number of participants. The median enrollment for phase 1 clinical trials is 33. A database with 20,000 participants would amount to over 600 trials of that size that have completed recently within a limited geographic area. This is arguably a large number of trials and represents a large database size for our evaluation.

The CD database was sampled from one of the generated (perturbed) data sets. We sampled 1000 records from a second generated data set to simulate the queries. The accuracy and performance results were computed across the 1000 queries. We used a prevalence of 2% for participants that are in the CD, which is consistent with the prevalence range reported in the literature (as summarized in Appendix A). *Prevalence* represents the percentage of records in the CD that would match the queries. Note that, in general, higher values of prevalence tended to have higher accuracy, therefore the value we chose was on the conservative side.

The metrics we computed were sensitivity, precision (also known as positive predictive value) and the negative predictive value. In the following, let TP denote the number of true

matches, FP the number of false matches, TN the number of true non-matches, and FN the number of false non-matches.

To determine whether the secure string matching protocol had an impact on accuracy, we also compared these metrics with non-secure string-matching methods using the Levenshtein distance and the Jaro-Winkler score [57] using the exact same infrastructure. The whole evaluation was performed on 50 independently perturbed data sets (based on five distinct populations, from each of which we sampled 10 datasets) and the above metrics averaged across them.

Computational performance was measured in terms of seconds to execute the query. The query consisted of encryption at the site, matching with the database at the CD, and decryption at the KH. Other computations, such as the FS model with EM algorithm, consume negligible time compared to the cryptographic functions. Communication was not considered as this would vary depending on network traffic at the time the query was executed, and each query only requires a fixed sequence of 3 messages.

Total computation time at the CD and KH were measured for 1000 queries on CD databases of different sizes. For our simulations, we used 32 core and 64 core clusters (16/32 dual core machines) to represent the KH, and all of the computations were parallelized. Record matching problems are known to be easily parallelizable for improved performance [27].

3.9.3 Results

The performance results in terms of computation time as the database size increases are shown in Figure 22. For a 20,000 individual database, the total computation time is just under 20 seconds on a 64 cores machine for a query. Note that the protocol has linear complexity. This means that if we double the number of machines, we obtain an almost double increase in performance (taking into account minor overhead in machine communication time).

The accuracy results for the three metrics (sensitivity, precision, and negative predictive value) against database size are provided in Figure 23, Figure 24, and Figure 25, respectively. There are three key observations from these graphs: (1) there is little difference in accuracy between our secure computation protocol and one that does not use secure matching on the last

names, except for precision where we perform consistently better. A high precision value indicates that the protocol disqualifies very few individuals incorrectly and this is important because CROs have a hard time recruiting trial participants. (2) The accuracy on the three metrics is around 0.9 for almost all database sizes. (3) There is generally little variation in accuracy as the database size changes.

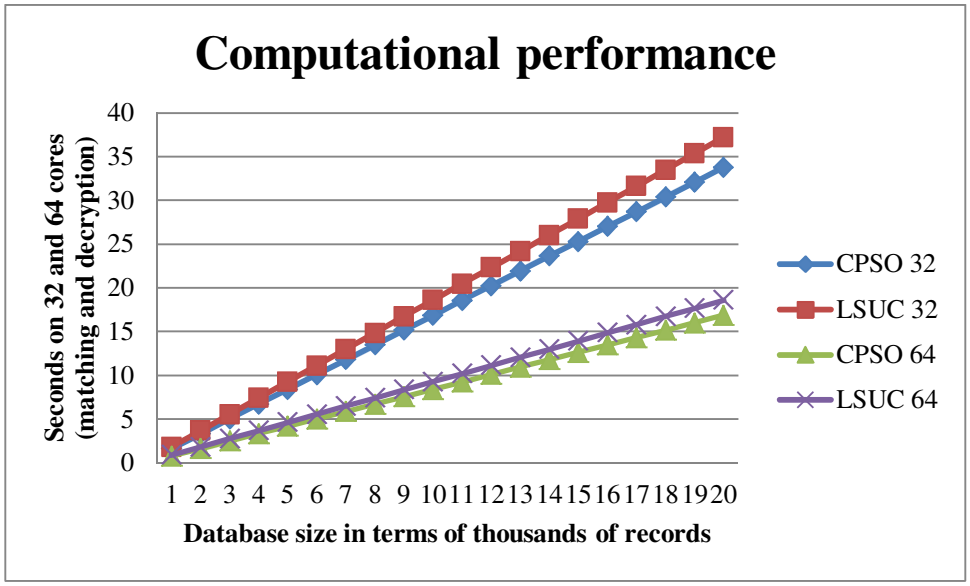


Figure 22 - Total computational performance for responding to a query for each of the two data sets.

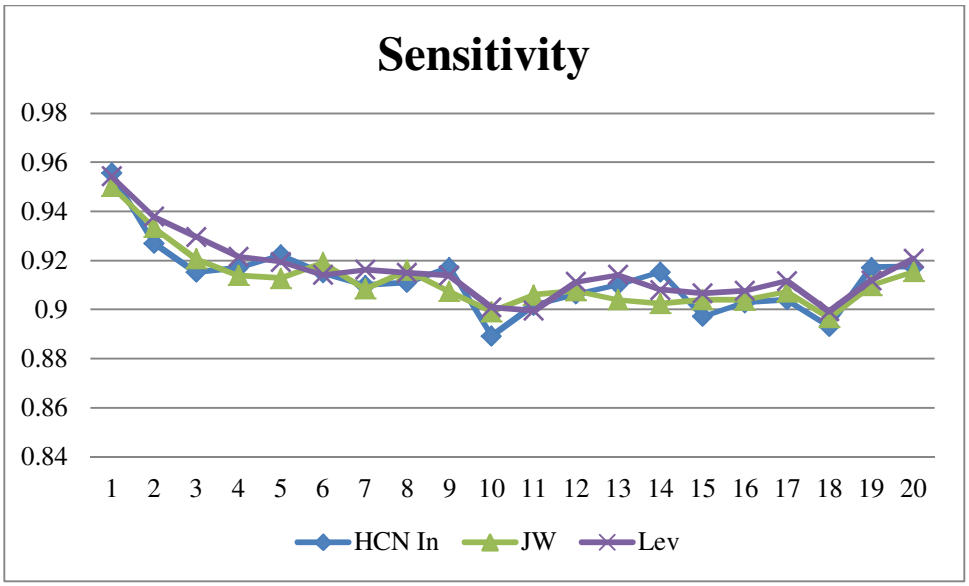


Figure 23 - Sensitivity of the dice coefficient approach for both data sets, compared to distance approaches from Jaro-Winkler (JW) and Levenstein (Lev), according to the database size in terms of thousands of records.

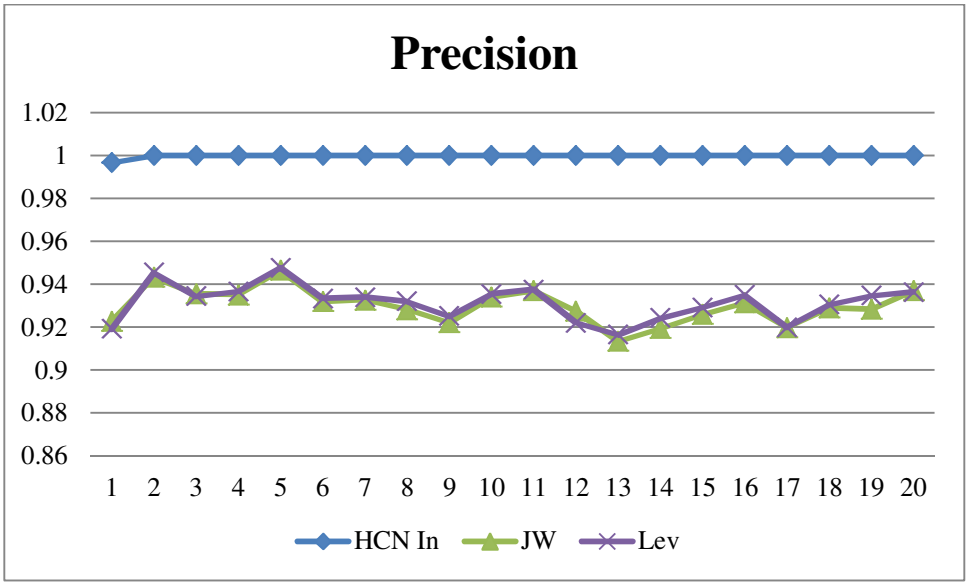


Figure 24 - Precision of the dice coefficient approach for both data sets, compared to distance approaches from Jaro-Winkler (JW) and Levenstein (Lev), according to the database size in terms of thousands of records.

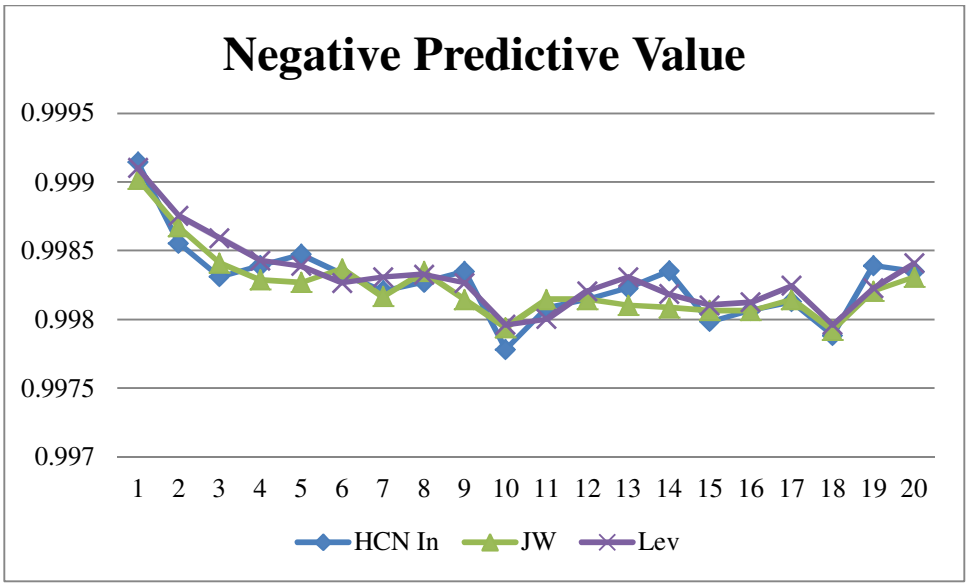


Figure 25 - Negative predictive value of the dice coefficient approach for both data sets, compared to distance approaches from Jaro-Winkler (JW) and Levenstein (Lev), according to the database size in terms of thousands of records.

3.10. Summary

We have provided a privacy-preserving protocol named TRACK for tracking phase 1 clinical trial participants to ensure that there are no concurrent enrollments. Our protocol is customized for this specific problem, and provides acceptable computational and accuracy results. The main benefit of such a protocol is that participants in these trials will have a much smaller risk to see their personal information being compromised, compared to current systems that hold their personal information on centralized databases accessible over the Internet. TRACK uses a new scheme (HCN In) to represent a last name that protects it against dictionary and frequency attacks both at the storage level and throughout the protocol runtime. A minimalistic combination of fields has been evaluated to enhance performance while representing the identity of an individual accurately.

We developed the DataRealtor tool to compensate for some limitations of current dataset generation infrastructures and to introduce the notion of prevalence to have the best representation of a real-world scenario.

We have also developed the ProEvalNet tool that was proven through our evaluation of a clinical trial protocol to help researchers evaluate various algorithms on multiple data sets distributed over several machines. The efficiency of our approach comes from the task distribution to complete a large task in a shorter amount of time, the interval specification to allow for prompt recovery from system crashes, and the flexibility to send commands and monitor progress on a secure network through a public infrastructure. ProEvalNet also provides flexibility to filter specific fields from a data set in order to test various aspects of a protocol without the need to generate additional data sets. The infrastructure we demonstrated could be re-used by other researchers to overcome the fundamental problems we have encountered.

In conclusion, TRACK satisfied our eight requirements offering strong privacy measures while accounting for small computational power at clinical trial sites. We developed a proof of concept and a sample user interface can be seen in Appendix B.

Chapter 4. Organizational SHARE protocol

Two organizations can benefit from a privacy-preserving protocol that is capable of discovering common entities in their data sets. In the healthcare context, discovering common patients enables the delivery of better and safer health services by sharing their medical records: doctors will be able to perform better assessments with access to a patient's complete medical record. In a cloud computing environment, patients with very challenging diseases can be added to a special group where their medical history can be collected from various different organizations in order to help solve their cases with access to more complete information. Other benefits include gathering statistics for disease control: reports could be generated to provide insight and relationships between diseases and causes of death. The aforementioned goals can be accomplished without the need to disclose information about the patients that are not in common between the two organizations. The privacy of those individuals needs to be protected. This research area is known as *privacy-preserving record linkage* (PPRL).

Our use case considers the healthcare context but PPRL has many practical applications in other domains. In the security and intelligence communities, PPRL enables different agencies to gather information about suspects or criminals they have in common without revealing any additional information about their most wanted list. It also enables airport security by matching the list of passengers on a plane against a "no fly list" while protecting the privacy of the individuals by not revealing their identities. In the business and marketing industry, two organizations can create a common plan to target the clients they have in common without revealing the identities of the clients they do not have in common for competitive reasons.

Private record linkage between two organizations could require linking two large data sets consisting of millions of records. Our previous duration upper bound (reported in section 3.9.3) per individual check for eligibility in a clinical trial is unacceptable under this scenario. We need new techniques to allow linking *millions* of records in a reasonable time. Depending on the organization, a duration constraint of multiple hours to a few days may be acceptable because they are not expecting an immediate answer. In the clinical trial case, the participant is *waiting* to

find out if he/she is eligible or not, so the result must be available in a matter of seconds or of a couple of minutes at most.

From the literature review, we learned that the three factors that compete against each other in a private record linkage system are performance, security, and accuracy. If we want to achieve the best performance, security and/or accuracy would not be at their best. The other combinations are true as well; when one of the factors is at its best, the other two will not. Our goal is to achieve the best security at an acceptable performance cost and accuracy given the confidentiality of medical records.

We named our protocol *SHARE* (corresponding to P4 in Figure 1) because our goal is to allow two organizations to share information about the individuals they have in common.

4.1. Motivation

Healthcare organizations could provide better care to their patients if they had their complete medical history. However, the complete details in patient's medical record can be distributed amongst multiple organizations without one organization containing the entirety of the information. Allowing organizations to link their medical databases in order to complete their patient records without learning anything about the patients they do not have in common will fulfill their goal of providing a better service to their patients. Statistical analysis based on empirical evidence is also a strong tool in combating the spread of disease and in performing better decisions where time and effort should be spent. Interactions between multiple organizations will lead to better statistics computed by either one of them (for example, how many deaths are caused by cancer?) and better long-term healthcare.

4.2. Problem statement

In the context of database linkage, two healthcare organizations want to find out what patients they have in common while protecting the identities of all the other patients that they do not have in common: it is possible that the sharing be only one way, for example if a cancer center wants to know which patients it has at another hospital but does not need to tell the hospital the result. It is possible for organizations to hold very large databases, with millions of records. Applying

record linkage to large databases has shown to be a heavily time consuming task. Techniques have been developed, such as *blocking*, to optimize the linkage mechanism while trying to maintain linkage accuracy. The challenge is to develop a blocking technique that is effective and that satisfies strong privacy requirements.

4.3. Use case

We allow two parties, A and B, to discover which records they have in common between their two data sets. These parties could be part of the same organization or two distinct organizations. In both cases, they only want to exchange information or gather statistics about the patients they have in common. The identities of the patients that are not in common must always be protected. The sharing of the common patients can also be one-way or two-ways.

The simplistic use case progresses as follows:

1. Party A sends party B encoded information about its data set.
2. Party B merges this information with encoded information about its data set.
3. Party B sends the information to the *key holder*.
4. The key holder can inform:
    a. One party about its matching records with the other party, or
    b. Both parties about their matching records.

The outcome of this linkage procedure is that one party will be able to discover which of its records also exist in the other party's data set and share its patient information in those records.

The choice of the last step will depend on the intended usage of the protocol by both parties. For example if one party wanted to find out which of its patients had a particular disease, it does not need to share additional information with the other party: it is sufficient to only get the information in a one-way transfer. However, if the intended usage was to complete medical records with both parties containing various distinct information about the patients, then the transfer of information must be bidirectional.

4.4. Literature review

4.4.1 Efficient private record linkage

Scannapieco et al. [109] argued that cryptographic protocols are inefficient and proposed a protocol that embeds the strings of the records to be matched in a vector space. The space is built from random strings and strings from the records are embedded using the *Sparse Map* method [58]. Both database owners send their vectors to a third party that compares them using the standard Euclidean distance. To make the protocol more efficient, using Sparse Map, the authors reduce the space dimension by applying a greedy re-sampling method and by distance approximations. However, the methods to achieve the performance gains negatively affect the accuracy of the protocol. Another problem with this protocol is that the third party will be able to mount a statistical attack based on the data it receives [53].

Pang et al. [92] presented approximate matching in *Privacy Preserving Record Linkage* (PPRL) using a public reference table. They presented an efficient three-party protocol that allows handling typographical errors in the data. However, a major drawback of this protocol is that if one party colludes with the record linkage party, they can help reveal the other party's private data. The other drawback of this protocol is that the size of clusters and the distribution of distances in a cluster can allow identification of rare names.

Yakout et al. [138] build on the protocol of Scannapieco [109] and try to optimize it by reducing the number of candidate pairs to be compared. They do not require a third party. However, the use of the secure scalar product protocol to do the comparisons requires extensive computations and thus rendering their two-party protocol less efficient than its predecessor [109]. The protocol they use has been suggested to be weakly secure [53][126]. Yakout et al. [137] developed a new protocol that has acceptable performance for large datasets but it is only capable of handling string fields at the moment. Their experimental evaluation was limited to datasets that contain only the first name and last name fields.

4.4.2 Private blocking

Private blocking is a recent research area that has been gaining attention due to the need of privately linking very large databases (in terms of millions of records) between organizations, i.e. hospitals, credit card companies, etc.

The first article about private blocking was authored by Al Lawati et al. and published in 2005 [2]. They presented a three-party protocol for private record linkage. In their approach, they discuss three alternative blocking techniques. The first is called *simple blocking* and consists of generating hash signatures from record values in order to group records in blocks. One drawback of this approach is that a record pair could be compared twice because a record can be added to more than one block. The second alternative is called *record-aware blocking*. The authors address the previous drawback by coupling a record identifier with every hash signature so that each record pair is now unique. The third alternative offers the best privacy protection and is entitled *frugal third party*. It uses a secure set intersection algorithm relying on a public key to determine the blocks they have in common. Only the common blocks will be sent to the third party for matching, resulting in more computations performed by the parties wishing to link their data sets, and fewer computations for the third party performing the matching. The blocking time for the third alternative is significantly larger than for the first two. All three methods suggested above are susceptible to frequency attacks.

Inan et al. [61] proposed a hybrid approach to PPRL. Their approach suggests building blocks based on attribute value generalization and specifying a percentage of records that will be compared using *secure multi-party computation* (SMC). They are able to achieve 100% precision, which means that irrelevant record pairs are protected against disclosure. Recall is affected by the upper bound chosen by the parties to restrict SMC costs. Hall and Fienberg [53] report that since Inan proposed this method, there have been many published vulnerabilities in the k-anonymization framework. Inan et al. [62] modified their previous approach that was based on k-anonymity generalization to introduce a new one based on *differential privacy* where the user can tweak the parameters of the method to either boost accuracy, privacy, or scalability.

Vatsalan et al. [130] proposed a two-party protocol based on reference values. Their work starts in a way similar to Pang's [92] but they remove the third party from Pang's protocol and

provide privacy by replacing the exact individual attribute table comparison values by ranges of values or *bins*. They provide scalability through phonetic encoding to generate blocks: the first step of their protocol is to generate the different blocks by applying an encoding function such as Soundex [130]. The next step is to use public reference lists to generate one or multiple reference values for each block. Following that step, each party can calculate the similarities between its records and the reference values: this work reduces the amount of time spent in the record linkage phase. The quality of the linkage will be dependent on the public reference that is chosen and for that reason, it can vary and be poor.

Karakasidis and Verykios [71] proposed a framework for secure record linkage in terms of secure blocking and secure matching. They claim that their blocking technique offers three levels of privacy: phonetic encoding, encryption, and random noise addition. However, that technique requires the sharing of a secret key for symmetric encryption using the MD5 message-digest algorithm (specified in IETF's RFC 1321). All the common phonetic codes could be discovered by the other party since a deterministic encryption method is being applied: the same phonetic code will map to the same cipher text, which will be the same for both parties since they share the same encryption key. The authors use MD5 to store elements in a Bloom filter, so it is susceptible to frequency attacks. The authors also do not describe how they calibrate the thresholds used in the paper stating that it is outside its scope. The calibration methods are an important step in assessing the correctness of a method.

Karakasidis and Verykios published another paper [72] where they use reference table based k-anonymous private blocking and claim that it is the only private blocking approach for any kind of data. They also claim that no information is leaked using their approach and that it is not susceptible to frequency attacks. However, their method requires a trusted third party. The clusters that they create are of different sizes (though a minimum size is defined but the authors do not mention how they can guarantee it), thus, allowing an attacker to gather some statistics. The solution they provide is only for private blocking, and they suggest the use of other techniques to perform PPRL afterwards.

Durham suggests a private blocking method based on Bloom filters *locality sensitive hashing* (LSH) in her thesis [36]. She describes LSH as a method that can map objects into partitions in a way that similar objects will share a partition with a high probability while

different ones will not. A hamming distance method (number of positions at which strings differ) is used to create partitions from the Bloom filters record encodings. Durham suggests that it is not obvious how an attacker can use the properties of the partitions to determine information about the plain text values, but that is not a real privacy guarantee especially with recently published attacks on the Bloom filter methods [86].

Schnell [112] and Wen [135] offer additional techniques based on Bloom filter encodings. However, given the privacy concerns over the level of privacy offered by these encodings [71][78][80][86], we cannot adopt them in a protocol where a strong privacy guarantee is required.

Karapiperis [74] provides a technique based on Bloom filters and a homomorphic matching technique. On top of the privacy concerns over Bloom filters, their protocol also requires a trusted third party. We discussed the difficulty in earning trust and the drawbacks it could have in terms of administrative approval when it comes to healthcare data. A protocol requiring a trusted third party cannot be considered for use in a real-world scenario.

Steorts [115] and Vatsalan [131] provide an evaluation of several privacy-preserving record linkage techniques and Vatsalan [129] also provides an evaluation framework with comparative results for some of the most recent techniques. Where applicable, we compare the results that they have to the ones we obtain for our protocol in section 4.10.

4.5. Requirements

In order to present the requirements for a well thought solution to private record linkage, we take into consideration the limitations of the protocols described in the literature and the considerations of real-world feasibility and practicality. In this scenario, one healthcare organization is trying to perform private record linkage with another healthcare organization to uncover which records they have in common. We outline the following requirements:

1) Information shall be exchanged in a privacy-preserving way.
    a) Information sent from an original party to another party shall not have personal identity information in the clear.

b) A third party that is given encoded information shall not be able to find out the original information associated with it.
c) No party shall be able to view or discover any plain record in the other's database.

2) The protocol shall not be susceptible to frequency attacks.
3) The protocol shall handle very large data sets (in terms of millions of records).
   a) The scenario we will be investigating accounts for a comparison of a data set of two million records, which accounts for a small to medium city (e.g., the Ottawa area has 900000 habitants) to another data set of four million records. Handling millions of records is important for large organizations. For example, the Mayo clinic had 1.3 million patients in 2014 [84] whereas Kaiser Permanente has around 10 million members, including around 3.5 million in each of North Carolina and South Carolina [65].

4.6. Methods

Private record linkage requires the ability to match records without knowing the original value of their fields. Transferring data to another party has to take into consideration two privacy measures: the other party cannot decode the data and cannot run a statistical analysis to deduce the original values. The private matching strategy of the clinical trial use case (from Chapter 3) satisfies these conditions but cannot be reused in our current scenario because it will be too slow for matching millions of records.

In order to develop the best methods for linking millions of records, we need to understand our environment and take advantage of what it has to offer. The main difference between this use case and the phase 1 clinical trial use case is that one of the parties knows the original values of the records to be compared. The central database for the clinical trials was responsible of forwarding comparison queries to the key holder, however, in this scenario, there is no central database. Instead, one of the parties will forward the queries to the key holder and that party knows the original values of its own records that will be sent for comparison. This key benefit allows us to take advantage of several strategies outlined in the following in order to optimize the comparison of the large amounts of data.

4.6.1 Private string matching

In the clinical trials scenario, the central database contained encrypted information and the site wishing to perform a query sent the CD encrypted data. In order to hide the individual name lengths from the key holder, the CD added random encryptions to the list of bigrams being compared, which did not affect the calculation of the dice coefficient. In our new scenario, we develop a method to minimize the length of the list of bigrams sent to be compared at the key holder's location. We also add random bigrams to mask the length of one of the names being compared as explained next.

Common Length Name Representation (CLNR)

The solution to keep the length of a name confidential is to represent names of different lengths by representations of the *same* length. We call this method CLNR [46]. The content of these representations is very hard to be differentiated because it belongs to an encrypted space that uses the ElGamal cryptosystem, which provides randomness even if we are encrypting the same value several times (the output encryption is different every time).

To represent a name that will be used in our protocol, we use a hash table where each bigram will be indexed into a position in the table. The remaining positions in the table will be filled with random data. All the values of the table are encrypted with the public key of the ElGamal cryptosystem.

We performed experiments to determine the appropriate size of the table required to store the bigrams with a minimum number of collisions in order to have all the bigrams needed for comparison. Figure 26 shows the results for both the maximum number of bigrams (16) and for an above average number of bigrams (8). The graph represents the results of averaging 10 000 runs of simulations adding bigrams to a hash table using the MD5 hash function.

We determine that a hash table of size 512 is appropriate for holding a name in our scenario. For the worst case scenario of 16 bigrams, even if we have one rare collision, its impact should not be relevant because when we are comparing a higher number of bigrams, if most of them match, the name will be considered a match. With a higher number of bigrams, a collision is slightly more likely but its effect is also less harmful.

The hash table has one item at each position. When a party sends an encrypted hash table to another party, the other party is not capable of performing any statistical attack because every position in the table is filled. However, because the other party knows the value of its names, it can determine which positions in the hash table it wants to match its known bigrams against because the index will be the same.

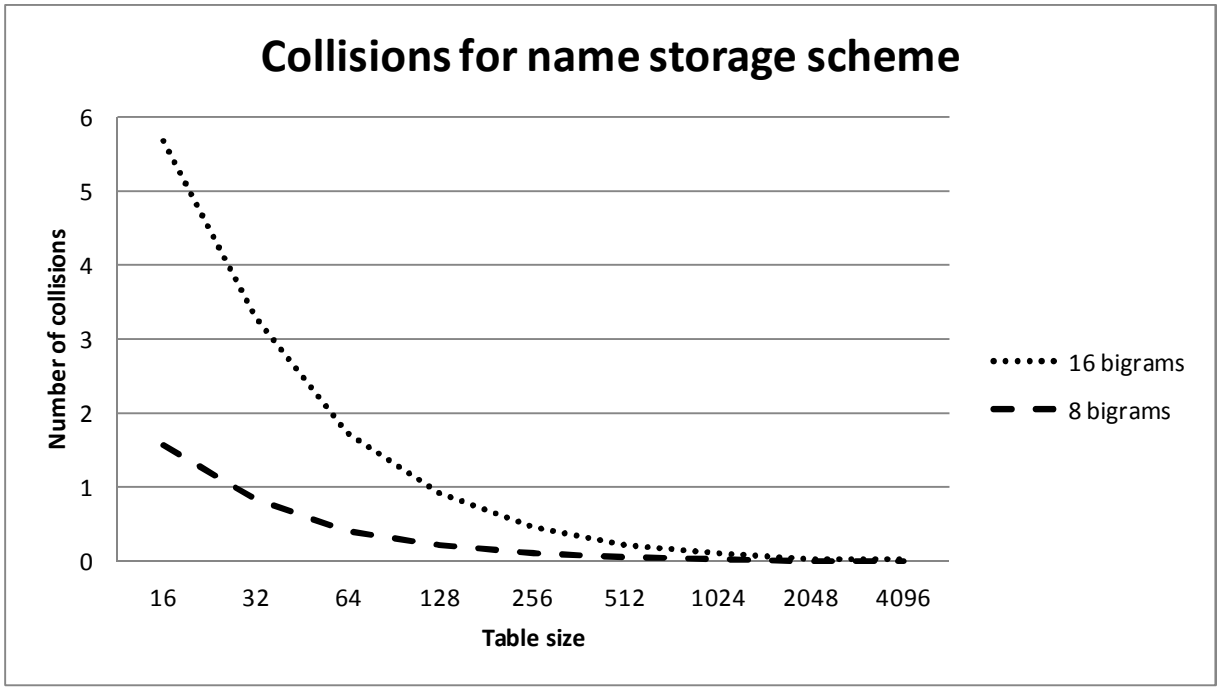


Figure 26 - Number of collisions for bigrams in a hash table of various table sizes

Name matching

In the SHARE protocol, one of the parties knows the original value of the name it is trying to match. The party that will forward the bigrams to be matched to the key holder (KH) knows which indices in the hash table it needs to compare against because it can derive them from the original name. However, it does not know if the encrypted values in the vector match them or not: this is the job of the key holder. Instead of sending the key holder a list of all the potential matches like the TRACK protocol, the party performing the query to the KH can now send the KH only the bigrams that have a real potential of being matches. This number corresponds to the length of the known name; therefore, the querying party can also include encryptions of random data that do not match any real-world data. Since such addition does not match, it does not affect

the calculation of the dice coefficient but conceals the value of the length of one of the names being compared.

How much are we saving?

The last name lengths in the 1990 US Census with the highest percentages are between four and eight inclusively (see Figure 2). If we were to use a standard comparison of every bigram, the number of bigrams compared would likely be between 16 and 64 (multiplying the name lengths). However, with the hash table based approach, and knowing the original value of one of the names, we would only compare 4 to 8 bigram tuples (the length of the known name). Therefore we reduce the complexity of the problem from $O(n^2)$ to $O(n)$.

This optimization does incur a cost in the name storage scheme to protect against statistical attacks. We need to store 512 bigram encryptions instead of an average of eight but we are less concerned with the storage space because terabytes of space are sold today at a very low cost accessible to individuals in the general population. The additional data does not require any processing and can be computed ahead of time so the overhead it requires is negligible at the time of private matching.

Our scenario compares two million records to four million records. However because the maximum number of matches is two millions, we will use that dataset as the starting point. Let us consider a best case (name length of four) and a worst case scenario (name length of eight).

In a scenario where all names have four characters, a standard comparison of every bigram would incur the cost of 2M*4*4 = 32 million bigram comparisons utilizing a storage space of 2M*4 = 8 million characters. Using the hash table approach would only compare 2M*4 = 8 million records but would utilize a storage space of 2M*512=1024 million characters. This is a 75% improvement in terms of computation power at the cost of 128 times extra storage space. However, that storage is temporary and is deleted after the matching is completed.

In a scenario where all names are eight characters, a standard comparison of every bigram would incur the cost of 2M*8*8 = 128 million bigram comparisons utilizing a storage space of 2M*8 = 16 million characters. Using the hash table approach would only compare 2M*8 = 16 million records and utilizes the same storage space as the 4-character names of 1024 million characters because it is independent from the length of the name. This is an 87.5%

improvement in computational power at the cost of 64 times the temporary extra storage space. Given the wide availability of storage space (at very little cost) and that the extra data will be thrown away and will not be processed because it is just masking the real information, we believe that this tradeoff is worth it to protect privacy and obtain a computational performance gain.

4.6.2 Efficient private record matching

Our data sets being compared have a unique identifier for each of their records: the health card number. In the SHARE protocol, we assume that two records that will be classified as a match need to have the *same* health card number.

Initially, we experimented with the idea of *indexing* in order to perform efficient private record linkage: the vision was to match only unique values in a column and propagate the match result to all duplicate values in that column (for example, the last name Smith will only be matched once against distinct names from the other dataset). However, we could not provide a solid privacy-preserving solution based on indexing. Therefore, we decided to experiment with private blocking based on hashing methodologies.

In order to perform data matching efficiently, several approaches use a deterministic hash function to allocate a position to each record based on value hashing of its health card number. We have already discussed the potential dictionary attacks against the hashing strategies, however, we are protected from dictionary attacks even if we are using deterministic hash functions because we are not storing the hash value of the fields but only using that value modulo the length of a list to determine an index for the record based on that field. We make sure that each position in the list has the same amount of encrypted values, which correspond to encrypted patient data. The positions that do not have data are filled with encrypted random data to fill the gaps. Based on the semantic security of the exponential ElGamal encryption, an attacker cannot tell if the encrypted data represents real patient data or random data. Figure 27 illustrates the process of preparing the original data to be sent to the other party for linkage.

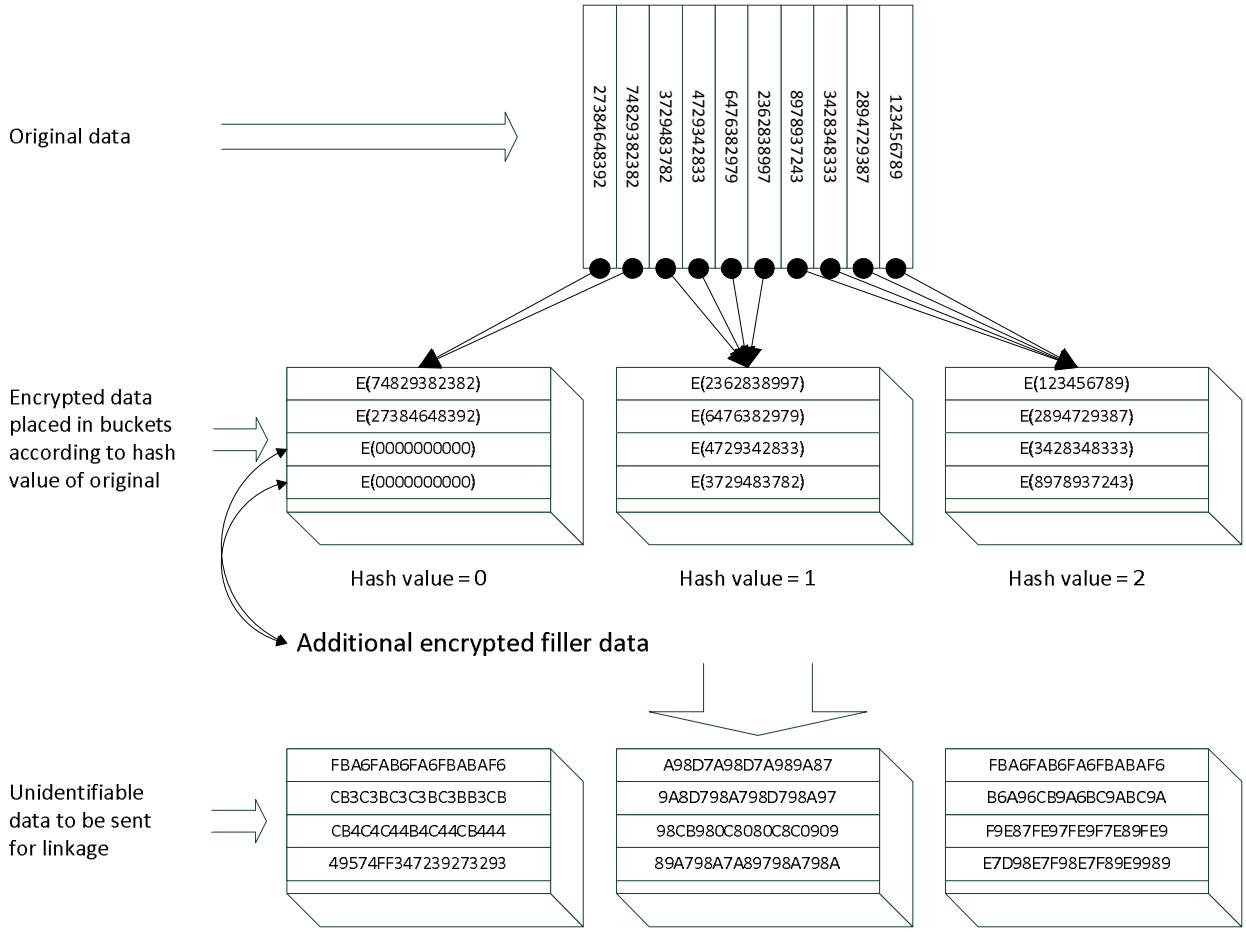


Figure 27 - Hashing and bucketing with filler data

This idea for efficient data matching relies on the fact that one of the parties knows the original value of the health card number. Given that fact, it can determine which position in the encrypted hash table that it received might have the matching record. Only the record-pairs created from the items in the bucket at that position need to be sent to the key holder for comparison instead of comparing against every record of the other data set.

In order to fill as little gaps as possible, we reviewed the literature for the best hashing data structures in terms of space utilization. The more space we could make our original records utilize, the less space would be required to be filled with random data to reduce the communication costs and enhance privacy. We also tried to validate some of the theoretical and experimental results in the literature for our scenario and they were different. This difference in results motivated us to run our own experimentation in section 4.7.

Freedman performed a study [49] where records were placed in bins using hashing with balanced allocations based on the theoretical results of Azar et al. [6]. They mention that Broder and Mitzenmacher [16] calculate the probability of a collision when we map a number of items to the same number of bins with *i* or more items: they claim that if each bin contains 5 items, the probability of a bin overflowing is 10^{-58}. They also claim if the hashing searches for the emptiest in three bins, and each bin contains three items, the probability of a collision is 10^{-33}. We tried to validate those claims in our empirical evaluation and our experiments showed us that the probabilities mentioned previously are questionable because we obtained collisions using the recommended settings. We ran the same experiment for our scenario using 32000 health card numbers and there were bins that had over 5 items (details in section 4.7). Based on the previous results, we decided to perform many evaluations to make sure the claims in the literature worked under our scenario with a health card number and millions of records.

Cuckoo hashing [90] offers a new scheme that offers worst-case constant lookup time. This is desirable in our scenario because the number of possible locations for a matching record is a constant and is quickly accessible. The scheme utilizes two hash functions to calculate the location of an element. The result of the first hash function is checked; if that location is empty, the element is inserted at that location, otherwise, it is inserted there and the existing element is kicked out and placed at the location specified by the second hash function (again, if there was an element at that location too, it is kicked out in the same way). Space utilization is maximized by moving elements around when a collision occurs. However, an insertion may require the relocation of a large number of elements and can even run into a cycle where the entire hash table needs to be rebuilt using different hash functions. The authors claim that based on random graph theory, if the load factor of the hash table is kept under 50%, the probability of running into a cycle is very low.

Cuckoo hashing with a stash [77] is an improved scheme over the original Cuckoo method. It admits the problem of the original method where the rehashing of the entire hash table may be necessary and that it is an expensive operation with a small but practically significant probability. The proposed solution adds a small side stash that can hold elements that may cause an infinite loop. The author claims that a stash of size three or four yields to major improvements

from the original scheme. They show their claims experimentally for a set of size 1000 and another one of size 10000 for different variations of Cuckoo hashing.

In [76], Kirsch and Mitzenmacher show that we can significantly increase the space utilization of a multiple hashing scheme such as Cuckoo by allowing a single move of an element when a collision occurs. We will explore this scheme in our experimentation in section 4.7 under the parameters of our scenario.

Random-walk Cuckoo hashing [52] provides an analysis of the insertion technique in a Cuckoo scheme with multiple insertion or relocation positions. The previously known insertion technique uses a breadth-first search approach that the authors claim to be inefficient. The insertion technique improves the performance of a protocol but does not affect the privacy because the end result will be the same hash table. The random-walk technique relocates items from random bins instead of using a breadth-first search method to find the emptiest bin and the authors show that this algorithm is more efficient then breadth-first search and achieves a polylogarithmic time with high probability of success.

In a research paper from Google and Microsoft [82], a review of the space utilization of Cuckoo hashing and its variants is described. They mention that the original Cuckoo hashing [90] achieves a space utilization of 50%. Fotakis suggested the use of multiple buckets instead of only two in [48]. Dietzfelbinger and Weidling [34] suggested the use of buckets that contain more than one item. The choice of two buckets that can contain two items leads to an improved space utilization of 89.7%. This would require to lookup four items if the hash table was completely full each time an element needs to be located. The authors also introduce the notion of 3.5 way Cuckoo for the price of 2 and a bit where they suggest an alternative implementation that uses overlapping memory instead of independent memory blocks that allows to achieve even higher space utilization of 96.5%.

Microsoft performed an evaluation of hashing methodologies [42] with a dataset of 64000 records. They highlight the techniques proposed by Fotakis et al. [48] and by Panigrahy [94]. They also evaluate different alternatives of the highlighted schemes and determine that using four independent locations is superior to using two pairs of dependent

locations, but this adds pressure on the memory system and requires additional computation of random values.

Astikis [3] presented a scheme for fast and compact hash tables that contain integer keys that perform better than bucketized Cuckoo hashing under certain scenarios. However, he also claims that bucketized Cuckoo performs better than his scheme under heavy load and with distinct data, which is what our scenario is about: maximal load of the hash table with distinct health card numbers. Our data set is also predefined so we do not need to dynamically expand the size of the table where the array hash implementation would be more efficient.

An interesting scheme is presented by Broder and Mitzenmacher [16] where a hash table is split into smaller sub-tables. Items are inserted into a sub-table that has buckets of a certain size. When a bucket overflows, we insert the item at a location in the next sub-table. This scheme could be helpful for our scenario because the left-most sub-tables are most likely to obtain the health card numbers we are looking for.

We validate these methods for the parameters of our scenario (millions of health card numbers) in the next section.

4.7. Empirical evaluation

The literature reviewed did not offer experiments that evaluated datasets that match our scenario. Certain theoretical results were also proven incorrect in our evaluations (outlined in section 4.7.1). Therefore, we needed to run our own experiments to validate the best hashing scheme to use.

We ran our evaluations using the following four hash functions[6]: MD5, SHA1, SHA256, and SHA512. We chose those four hash functions because they are widely known and they have an implementation in the .NET library, which assumingly is very efficient. Our goal was to find the best method for the minimalistic comparison of record pairs. The factors which impact the number of record pairs to be compared would be the size of buckets in the hash table or the number of hash functions that we use to determine several candidate locations for an item. These

[6] See http://en.wikipedia.org/wiki/Secure_Hash_Algorithm and http://en.wikipedia.org/wiki/MD5

factors are tuned to minimize the number of collisions in the hash table caused by an overflow of the capacity of a location in the hash table. We also try to optimize the space that the hash structure occupies. We will also be showing the amount of data transfer required for the different alternatives and how the same amount of space can work more efficiently in different setups.

The hashing schemes that we evaluate are the following: simple hashing (one hash function), multiple hashing (offers multiple possible locations for an item in a table), simple hashing with balanced buckets (a location can contain more than one item), multiple hashing with balanced buckets, and Cuckoo hashing with balanced buckets using only few moves.

4.7.1 Mismatch between theory and experimentation

In Freedman's paper [49], it is claimed that hashing n items to n bins with bins with capacity of 5, the probability of a bin overflowing is 10^{-58}. We ran the same experiment for our scenario using 32000 health card numbers and there were bins that had over 5 items. As a proof, we implemented the experiment in C# using the MD5 hash function implementation of the .NET library. Here are some of the keys and health card numbers that had over five items in their bins. In total, there were 25 collisions, which is closer to a probability of overflowing of 10^{-3}. We show partial colliding content in Table 8.

Table 8 - Keys and bin contents of an MD5 hash of 32000 health card numbers

Key	Bin content
27504 (9 items in the bin)	1306089264 3426876422 4059190525 5592390833 4746735261 8416903741 6637215047 3671212853 1018511184
2519 (7 items in the bin)	8741384160 4644643794 9568354414 7450836929 9553160714 5864311949 2964041445

Note that we calculated the number of collisions under the same scenario for the three SHA hashing schemes and the results were the following: 33 collisions for SHA-1, 15 collisions for SHA-256, and 19 collisions for SHA-512.

Table 8 shows that the presented theory is not applicable to our use case. We performed our own evaluation of several hash methods with different table space and bucket sizes to discover the best structure for our scenario.

4.7.2 Evaluation of collisions in various hashing structures

We evaluated the number of collisions for each of the following hash functions: MD5, SHA-1, SHA-256, and SHA-512. The first round of evaluation was based on the size of the data set, and the size of the hash table relative to the data set being hashed: same size, twice, four times, or eight times the size. We also utilized a bucketing strategy for the hash tables with bucket sizes varying between one, two, four, and eight items. The goal was to find out if a simple hashing strategy is enough to index our records and the size of the hash table/bucket required to achieve that goal with the fewest collisions possible.

The size of our data set was varied between one million and four million records. We performed the evaluation on 50 different data sets independently in order to avoid bias. The results are discussed after the following graphs. For space efficiency and comparative information, we used the same horizontal axis with values 1, 2, 4, and 8 to represent either the multiplication element of the table size based on the dataset size (for example, if the dataset size was one million, the axis points represent a table size of one million, then two, four, and eight millions) or the bucket size used in a table whose size is equal to the size of the dataset (for example, if the dataset size was one million, the axis points represent a table size of one million with bucket size one, then two, four, and eight elements per bucket).

Figure 28-Figure 31 show that the four hash functions that we are evaluating behave in a very similar way. Further evaluation can hence be only performed on MD5 in order to compare different structures of a hash table.

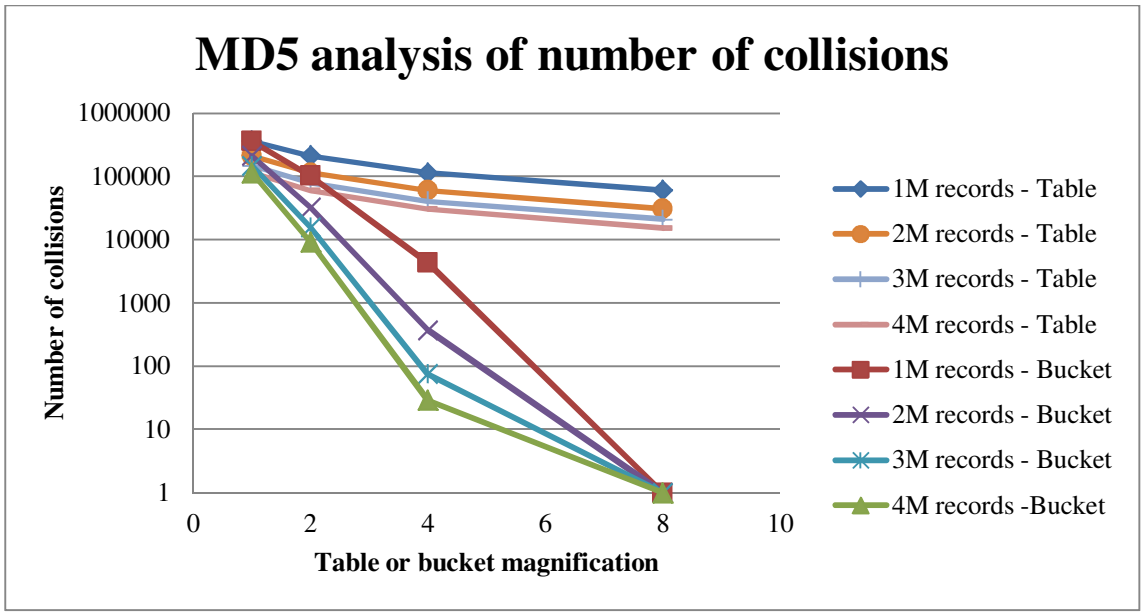


Figure 28 - Number of collisions using MD5

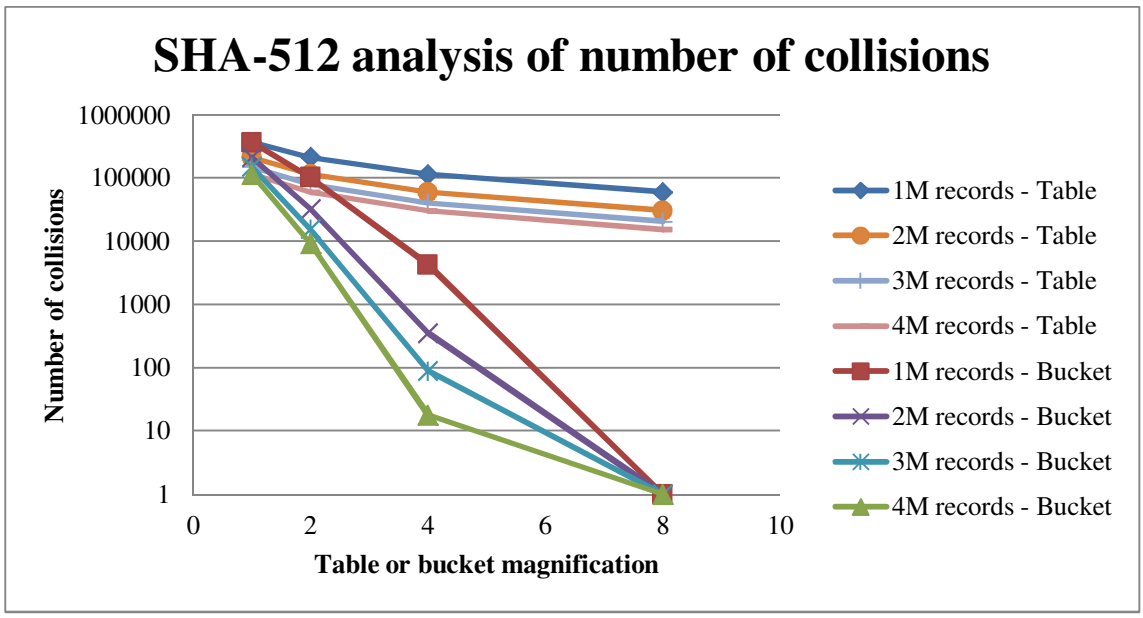


Figure 29 - Number of collisions using SHA-512

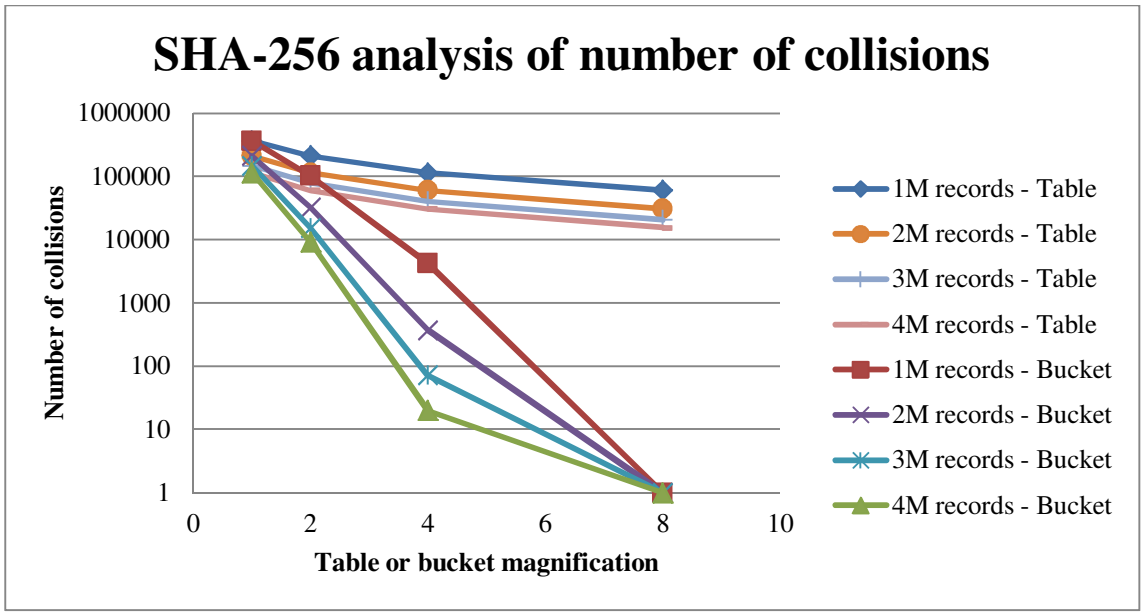


Figure 30 - Number of collisions using SHA-256

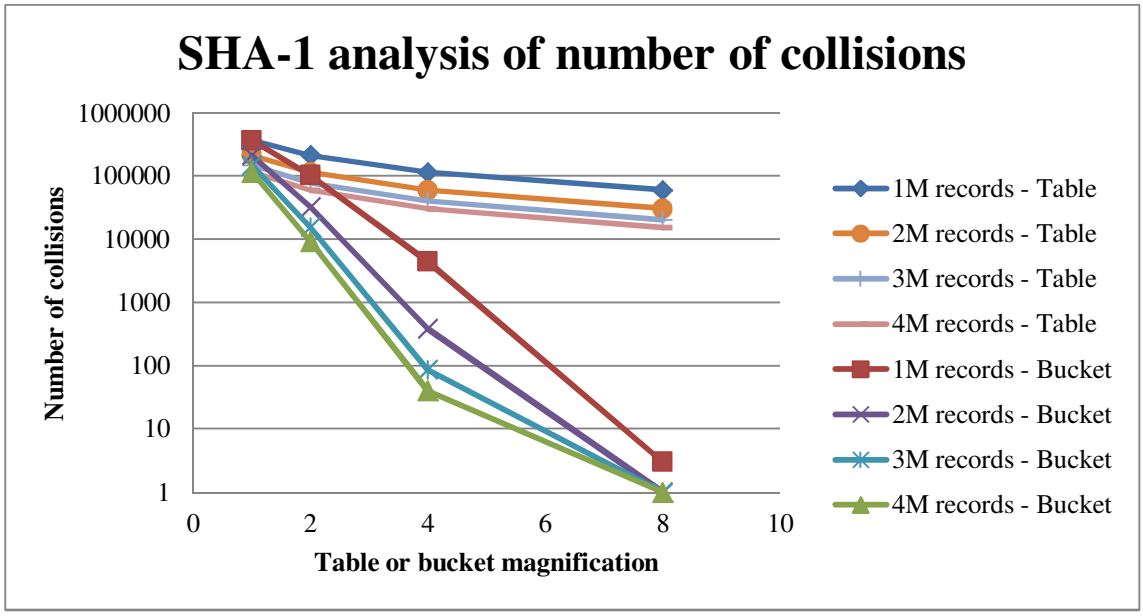


Figure 31 - Number of collisions using SHA-1

The four methods demonstrate the same properties that the use of buckets is far more superior for collision reduction than using a bigger table size. For the exact same space utilization, the number of collision is far less using buckets than using a bigger table. For example, for a dataset size of four million records, expanding the table size four times to 16 million still produces

40368 collisions while using a table size equal to the dataset size with a bucket size of four produces only 40 collisions, which is 1000 times smaller. The experiments also show us that using a bucket size of 8 enables minimal or no collisions. However, we jumped from a bucket size of four to eight; let us have a finer granularity between these two intervals to discover if we can have a suitable bucket size of less than eight. Figure 32 shows the results and we realize that the size of the hash table plays an important role in reducing the number of collisions (the size of the table is equal to the size of the dataset): for smaller datasets (one million records), the number of collisions is quite high with a bucket size of four but for the dataset of our scenario (four million records), a bucket size of five is enough to have near zero collisions (two collisions in the scenario), a bucket size of six produces only one and a bucket size of seven or eight have no collisions.

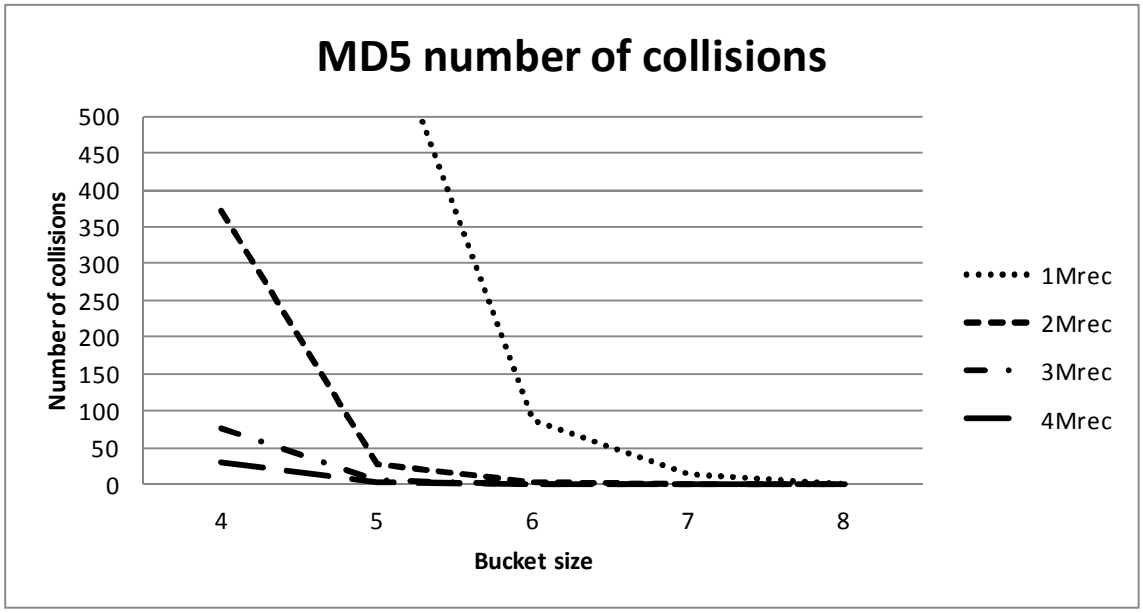


Figure 32 - Finger granularity bucket size for MD5

The next set of experiments will use additional techniques to discover if we can be more efficient in terms of the number of positions that an item can occupy in the hash table so we can reduce the size of the hash table required for minimal collisions.

Traditional hash tables use one hash function to map an item to a location in the hash table. The first modification we want to evaluate with millions of items is the use of two hash functions instead of one to map an item to two possible locations in the hash table: if the first

position is empty, we simply place the item there. However if the position already had an item, we will use the value from the second hash function to place the item at the other position. We are interested to see how this method compares to using a bucket size of two (same number of options for the location of an item). Furthermore, we performed evaluations using four hash functions and compared them to using a bucket of size 4 to understand the behavior of these data structures when dealing with millions of items.

Figure 33 and Figure 34 show the results of the evaluation and averages across 50 independent data sets for two hash functions and four hash functions, respectively, compared to one hash function with a bucket size of two and a bucket size of four, respectively. The evaluation is performed for dataset sizes ranging from one million to four millions. When comparing the two techniques, the table size used for both techniques is the same.

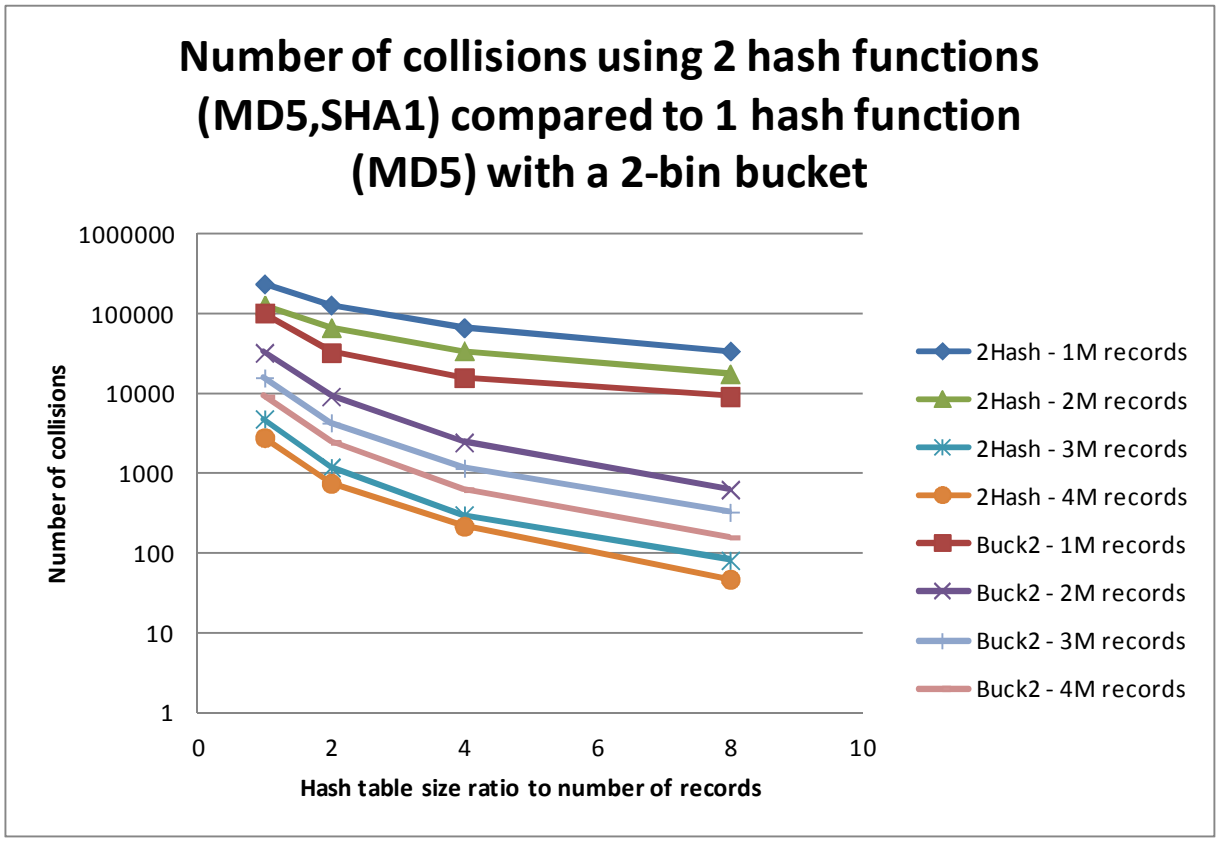


Figure 33 - Number of collisions using 2 hash functions compared to one with 2-bin bucket for the same table size

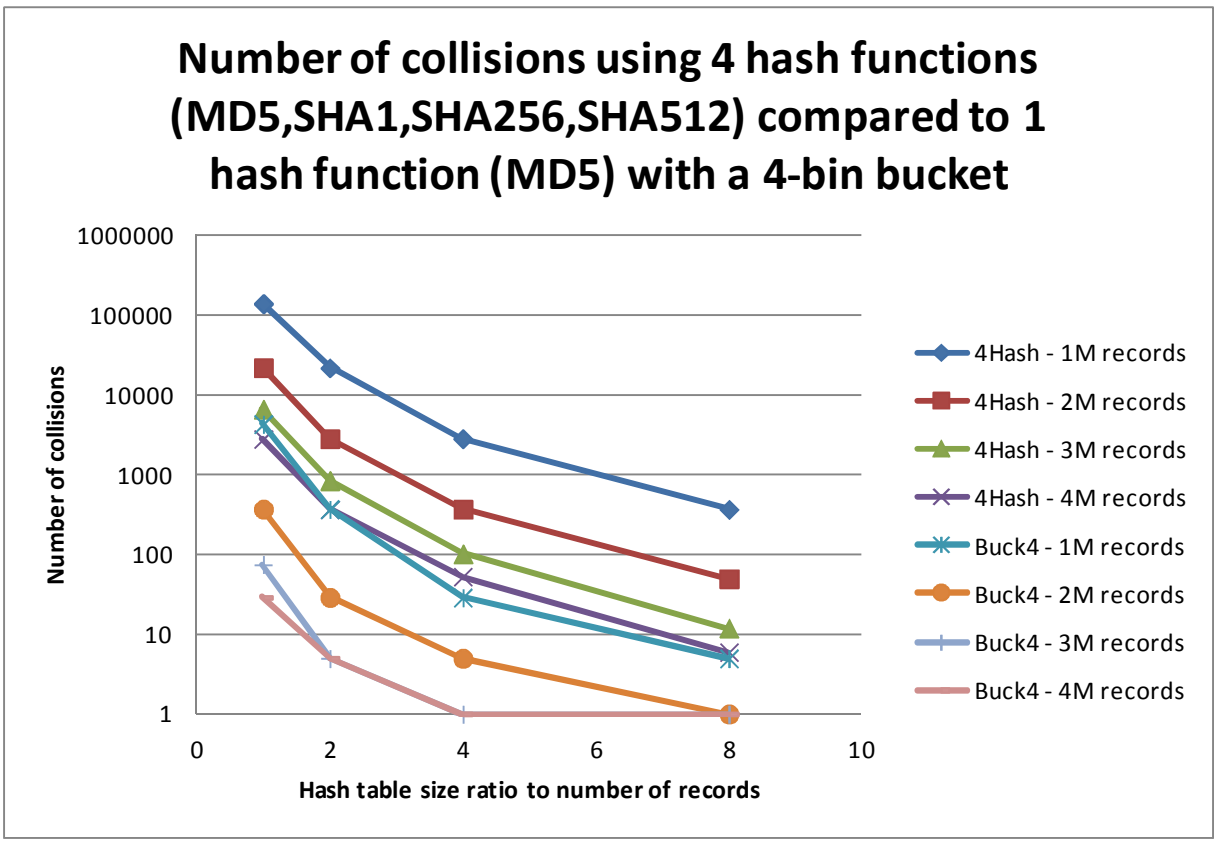


Figure 34 - Number of collisions using 4 hash functions compared to one with 4-bin bucket for the same table size

We notice that for our scenario containing four million records, a hash table of size 16,000,000 containing 4-bin buckets at each location is capable of containing all the records with no collisions. Note that we are less concerned with the size of the hash table and more concerned with how many locations an item can occupy in the table. The reason we are less concerned with the size is that we will not be checking every element in the table but only those at the location that match potential location of an existing dataset.

The previous hashing strategies offer schemes where once an item is placed in the table, it remains there and no other element can take its place. Cuckoo hashing on the other hand suggests a methodology where items can be moved to make space for others. Similar to using multiple hash functions, Cuckoo hashing suggest the use of two hash functions to determine two locations for an item in a hash table. If an item is at the location determined by the first hash function, it is allowed to be relocated to the location suggested by the second hash function in

case another item needs to be placed at its position. This does not affect how the lookup for an item is performed, since we have to look at both locations anyway. This technique might maximize the space utilization. The question that we need to answer is: can we do better than a bucket of size 4 with a hash table of size 16,000,000?

Figure 35 shows that Cuckoo hashing is not advantageous in our scenario. Because the Cuckoo scheme uses two locations for an item, a bucket size of two will require checking four locations. This corresponds to the minimum number of locations that we already achieved using balanced buckets with regular hashing. Under this configuration, Cuckoo hashing produces more collisions than the balanced buckets. We do not investigate configurations of Cuckoo with more hash locations or larger buckets including the d-left scheme [16] because this would require more than four locations to check against when trying to find a match for an item (two hash functions with a bucket size of two produces four locations to check against) and our goal is to reduce the number of items we need to check against when trying to match every item in a dataset.

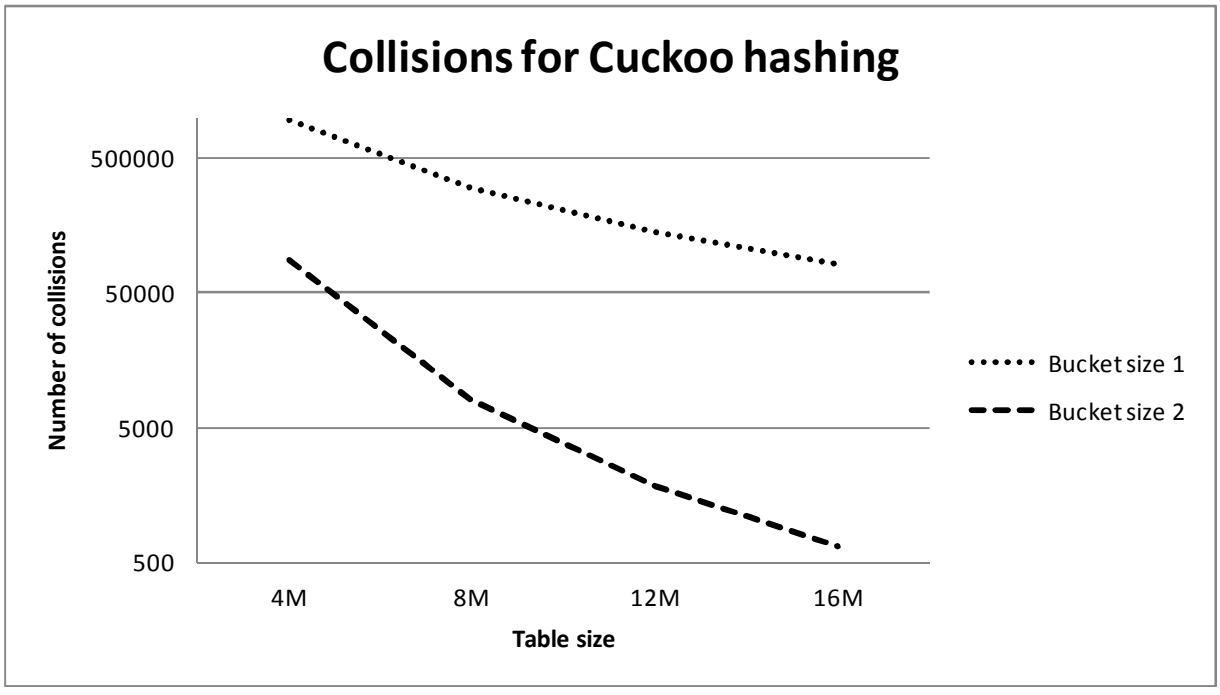


Figure 35 - Number of collision for Cuckoo hashing

4.8. SHARE protocol

In the following, we describe our SHARE protocol that can be used by two health organizations by the means of a semi-trusted third party to discover the patients they have in common. We have two parties interested in performing private matching of their patient records.

We are able to introduce an optimization for the matching methodology by delegating the task of forwarding the data to be compared to the party with the fewest records. The reason is that the maximum number of matches will be the maximum number of common records; therefore, if we can limit the number of records compared to the ones in the smaller data set, we would have achieved the lowest number of comparisons that leads to the discovery of all common records.

Let *A* be the party with the lowest number of records and *B* the party with the highest number of records. If *A* and *B* want to optimize the running time for the linkage procedure, they need to exchange the approximate number of records they have with each other in order to assign the roles properly. The exact number of records is not required because the optimization will not be worth it unless the difference is considerable (at least in terms of hundreds of records). Otherwise, the linkage procedure will still produce the same results but may take a longer time.

Phase 0 – Initiation

The KH generates a key pair in accordance with the settings specified in section 3.3.2 and sends the public key to site *A* and site *B* as illustrated in Figure 36.

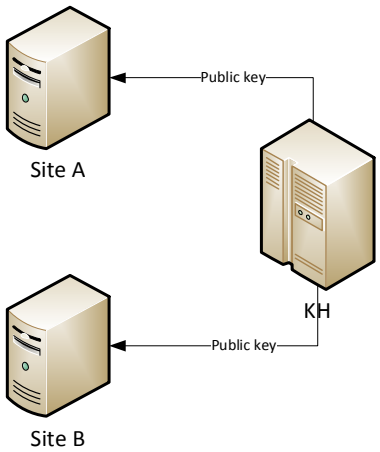


Figure 36 - SHARE protocol, phase 0, key sharing initiated by the key holder

Phase 1 – At the site B

Figure 37 illustrates the following steps of the protocol:

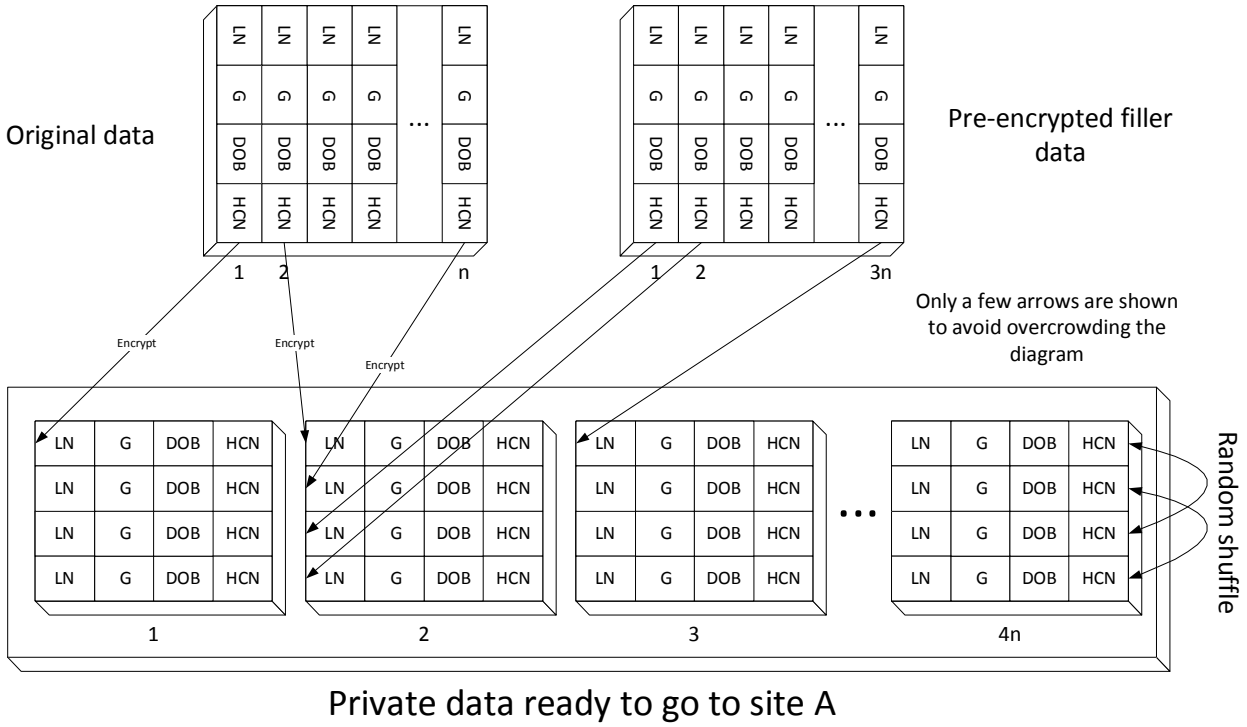


Figure 37 - SHARE protocol, phase 1, at the site B

1. *B* creates a table that is *four times* as the size of the dataset and that contains buckets with a capacity of four at each location (as in Figure 37).
2. *B* applies the hash function to the health card numbers in its dataset to calculate an index for each record in the table (see Figure 27).
3. *B* encrypts the records using the same scheme as the screening phase (section 3.4.2) of the TRACK protocol except the last name field where it uses the CLNR method from section 4.6.1 and adds the records at their suggested indices (using the hashed value of the health card number).
4. *B* fills in the gaps in the table with encrypted data that does not match real-world data: for example, a filler HCN can be 000000000 and a filler name can be a series of underscores. This is important because using a random value may have a minor probability of false positives; using a value that does not match a real-world value protects against that unlikely scenario.

a. Based on the semantic security of exponential ElGamal, party *A* will not be able to tell the difference between encryptions that represent real data and those that do not.

5. *B* randomly shuffles the order of the items in each bucket that contained a real item.
    a. We do this because the filler records are added after the real records even though all of them are encrypted. We do not want another party to be able to use this information even though we cannot think of an attack that may utilize it.
6. *B* sends A the entire table with encrypted data.

Phase 2 – At the site A

Figure 38 illustrates the following steps of the protocol:

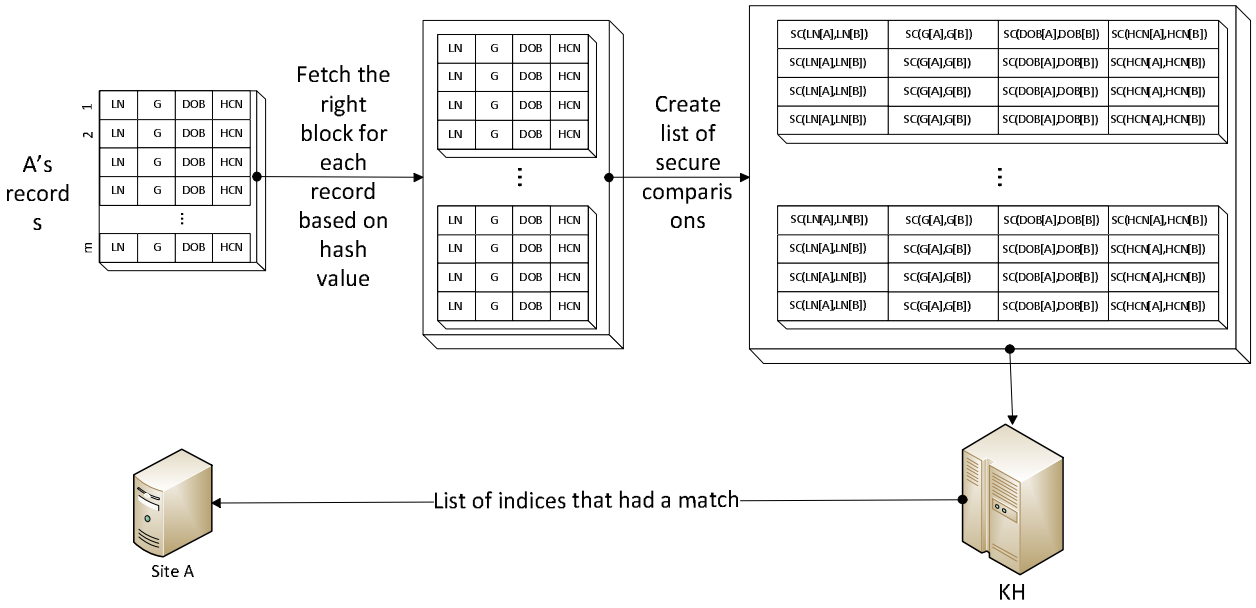


Figure 38 - SHARE protocol, phases 2 to 4, at the site A and then at the KH

1. *A* receives the table with encrypted data
    a. If party *B* had four million records, *A* receives a table of length 16 millions with buckets containing four items at each position.
2. *A* tries to prepare matching each of its records using the following method
    a. Find out the potential locations of a record from *A* in *B*'s table based on the hashing scheme with the health card number
    b. Encode *A*'s record with the same ElGamal public key as *B* similar to the TRACK protocol.

c. For each potential matching record in B, prepare the encoded comparison for each field
    i. Each encrypted bigram of the last name is multiplied by its potential match in *B*'s CNLR hash table to form a list of encoded bigram comparisons. Note that we limit the maximum number of bigrams to 16 to prevent the discovery of lengthy rare names (experiments in section 3.3.5).
    ii. The other field comparisons are encoded similar to the screening phase (section 3.4.2) of the TRACK protocol
d. Send the result to the key holder to discover if there was a match or not. This step is described in Phase 3.

Phase 3 – At the key holder

1. The key holder receives a list of four encrypted comparison of records
2. For each item in the list, the KH determines if the result is a match by trying to match all the fields (a match would lead to a decryption result of zero of the encrypted comparison, otherwise it would be a random number).
    a. When a match is found, the KH stops comparing the rest of the items in the list (bucket) it received because the health card numbers are unique.
    b. A match based solely on the health card number is not enough: it is more robust to match against all the fields in case a human error substituted a health card number by accident and to guard against a dictionary attack based on the HCN.

Phase 4 – Cleanup and communicating the results

At the end of this phase, *A* would have known which of its records are also present in *B*'s dataset based on the results obtained from the *KH*. *A* deletes all the data it temporarily received from *B* in order to perform the comparison. Depending on the use case or agreement, *A* can send additional information to *B* about the matched records or request additional information about them. *B* also disregards the filler data it used for this operation, otherwise, it risks the possibility of identification of filler data (in case it used the same filler in two distinct linking operations).

4.9. Requirements checkpoint

This section discusses how the requirements from section 4.5 are met by the SHARE protocol. The first requirement states that information shall be exchanged in a privacy-preserving way. Party *B* first encrypts its dataset with the exponential ElGamal scheme that utilizes a different random number for every encryption and a name representation scheme not susceptible to frequency attacks because every name is represented in the encrypted dataset with the same length (satisfies requirement *1a* and *2*). Given the encrypted dataset, party *A* is clueless of its contents. Party *B* does not have access to party *A*'s dataset anyway. For each comparison sent to the semi-trusted third party (key holder), party *A* multiplies two encryptions and raises them to the power of a random number. Even with its private key, the key holder can only find out if the result is equal to zero or to a random number satisfying requirement *1b*. The original datasets are protected from disclosure to any of the parties satisfying requirement *1c*.

The hashing data structure using balanced allocations with the appropriate parameters that we suggested allows the protocol to scale linearly with the database size. To get an idea of the amount of time it needs to link our scenario of two million records compared to four million records in a worst-case scenario (meaning when looking for an item in a bin, it will always be the last one, which is very unlikely): 4 elements per bin * 2 million health card numbers = 8 million secure comparisons (SC). For each match, to have a strong confidence, we can also match the other fields: 2 million last names + 2 million dates of birth + 2 million genders. The last name has on average 6 bigrams (2M*6 = 12 million SC), the date of birth has 5 variations (2M*5 = 10 million SC) and the gender requires one SC (2M*1 = 2 million SC). In total: 8 millions + 12 millions + 10 millions + 2 millions = 32 million secure comparisons. A decryption takes roughly under 1 millisecond with our implementation and there have been other optimized implementation for the same parameter with specialized math libraries that can perform the decryption in 155 microseconds [49]. Using the optimized implementation, the amount of time taken to decrypt the required information is about 83 minutes. This is a reasonable amount of time for an organization wanting to find our which records it has in common with another organization. Requirement *3* is therefore satisfied.

Note that the SHARE protocol can be implemented as a two-party protocol. However, the utilization of a professional organization with an expertise in computer security to play the role

of the KH and to hold the private key provides better guarantees for the privacy of the exchanged encrypted data between the two parties that wish to link their records.

4.10. Comparison with other methods

Vatsalan et al. [129] present an evaluation framework for privacy preserving record linkage. We will discuss some of the results presented for six privacy-preserving blocking techniques and show how our technique fits within these comparisons. We reproduced some of the graphs from Vatsalan et al. [129] and inserted the results of our methods to show the comparison side by side. We note that the SHARE protocol utilizes an exact match on the health card number, which might give it an unfair advantage over the other techniques that offer more flexibility in the matching process. Therefore, SHARE can be used in a first round of matching to efficiently eliminate all the matching pairs from future rounds that use less efficient but more accurate protocols if they can offer the required level of privacy in order to obtain the maximum number of matches.

The first technique, called *k-NN* and presented by Karakasidis and Verykios [72], is a three-party method based on k-nearest neighbor clustering and reference values. The second technique called *HLSH* is presented by Durham [36] and is based on locality sensitive hashing applied to Bloom filters [12]. Two additional techniques presented by Vatsalan and Christen [128] and are based on sorted neighborhood clustering: *SNC-3PSim* is based on similarity between reference values and *SNC-3PSize* is based on the size of the block. The *HCLUST* technique presented bu Kuzu et al. [80] adds noise to blocks generated using hierarchical clustering. Finally the *SNC-2P* method presented of Vatsalan et al. [132] uses two different sets of reference values to generate k-anonymous blocks then both parties exchange some of the block values to use a sliding window approach to determine the blocks to compare.

Blocking time

Blocking time estimates the amount of time needed in seconds to discover which record should be placed in which block. A relatively shorter blocking time means the evaluated approach is more scalable than one with a longer blocking time. Our blocking approach consists of indexing the elements into their respective buckets (buckets are blocks here) based on their hash value and the selected hashing scheme then by filling in the gaps with random encrypted data to make all

the blocks the same size. Note that we do not consider the random data generation and its encryption to be part of our blocking time because this data can be computed ahead of time and simply copied into the empty spaces which would have negligible time. Computing a hash value for a health card number took an average of 4E-06 seconds on an Intel Centrino 2 laptop with a dual core processor running at 2.26 GHz with 4 GB of main memory on Windows Vista. Vatsalan et al. performed the evaluation on a computer server with two 64-bit Intel Xeon CPUs running at 2.4 GHz with 128 GB of main memory, which places us at a major disadvantage. Figure 39 (adapted from [129]) shows the comparison of all the approaches, this time including SHARE. We notice that the scheme used in our SHARE protocol performs best for larger datasets (over 172938 records) and grows linearly with the size of the dataset. SNC-3PSim and SNC-3PSize performed better than SHARE for datasets containing under 172938 records but the blocking time for both these approaches and SHARE is under one second, which is acceptable.

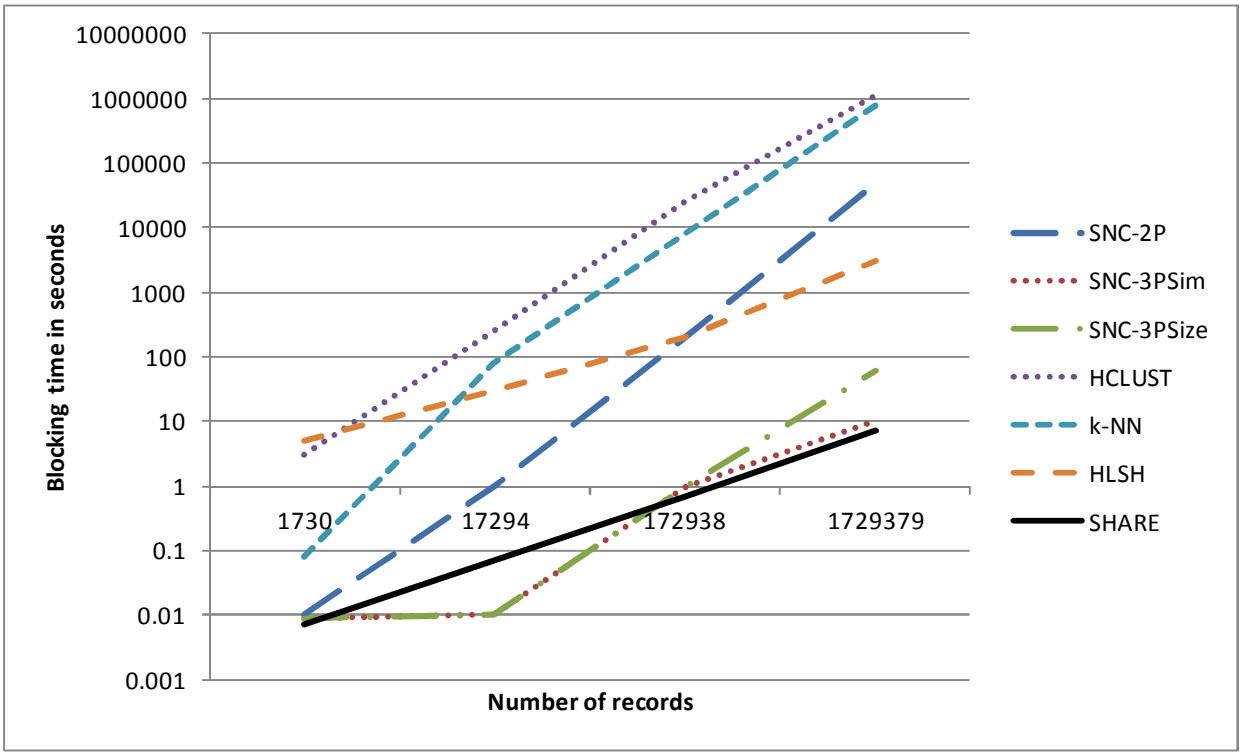


Figure 39 - Comparison of blocking time, modified from Vatsalan et al. [129]

Reduction ratio and pair completeness

Reduction ratio (RR) represents the inverse of percentage of records that need to be compared from the entire dataset. A high reduction ratio means that only a small portion of the dataset

needs to be compared to find out if there is a match or not. It is a measure of how *efficient* a blocking technique is.

Given two datasets, pair completeness (PC) measures how many true matches were assigned to the correct block versus the total number of true matches between the two entire datasets (the closer PC is to a value of 1, the better). It is a measure of how *effective* the blocking technique is.

Figure 40 (adapted from [129]) compares the RR and PC of six privacy-preserving techniques to the one used in our SHARE protocol. We observe that the reduction ratio is very high for all the techniques and that our technique performs best followed by SNC-2P and HLSH in terms of pair completeness.

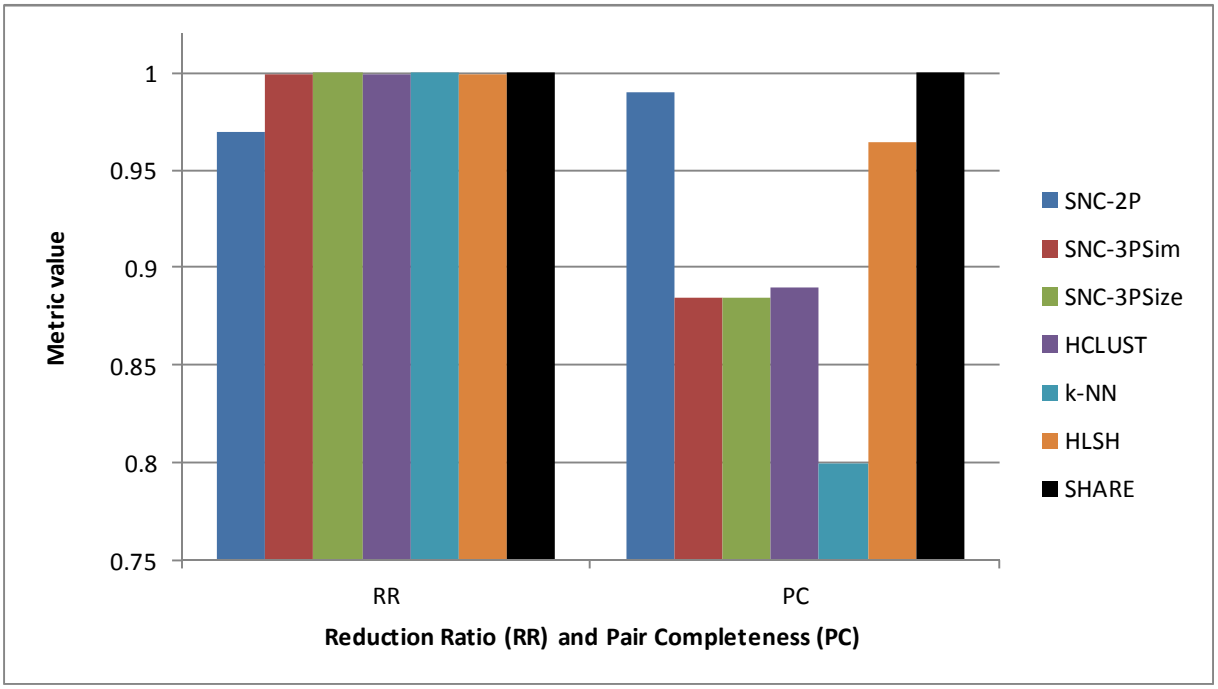


Figure 40 - Reduction Ratio and Pair Completeness comparison, adapted from Vatsalan et al. [129]

Block size

The sizes of the blocks generated by the six private blocking approaches are compared in [129] and it is stated that SNC-based approaches and HCLUST have lower variance between block sizes than HLSH, which make frequency attacks using block size more difficult. However, none

of the compared approaches have a constant block size. Figure 41 (based on the results of [129]) shows the block size distribution of the compared approaches and of our SHARE protocol (note that the results for 3P-Sim and 3P-Size are almost identical so they were represented by a common entry named SNC-3P): the box plot[7] shows the maximum (but rare) block size indicated by the red star, the horizontal line inside each box indicates the median value (50% of the values are bigger than the median) while the vertical line above it shows the highest 25% of the values and the one under it shows the lowest 25%. Our approach has a constant block size. A frequency attach based on block size using our method is infeasible since all blocks have the same size. This method of privacy protection is not offered by any of the compared approaches.

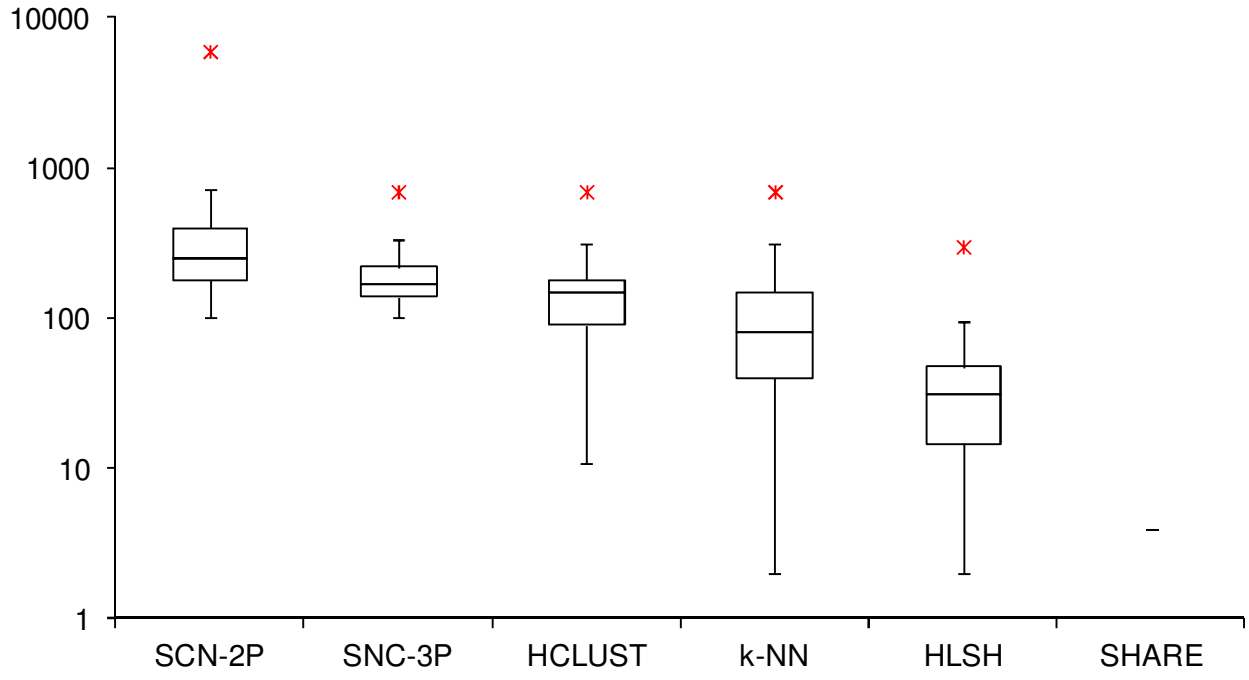


Figure 41 - Box plot showing the distribution of the block size, based on Vatsalan et al. [129]

[7] How to read a box plot can be found at
http://www.cbsolution.net/techniques/ontarget/box_plots_clearly_explained.html

4.11. Summary

In this chapter, we discovered the balanced allocation method with one hash function and a bucket of size four is the most optimal approach to organize four million numeric health card numbers. We provide a strong privacy measure by filling in the unoccupied items in the hash table with encrypted data that cannot be distinguished from real numbers. These gap fillers do not affect the computational performance of the matching protocol because the receiving party will only check the positions that its numbers could have occupied (there is some overhead in the positions that required topping up the bucket). In future work, it would be a good idea to test datasets larger than four millions to find out of the same hashing structure is still the most optimal or not but from the trend that we observed, the larger the hash table is, even if the dataset is also growing, the fewer collisions were observed. Hence, we believe that a hash table four times the size of the dataset with a bucket containing four elements should work well.

Chapter 5. Conclusion

Privacy-preserving patient tracking for phase 1 clinical trials adds an important measure to protect the integrity of clinical trials and the safety of their participants. The ability to verify if a participant is concurrently enrolled in another clinical trial allows a contract research organization to disqualify that participant immediately. Detecting a dual enroller at a later stage in the process will require the CRO to throw away the data already collected on the participant and search for another person to replace him. This incurs additional costs and time delays in the evaluation of a new drug aimed to be released to the public. The verification process also protects the participants from potentially harmful drug interactions. CROs require a protocol to check if a participant is concurrently enrolled at another site while protecting the individual's identity information for two reasons: the first is privacy and the second is the challenge of finding participants in phase 1 clinical trials so an organization wants to protect its population of participants from disclosure to a competitor.

The challenges with private matching and private record linkage were approached by several researchers through different techniques which we outlined in our literature review. Though using Bloom filters has growing interest in the literature, it has also been shown that such filters are not mature enough to provide privacy guarantees. Kuzu et al. [79] presented the first cryptanalysis attack on Bloom filter encodings and showed that they were able to recover some of the names in a database correctly. Researchers at the German Record Linkage Center [86] claim that little research on the security of Bloom filters has been published and present an attack that requires less computational effort and less resources than the one presented in [79] to be successful. They also offer techniques to make their attack more difficult to achieve but not infeasible. While the two attacks in [79] and [86] were based on Bloom filters that encode only one field at a time, and some of the previous authors [36] claiming that encoding more than one field in the Bloom filter makes it harder to be decoded, Kroll and Steinmetzer [78] present the first successful automated attack on Bloom filters encoding more than one identifier. In light of these recent attacks that are evolving and becoming less computationally demanding, we do not

recommend that Bloom filters be used at the moment where strong privacy guarantees are required. The medical domain requires a strong privacy measure because patient and participant data is of a very sensitive nature and is the subject of many laws.

We started our protocol development by considering Bloom filters and opted out of their usage while basing our protocols on a better established foundation that has been proven to be private and reliable (widely used in the industry), notably the exponential ElGamal cryptographic scheme, which is based on a hard mathematical problem of solving a discreet logarithm. We provided several optimizations and techniques to make our work feasible in a phase 1 clinical trial scenario.

Our evaluations used multiple datasets from multiple English speaking countries: Canada, United States, and Australia. Our evaluations were performed over different independent versions of the data sets to reduce bias and improve generalization. The datasets were generated from data extracted from phone books, census, professional databases and voter lists.

Our research was expanded to include the use case of linking large datasets containing millions of records in a privacy-preserving manner. While techniques existed to perform this task, we put our primary focus on privacy protection and limited disclosure. We analyzed existing hashing data structures to discover the appropriate configuration for our use case. We also created a new scheme for representing a last name called *CLNR,* which provides better efficiency in terms of performance while maintaining the same accuracy of our *HCN In* scheme. The *CLNR* scheme also provides strong privacy measures by representing every name using a hash table of the same length. Our privacy-preserving record linkage scheme has better blocking time than existing approaches with large datasets. It does however require a unique identifier, which is the health card number in case of Canada. The *SHARE* protocol provides a fixed blocking scheme which does not disclose any additional information about the dataset, a feature that is not provided by most current private record linkage protocols.

In this chapter, we discuss our contributions, the expected impact of our work, current threats to validity, some limitations, and future directions.

5.1. Contributions

The main contribution of our work is the research and development of the *TRACK* protocol for private matching of an individual in a data set in the context of phase 1 clinical trials. Our TRACK protocol enables multiple clinical trial sites with limited computational power to collaborate in a privacy-preserving manner to discover dual enrollers. The protocol was developed with extensive analysis to guard against frequency attacks (based on the *HCN In* name representation scheme [45]) that most protocols based on a deterministic encryption scheme (e.g., [14][29][71][109][113]) would be susceptible to. It provides error handling in names and dates of births, only requires semi-trusted parties, and has a linear performance as the database size increases. Table 1 on page 32 classified the protocols based on important criteria for the phase 1 trials scenario. In Table 9, we show how our TRACK protocol fits within this context.

Table 9 - Complement to Table 1 on protocols' characteristics and drawbacks

Authors	Technique	FAP	LCPS	EH	NTP
Farah (this thesis)	TRACK, HCN In, SMC	✓	✓	±	±

Legend: **FAP** (Frequency Attack Prevention), **LCPS** (Low Computational Power at Site), **EH** (Error Handling), **NTP** (No Trusted Party required). '✓' for well supported, '±' for partial support, and '' for no support.

An additional contribution of our work is the research and development of the *SHARE* protocol to link large datasets (simulations were based on two million against four million records) in a privacy-preserving way in the presence of a unique identifier. Our SHARE protocol includes techniques and a *CLNR* name representation scheme that take advantage of the context of the application of the protocol where one party knows the data it is trying to match: for example, one hospital trying to match its patients with another hospital. It has a linear complexity. SHARE offers strong privacy measures by building a constant block size to compare data efficiently, a measure that is not available in most blocking techniques [80][128][132] where protocols rely on adding noise or on a certain data distribution. However, SHARE requires a unique identifier to work properly, unlike other protocols that sample data from multiple fields in a record to be more robust in case of missing fields. Organizations may perform private record linkage incrementally by running the SHARE protocol to get a first set of matches very efficiently on a

large number of health card numbers, then eliminating these matches from the next round where another less-efficient protocol that uses more fields is applied (for example [62] or [130]), if the latter can provide an acceptable level of privacy protection, in order to obtain the maximum number of matches possible.

The known attacks listed in our literature review that TRACK and SHARE can handle are discussed in Table 10, together with the protective measures established by the protocols to guard against them. Note that the concurrency and space reduction attacks are types of attacks that we illustrated during the development of the TRACK protocol and guarded against.

Table 10 - List of attacks TRACK and SHARE guard against

Type of attack	TRACK guards	SHARE guards
Dictionary	ElGamal randomized encryptions	ElGamal randomized encryptions
Frequency/Statistical	ElGamal randomized encryptions, HCN In	ElGamal randomized encryptions, CLNR
Space reduction (section 3.2.2)	HCN In, adding random bigrams	CLNR
Concurrency attack	Individual checked while present at the clinical trial site (cannot be in two places at once).	Not applicable.
Eavesdropping	SSL, digital signatures	SSL, digital signatures
Spoofing (malicious)	SSL, digital signatures	SSL, digital signatures
Bloom filter specific [78][79][86]	Not applicable. Those attacks rely on the analysis of the Bloom filter structure, which we do not utilize in our techniques.	

Minor contributions include the creation of several datasets based on the 1990 US Census, the lists of doctors and lawyers in Ontario, the North Carolina voters' database, and parts of the Australian phonebook along with the *DataRealtor* tool [44] that helped making them more in line with real-world scenarios by setting the correct prevalence and offering utilities to tweak them. Both the datasets and the DataRealtor tool are provided to other researchers (Appendix C) to help them test their work and allow them to create additional datasets easily. The *ProEvalNet* tool was also created to run the evaluations efficiently [43] by supporting execution on multiple machines and then merging of the results. ProEvalNet also offers the capability of operating from behind a firewall by accepting commands to run evaluations from a Dropbox folder (only

the commands are supplied via Dropbox, the dataset information is secured on the machines and not shared via Dropbox or another file synchronization utility). ProEvalNet is also made available in Appendix C. A website based on ASP.NET was built to demonstrate a user-friendly interface that can be used by clinical trial sites to create studies, add participants and check the eligibility of participants. Some screenshots can be seen in Appendix B.

5.2. Expected impact

A major impact of our work will be on the social scale. Developing protocols with real-world scenarios in mind will hopefully facilitate their adoption in industrial product development. Contract research organizations will be able to run phase 1 clinical trials with better confidence and assurance of the quality of their results, hence making new drug development processes faster and allowing the quicker delivery of safe and effective drugs to consumers. Hospitals will be able to take advantage of private record linkage systems to complete their patients' medical records or to coordinate with centers for disease control on the spread of pandemics or population health statistics.

From a computer science perspective, the designs of our protocols aim to achieve a solid security measure with a practical performance criterion. This is especially important in our day and age because of the emergence of mobile devices that have a somehow limited computing capacity but rely on heavier back-ends or cloud computing. Our phase 1 private tracking protocol fits nicely in the context of today's technology where it can be run simply from a tablet at the recruiting clinical trial site.

5.3. Limitations and threats to validity

The steps that led us to develop our privacy-preserving TRACK protocol started with the emergence of the concurrent enrolment problem in phase 1 clinical trials. The prevalence of this real-world problem is documented in Appendix A.

Limitations to the TRACK protocol include the situation where one of the clinical trial sites in proximity of the others refuses to be included in the system for detecting dual enrollers: this site would pose a threat to the concept because all the individuals that go to that site would

not be detected by our protocol. However, we made the process of a site joining the team very easy: the key holder sends the new site the public key that is used to contribute all its records to the central database and it becomes part of the private matching group as soon as it completes that step.

We conducted various experiments to test and improve our protocols. Unlike observational studies that can be affected by human factors, we had better control on the variables because our studies were experimental using computer programs to process data.

Runeson et al. [107] state that "the validity of a study denotes the trustworthiness of the results, and to what extent the results are not biased by the researchers' subjective point of view". They classify the threats to validity in four categories: construct validity, internal validity, external validity, and reliability.

Construct validity is concerned with whether researchers are indeed measuring the concept or metric they have in mind. In case of a study that includes human participants, the interviewed person may have a misconception of the question and his understanding of the question may not be the same as the researcher. In such a case, there would be threats to construct validity. However, in our experiments, we measured metrics that are well defined in the literature based on mathematical formulas seen in multiple papers from different sources. The threats that could have influenced these calculations are related to internal validity.

Internal validity needs to be examined when researchers are investigating how one factor influences the outcome of an experiment. Researchers need to mitigate the risks that other factors may also be contributing to that outcome. In our performance measurements, we made sure that the computer was in an idle state before running our evaluation program: unnecessary programs were terminated to ensure that the performance measurement was affected only by our running program.

External validity determines if the results of a given evaluation can be generalized to other domains and similar use cases. It was a main consideration while we designed our experimental evaluations. The risks that we wanted to mitigate in our evaluations included the following: bias in using a dataset from only one country, bias in introducing typographical errors not representative of real-world errors, and inaccurate name frequencies in generated datasets

compared to their real-world counterparts. Prevalence was also an important factor for phase 1 clinical trials. We wanted to measure the accuracy of our protocol given a similar percentage of individuals that try to enroll in trials concurrently. We chose to set the prevalence value at 2% because clinical trial managers reported a concurrent enrolment percentage between 2% and 7% (Appendix A) and we discovered from our experiments with prevalence values between 1% and 10% that higher prevalence values tend to have better accuracy so we wanted to provide a worst case scenario with acceptable accuracy numbers.

To address the first risk of a dataset from a single country, we used data sources based on the 1990 US census data, the College of Physicians and Surgeons of Ontario (CPSO) database and the Law Society of Upper Canada (LSUC) database. It could be argued that the doctor and lawyer names hold a particular socio-economical status that is different from less fortunate people who are more likely to participate in clinical trials. To mitigate this risk, we split-composed names into multiple independent names, for example Jean-Pierre became Jean and Pierre. We also included two additional datasets: the North Carolina voters' database and parts of the Australian phone book. Using data sets from Canada, the United States, and Australia gives us more confidence in the external validity of our work in western English speaking countries. However, it could still be argued that the datasets we generate based on the general population data do not represent accurately the subset of the population that actually participates in clinical trials.

Another threat to external validity is that the populations and typing mistakes that were generated are not similar to the ones that exist in real-world cases. To mitigate this threat, we used the Febrl tool [21][98] to generate our simulated populations based on the 1990 US census, the College of Physicians and Surgeons of Ontario (CPSO) database, the Law Society of Upper Canada (LSUC) database, the North Carolina voters' database and parts of the Australian phone book. Febrl has been used by many other researchers working on the same type of problems. It generates populations based on the name frequencies in the databases to simulate real-world populations. The errors introduced by Febrl are also following the same distribution of real errors in real-world datasets: insertions, deletions, substitutions, OCR, etc.

To gain confidence in our generated data sets, we analyzed each data source to extract its name frequencies in order data sets that closely matched with the real-world versions. Febrl

accepted a frequency file as input and generated populations with names based on the values of the frequency file. Based on our analysis of the original data sets, we created the frequency files required by Febrl and used Febrl to generate data sets of similar, smaller, or bigger sizes that still had similar frequency properties as the original data sets.

We created 50 different versions of each data set based on five distinct populations (we sampled 10 datasets from each population). The number 50 was chosen as a good reference point based on prior experiments in the Electronic Health Information Laboratory (EHIL). Our accuracy and performance results are the average results of running each experiment independently over 50 different versions of each data set.

Additional threats to external validity in our studies include our scenario of linking two million records against four million records in our SHARE protocol. We determined the appropriate hashing methodology and configuration but we did not provide a theoretical guarantee to this claim; it is based on experimentation. If similar datasets incur collisions under this configuration, the size of the bucket will have to be larger than four. We ran the experiments on 50 independent datasets in order to reduce bias. Some applications may require larger datasets. However, our experimental results showed a trend where larger hash tables, even if the dataset is also larger, encountered fewer collisions. Hence, we believe that a hash table four times the size of the dataset with a bucket containing four elements should work well for larger datasets.

In addition, our experiments where based on nine digit numbers for the unique identifier. It would be interesting to run the evaluations with different formats of the unique identifier, for example ten digits or ten digits with two letters to see if there is any difference with hashing values and table load to generalize the findings for different unique identifiers. We also tested only four different hash functions; testing a larger number of hash functions could provide different configurations that could be more optimal then our current settings.

Lastly, for external validity, the protocols have not yet been deployed in a real-world environment. However, we tried our best to mimic a real-world environment taking into consideration prevalence between the datasets and real-world errors in generated datasets.

The last type of validity concerns is called reliability: it highlights the threat of a research being dependent on a single researcher, stating that the same research results should be obtained if the same research was performed by another person. The evaluation code was completely written by the author of this thesis. However, he has over four years of full-time industrial software development experience which augments our confidence that the written code is correct. Moreover, logical variations of the experiments (for example, using fewer fields) affected the accuracy calculation as expected, which also boosts the confidence that the code is working correctly.

5.4. Future work

The following items would be interesting and important to investigate or to perform:

- Deployment of a system based on TRACK to a real group of CROs to better suit their needs and how to develop a complete solution for detecting concurrent enrolment.
- Deployment of a system based on SHARE to perform privacy-preserving record linkage between two real organizations.
- Evaluation of our methods using datasets from countries using another language (e.g., French) or alphabet (e.g., Arabic or Chinese).
- Evaluation of the SHARE hashing scheme with a wider range of hash functions to compare their distribution of the records across the hash table and determine if there is a more efficient representation technique to store them in a privacy-preserving way.
- Evaluation of the SHARE hashing scheme with datasets reaching one hundred million records to determine how to dynamically allocate a hashing structure and methodology.
- Extension of SHARE to support the linking of a few records only against an entire dataset to support the cases of time-sensitive access to information for urgent new patients, or for incremental verification.
- Evaluation of the effect of a last name change on the performance of TRACK and SHARE (though this is not a frequent change, usually a one-time event).

- Development of our data set evaluation infrastructure to support additional process automation in order to save researchers time and help them reach their evaluation goals faster.
- El Emam et al. [40] developed a protocol that secures the identity of a data provider in order to monitor the spread of a disease by aggregating data from several providers (without revealing their identities) and presenting only the sum of the results to a health authority. It would be of interest for clinical trial sites to include that work along the TRACK protocol to detect concurrent enrollers and report on their numbers without affecting the credibility of a site (in case it had many concurrent enrollers).

References

[1] Agrawal, R., Evfimievski, A., and Srikant, R.: Information sharing across private databases. ACM SIGMOD, San Diego, pp. 86-97, 2003.

[2] Al-Lawati, A., Lee, D., and McDaniel, P.: Blocking-aware private record linkage. 2nd international workshop on information quality in information systems. ACM, pp. 59-68, 2005.

[3] Askitis, N.: Fast and compact hash tables for integer keys. Proceedings of the Thirty-Second Australasian Conference on Computer Science-Volume 91. Australian Computer Society, pp. 113-122, 2009.

[4] Atallah, M.J., Kerschbaum, F., and Du, W.: Secure and private sequence comparisons. Proceedings of the 2003 ACM workshop on Privacy in the electronic society. ACM, pp. 39-44, 2003.

[5] Ayala-Rivera, V., McDonagh, P., Cerqueus, T., and Murphy, L.: Synthetic Data Generation using Benerator Tool. arXiv preprint arXiv:1311.3312, 2013.

[6] Azar, Y., Broder, A. Z., Karlin, A. R., and Upfal, E.: Balanced allocations. SIAM journal on computing, 29(1), pp. 180-200, 1999.

[7] Bachteler, T., Schnell, R., and Reiher, J.: An empirical comparison of approaches to approximate string matching in private record linkage. Statistics Canada Symposium, 2010.

[8] Baxter, R., Christen, P., Churches, T.: A comparison of fast blocking methods for record linkage. ACM SIGKDD, Vol. 3. ACM, pp. 25-27, 2003.

[9] Bellovin, S.M., and Cheswick, W.R.: Privacy-Enhanced Searches Using Encrypted Bloom Filters, Technical Report, Department of Computer Science, Columbia University, USA, CUCS-034-07, 2007.

[10] Berman, J.J.: Zero-check: a zero-knowledge protocol for reconciling patient identities across institutions. Archive of pathology and laboratory medicine, 128(3), pp. 344-346, 2004.

[11] Bigard, R.: National death index matching criteria. Available from: [http://www.cdc.gov/nchs/data/ndi/NDICriteria_front.pdf]

[12] Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Communications of the ACM, 13(7), pp. 422-426, 1970.

[13] Bonomi, L., Xiong, L., Chen, R., & Fung, B.: Frequent grams based embedding for privacy preserving record linkage. Proceedings of the 21st ACM international conference on information and knowledge management. ACM, pp. 1597-1601, 2012.

[14] Bouzelat, H., Quantin, C., and Dusserre, L.: Extraction and anonymity protocol of medical file. AMIA Fall Symposium, p. 323, 1996.

[15] Boyar, D., and Goldfarb, N.: Preventing overlapping enrollment in clinical studies. Journal of clinical research best practices, 6(4), pp. 1-4, 2010.

[16] Broder, A., and Mitzenmacher, M.: Using multiple hash functions to improve IP lookups. INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings, Vol. 3. IEEE, pp. 1454-1463, 2001.

[17] Broder, A., and Mitzenmacher, M.: Network applications of bloom filters: A survey. Internet Mathematics, 1(4), pp. 485-509, 2004.

[18] Buczak, A., Babin, S., and Moniz, L.: Data-driven approach for creating synthetic electronic medical records. BMC medical informatics and decision making, 10:59, pp. 1-28, 2010.

[19] Camps, N., and Daud, N.: Improving the efficacy of approximate personal name matching. Proceedings of the 8th International Conference on Applications of Natural Language to Information Systems, pp. 70-76, 2003.

[20] Canadian Institute for Health Research Best Practices: http://www.cihr-irsc.gc.ca/e/29072.html (accessed January 2015)

[21] Christen, P.: Probabilistic data generation for deduplication and data linkage. IDEAL'05, LNCS 3578. Springer Berlin Heidelberg, pp. 109-116, 2005.

[22] Christen, P.: Data matching. Springer Data-Centric Systems and Applications, 2012.

[23] Christen, P.: Geocode Matching and Privacy Preservation. ACM PinKDD, LNCS 5456. Springer Berlin Heidelberg, pp. 7-24, 2009.

[24] Christen, P.: Privacy-preserving data linkage and geocoding: Current approaches and research directions. PADM held at IEEE ICDM. IEEE CS, pp. 497-501, 2006.

[25] Christen, P., and Churches, T.: Secure health data linkage and geocoding: Current approaches and research directions. National e-Health Privacy and Security Symposium, Brisbane, 2006.

[26] Christen, P., and Goiser, K.: Quality and complexity measures for data linkage and deduplication. In: Quality Measures in Data Mining. Studies in Computational Intelligence Vol. 43. Springer Berlin Heidelberg, pp. 127-151, 2007.

[27] Christen, P., Zhu, J., Hegland, M., Roberts, S., Nielsen, O., Churches, T., and Lim, K.: High-Performance Computing Techniques for Record Linkage. Proceedings of the Australian Health Outcomes Conference, 2002.

[28] Churches, T.: A proposed architecture and method of operation for improving the protection of privacy and confidentiality in disease registers. BMC Medical Research Methodology, 3(1), 2003.

[29] Churches, T., and Christen, P.: Some methods for blindfolded record linkage. BMC Medical Informatics and Decision Making, 4(9), 2004.

[30] Clinical Trials Ontario Working Groups Summary Report: http://www.ctontario.ca/wp-content/uploads/2014/06/Clinical-Trials-Ontario-Working-Groups-April-2013.pdf (accessed January 2015)

[31] Cramer, R., Gennaro, R., and Schoenmakers, B.A.: A secure and optimally efficient multi-authority election scheme. European Transactions on Telecommunications, 8(5), pp. 481-490, 1997.

[32] Dean, J., and Ghemawat, S.: MapReduce: simplified data processing on large clusters. Communications of the ACM, 51(1), pp. 107-113, 2008.

[33] DeMets, D., Friedman, L., and Furberg, C.: Fundamentals of Clinical Trials. Springer 4th Edition, 2010.

[34] Dietzfelbinger, M., and Weidling, C.: Balanced allocation and dictionaries with tightly packed constant size bins. Theoretical Computer Science, 380(1), 47-68, 2007.

[35] Du, W., Atallah, M.J., and Kerschbaum, F.: Protocols for secure remote database access with approximate matching. ACM Workshop on Security and Privacy in Ecommerce. ACM, pp. 87-111, 2000.

[36] Durham, E.A.: A framework for accurate, efficient private record linkage. PhD Thesis, Vanderbilt University, USA, 2012.

[37] Durham, E.A., Xue, Y., Kantarcioglu, M., and Malin, B.: Private medical record linkage with approximate matching. AMIA Annual Symposium, p. 182, 2010.

[38] Durham, E.A., Xue, Y., Kantarcioglu, M., and Malin, B.: Quantifying the correctness, computational complexity, and security of privacy-preserving string comparators for record linkage. Information Fusion, 13(4), pp. 245-259, 2012.

[39] Dusserre, L., Quantin, C., and Bouzelat, H.: A one way public key cryptosystem for the linkage of nominal files in epidemiological studies. Medinfo, 8:644, 1995.

[40] El Emam, K., Hu, J., Mercer, J., Peyton, L., Kantarcioglu, M., Malin, B., Buckeridge, D.L., Samet, S., Earle, C.: A secure protocol for protecting the identity of providers when disclosing data for disease surveillance. JAMIA 18(3), pp. 212-217, 2011.

[41] El Emam, K., Samet, S., Hu, J., Peyton, L., Earle, C., Jayaraman G.C., Wong, T., Kantarcioglu, M., Dankar, F., and Essex, A.: A Protocol for the Secure Linking of Registries for HPV Surveillance. PLoS ONE 7(7): e39915. doi:10.1371/journal.pone.0039915, 2012.

[42] Erlingsson, U., Manasse, M., and McSherry, F.: A cool and practical alternative to traditional hash tables. Proc. 7th Workshop on Distributed Data and Structures (WDAS'06), 2006.

[43] Farah, H., Amyot, D., and El Emam, K. A Tool for Simple and Efficient Clinical Protocol Evaluation. Computer-Based Medical Systems (CBMS), 2014 IEEE 27th International Symposium on. IEEE, pp. 499-500, 2014.

[44] Farah, H., Amyot, D., and El Emam, K. Real-World Data Set Parameters and Synthesization for Matching Identity in Clinical Protocols. Computer-Based Medical Systems (CBMS), 2014 IEEE 27th International Symposium on. IEEE, pp. 263-266, 2014.

[45] Farah, H., Amyot, D., and El Emam, K.: Efficient Privacy-Preserving Identity Scheme for Electronic Validation of Phase 1 Clinical Trials. 6th International MCETECH Conference. Lecture Notes in Business Information Processing, vol. 209. Springer, pp. 183-196, 2015.

[46] Farah, H., Amyot, D., and El Emam, K.: Common Length Name Representation: An Efficient Privacy-Preserving Scheme. In proceedings of ICSE 2015, 1st International workshop on technical and legal aspects of data privacy and security. Florence, Italy. 2015 (to appear)

[47] Fellegi, I.P., and Sunter, A.B.: A Theory for Record Linkage. Journal of the American Statistical Association, 64(328), pp. 1183-1210, 1969.

[48] Fotakis, D., Pagh, R., Sanders, P., and Spirakis, P.: Space efficient hash tables with worst case constant access time. STACS 2003, LNCS 2607. Springer Berlin Heidelberg, pp. 271-282, 2003.

[49] Freedman, M. J., Hazay, C., Nissim, K., & Pinkas, B.: Efficient Set Intersection with Simulation-Based Security. Journal of Cryptology, pp. 1-41, 2013.

[50] Freedman M.J., Nissim, K., and Pinkas, B.: Efficient private matching and set intersection. Advances in Cryptology. EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, LNCS 3027. Springer, pp. 1-19, 2004.

[51] Friedman, C., and Sideli, R.: Tolerating spelling errors during patient validation. Computer and Biomedical Research, 25(5), pp. 486-509, 1992.

[52] Frieze, A., Melsted, P., and Mitzenmacher, M.: An analysis of random-walk Cuckoo hashing. APPROX and RANDOM 2009, LNCS5687. Springer Berlin Heidelberg, pp. 490-503, 2009.

[53] Hall, R., and Fienberg, S.E.: Privacy-preserving record linkage. Privacy in Statistical Databases, LNCS 6344. Springer Berlin Heidelberg, pp. 269-283, 2011.

[54] Health Canada Industry Guide for Good Clinical Practice website: http://www.hc-sc.gc.ca/dhp-mps/prodpharma/applic-demande/guide-ld/ich/efficac/e6-eng.php (accessed January 2015)

[55] Hermann, R., Heger-Mahn, D., Mahler, M., Seibert-Grafe, M., Klipping, C., Breithaupt-Grogler K., and de Mey, C.: Adverse events and discomfort in studies on healthy subjects: the volunteer's perspective. A survey conducted by the German Association for Applied Human Pharmacology. European journal of clinical pharmacology; 53(3-4), pp. 207-214, 1997.

[56] Hernández, M.A., and Stolfo, S.J.: The merge/purge problem for large databases. ACM SIGMOD Record, 24(2), ACM, pp. 127-138, 1995.

[57] Herzog, T.N., Scheuren, F., and Winkler, W.E.: Data quality and record linkage techniques. Springer, 2007.

[58] Hjaltason, G.R., and Samet, H.: Properties of Embedding Methods for Similarity Searching in Metric Spaces. IEEE TPAMI 25(5), pp. 530-549, 2003.

[59] https://www.dropbox.com/, visited on January 19, 2014

[60] Hylton, J.A.: Identifying and Merging Related Bibliographic Records. Master's thesis, Massachusetts Institute of Technology, USA, 1996.

[61] Inan, A., Kantarcioglu, M., Bertino, E., and Scannapieco, M.: A hybrid approach to private record linkage. IEEE ICDE. IEEE CS, pp. 496-505, 2008.

[62] Inan, A., Kantarcioglu, M., Ghinita, G., and Bertino, E.: Private record matching using differential privacy. International Conference on Extending Database Technology. ACM, pp. 123-134, 2010.

[63] Jain, N., Dahlin, M., and Tewari, R.: Using bloom filters to refine web search results. Proceedings of the Eighth International Workshop on the Web and Databases, 2005.

[64] Jaro, M.A.: Advances in Record-Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida. Journal of the American Statistical Association, 84(406), pp. 414-420, 1989.

[65] Kaiser Permanente website: http://share.kaiserpermanente.org/article/fast-facts-about-kaiser-permanente (accessed April 2015)

[66] Kantarcioglu, M., Inan, A., Jiang, W., and Malin, B.: Formal anonymity models for efficient privacy-preserving joins. Data and Knowledge Engineering, 68(11), pp. 1206-1223, 2009.

[67] Kantarcioglu, M., Jiang, W., and Malin, B.: A privacy-preserving framework for integrating person-specific databases. Privacy in Statistical Databases, LNCS 5262. Springer Berlin Heidelberg, pp. 298-314, 2008.

[68] Kantarcioglu, M., and Kardes, O.: Privacy preserving data mining in the malicious model. International Journal of Information and Computer Security, 2(4), pp. 353-375, 2009.

[69] Karakasidis, A., and Verykios, V.S.: Privacy preserving record linkage using phonetic codes. Fourth Balkan Conference, Informatics. IEEE CS, pp. 101-106, 2009.

[70] Karakasidis, A., and Verykios, V.S.: Advances in privacy preserving record linkage. E-activity and Innovative Technology, Advances in Applied Intelligence Technologies Book Series. IGI Global, pp. 22-34, 2010.

[71] Karakasidis, A., and Verykios, V.S.: Secure blocking+secure matching = Secure record linkage. Journal of Computing Science and Engineering, 5(3), pp. 101-106, 2011.

[72] Karakasidis, A., and Verykios, V.S.: Reference table based k-anonymous private blocking. 27th Symposium on Applied Computing. ACM, pp. 859-864, 2012.

[73] Karakasidis, A., Verykios, V.S., and Christen, P.: Fake injection strategies for private phonetic matching. Data Privacy Management and Autonomous Spontaneus Security, LNCS 7122. Springer Berlin Heidelberg, pp. 9-24, 2012.

[74] Karapiperis, D., and Verykios, V.: An LSH-based Blocking Approach with a Homomorphic Matching Technique for Privacy-Preserving Record Linkage. IEEE Transactions on Knowledge and Data Engineering, preprints, doi:10.1109/TKDE.2014.2349916, 2014.

[75] Kelman, C.W., Bass, A.J., and Holman, C.D.J.: Research use of linked health data – A best practice protocol. ANZ Journal of Public Health, 26(3), pp. 251-255, 2002.

[76] Kirsch, A., and Mitzenmacher, M.: The power of one move: Hashing schemes for hardware. Networking, IEEE/ACM Transactions on, 18(6), 1752-1765, 2010.

[77] Kirsch, A., Mitzenmacher, M., and Wieder, U.: More robust hashing: Cuckoo hashing with a stash. SIAM Journal on Computing, 39(4), 1543-1561, 2009.

[78] Kroll, M., and Steinmetzer, S.: Automated Cryptanalysis of Bloom Filter Encryptions of Health Records. arXiv preprint arXiv:1410.6739, 2014.

[79] Kuzu, M., Kantarcioglu, M., Durham, E.A., and Malin, B.: A constraint satisfaction cryptanalysis of Bloom filters in private record linkage. Privacy Enhancing Technologies, LNCS 6794. Springer Berlin Heidelberg, pp. 226-245, 2011.

[80] Kuzu, M., Kantarcioglu, M., Inan, A., Bertino, E., Durham, E., and Malin, B.: Efficient privacy-aware record integration. Proceedings of the 16th International Conference on Extending Database Technology. ACM, pp. 167-178, 2013.

[81] Lai, P.K., Yiu, S.M., Chow, K.P., Chong, C.F., and Hui, L.C.: An efficient Bloom filter based solution for multiparty private matching. International Conference on Security and Management. CSREA Press, pp. 26-29, 2006.

[82] Lehman, E., and Panigrahy, R.: 3.5-way Cuckoo hashing for the price of 2-and-a-bit. Algorithms-ESA 2009, LNCS 5757. Springer Berlin Heidelberg, pp. 671-681, 2009.

[83] Li, Y., Tygar, J.D., and Hellerstein, J.M.: Private matching. Computer Security in the 21st Century. Springer US, pp. 25-50, 2005.

[84] Mayo Clinic website: http://www.mayoclinic.org/about-mayo-clinic/facts-statistics (accessed April 2015)

[85] Mohammed, N., Fung, B.C., and Debbabi, M.: Anonymity meets game theory: secure data integration with malicious participants. VLDB Journal, 20(4), pp. 567-588, 2011.

[86] Niedermeyer, F., Steinmetzer, S., Kroll, M., and Schnell, R. Cryptanalysis of Basic Bloom Filters used for Privacy Preserving Record Linkage. Journal of Privacy and Confidentiality, 6(2), Article 3, 2014.

[87] O'Keefe, C.M., Yung, M., Gu, L., and Baxter, R.: Privacy-preserving data linkage protocols. WPES'04. ACM, pp. 94-102, 2004.

[88] Oates, B.: Researching information systems and computing. SAGE Publications, 2006.

[89] Office of the Auditor General of Ontario: Ontario Health Insurance Plan report (accessed April 2015): http://www.auditor.on.ca/en/reports_en/en08/408en08.pdf

[90] Pagh, R., and Rodler, F.F.: Cuckoo hashing. Journal of Algorithms, 51(2), pp. 122-144, 2004.

[91] Paillier, P.: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. the International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT'99), LNCS 1592. Springer Berlin Heidelberg, pp. 223-238, 1999.

[92] Pang, C., Gu, L., Hansen, D., and Maeder, A.: Privacy-preserving fuzzy matching using a public reference table. Intelligent Patient Management, Studies in Computational Intelligence Volume 189. Springer Berlin Heidelberg, pp. 71-89, 2009.

[93] Pang, C., and Hansen, D.: Improved record linkage for encrypted identifying data. Proceedings of the 14th Annual Health Informatics Conference. HISA, pp. 164-168, 2006.

[94] Panigrahy, R.: Efficient Hashing with Lookups in Two Memory Accesses.16th Annual ACM-SIAM Symposium on Discrete Algorithms. ACM/SIAM, pp. 830-839, 2005.

[95] Peyton, L., and Hu, J.: Federated identity management to link and protect healthcare data. International Journal of Electronic Business, 8(3), pp. 214-232, 2010.

[96] Pommerening, K., Miller, M., Schidtmann, I., and Michaelis, J.: Pseudonyms for cancer registries. Methods of Information in Medicine, 35(2), pp. 112-121, 1996.

[97] Porter, E.H., and Winkler, W.E.: Approximate string comparison and its effect on an advanced record linkage system. Research Report RR97/02, U.S. Census Bureau, 1997.

[98] Pudjijono, A.: Probabilistic data generation. Thesis, Australian National University, Canberra, Australia, 2008.

[99] Pudjijono, A., and Christen, P.: Accurate Synthetic Generation of Realistic Personal Information. 13th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, LNCS 5476. Springer Berlin Heidelberg, pp. 507-514, 2009.

[100] Quantin, C., Bouzelat, H., Allaert, F.A.A., Benhamiche, A.M., Faivre, J., and Dusserre, L.: How to ensure data quality of an epidemiological follow-up: Quality assessment of an anonymous record linkage procedure. International Journal of Medical Informatics, 49, pp. 117-122, 1998.

[101] Quantin, C., Bouzelat, H., Allaert, F.A.A., Benhamiche, A.M., Faivre, J., and Dusserre, L.: Automatic record hash coding and linkage for epidemiological follow-up data confidentiality. Methods of Information in Medicine, Schattauer, 37(3), pp. 271-277, 1998.

[102] Quantin, C., Bouzelat, H., and Dusserre, L.: Irreversible encryption method by generation of polynomials. Medical Informatics and the Internet in Medicine, Informa Healthcare, 21(2), pp. 113-121, 1996.

[103] Randall, S.M., Ferrante, A.M., Boyd, J.H., Bauer, J. K., and Semmens, J.B. Privacy-preserving record linkage on large real world datasets. Journal of biomedical informatics, Vol. 50, pp. 205-212, August, 2014.

[104] Ravikumar, P., Cohen, W.W., and Fienberg, S.E.: A secure protocol for computing string distance metrics. PSDM held at IEEE ICDM, Brighton, UK, pp. 40-46, 2004.

[105] Resnik, D., and Koski, G.: A national registry for healthy volunteers in phase 1 clinical trials. JAMA; 305(12), pp. 1236-1237, 2011.

[106] Rogers, H.J., and Willett, P.: Searching for historical word forms in text databases using spelling-correction methods: reverse error and phonetic coding methods. J Doc, 47(4), pp. 333-353, 1991.

[107] Runeson, P., Host, M., Rainer, A., and Regnell, B.: Case study research in software engineering: Guidelines and examples. John Wiley & Sons, 2012.

[108] Sauleau, E.A., Paumier, J. P., and Buemi, A.: Medical record linkage in health information systems by approximate string matching and clustering. BMC medical informatics and decision making, 5(1), 32, 2005.

[109] Scannapieco, M., Figotin, I., Bertino, E., and Elmagarmid, A.K.: Privacy preserving schema and data matching. ACM SIGMOD. ACM, pp. 653-664, 2007.

[110] Schadow, G., Grannis, S.J., and McDonald, C.J.: Discussion paper: Privacy-preserving distributed queries for a clinical case research network. CRPIT'14: Proceedings of the

IEEE international Conference on Privacy, Security and Data Mining. IEEE CS, pp. 55-65, 2002.

[111] Schneier, B.: Applied cryptography: Protocols, algorithms, and source code in C, 2nd edition. John Wiley and Sons, Inc., New York (1996)

[112] Schnell, R.: An efficient privacy-preserving record linkage technique for administrative data and censuses. Statistical Journal of the IAOS: Journal of the International Association for Official Statistics, 30(3), pp. 263-270, 2014.

[113] Schnell, R., Bachteler, T., and Reiher, J.: Privacy-preserving record linkage using Bloom filters. BMC Medical Informatics and Decision Making, 9(1), 41, 2009.

[114] Song, D.X., Wagner, D., and Perrig, A.: Practical techniques for searches on encrypted data. IEEE Symposium on Security and Privacy. IEEE CS, pp. 44-55, 2000.

[115] Steorts, R. C., Ventura, S. L., Sadinle, M., and Fienberg, S. E.: A comparison of blocking methods for record linkage. Privacy in Statistical Databases, LNCS 8744. Springer International Publishing, pp. 253-268, 2014.

[116] Talburt, J., Zhou, Y., and Shivaiah, S.: SOG: a synthetic occupancy generator to support entity resolution instruction and research. Int. Conf. on Information Quality, Potsdam, Germany, pp. 91-105, 2009.

[117] The College of Physicians and Surgeons of Ontario Public Register Website: http://www.cpso.on.ca/Public-Register/Public-Register (accessed December 2014)

[118] The Law Society of Upper Canada Directory Website: http://www2.lsuc.on.ca/ LawyerParalegalDirectory (accessed December 2014)

[119] Tran, K.N., Vatsalan, D., and Christen, P.: GeCo: an online personal data generator and corruptor. Proceedings of the 22nd ACM international conference on information & knowledge management. ACM, pp. 2473-2476, 2013.

[120] Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths. NIST Special Publication 800-131A, 2011.

[121] Trepetin, S.: Privacy in context: the costs and benefits of a new deidentification method. PhD thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 2006.

[122] Trepetin, S.: Privacy-preserving string comparisons in record linkage systems: a review. Information Security Journal: A Global Perspective, 17(5-6), pp. 253-266, 2008.

[123] Tromp, M., Reitsma, J., Ravelli, A., Meray, N., and Bonsel, G.: Record linkage: making the most out of errors in linking variables. Proceedings of the American Medical Informatics Association Annual Symposium, pp. 779-783, 2006.

[124] United States Census Bureau website: http://www.census.gov/main/www/cen1990.html (accessed January 2015)

[125] Vaidya, J., and Clifton, C.: Secure set intersection cardinality with application to association rule mining. Journal of Computer Security, 13(4), pp. 593-622, 2005.

[126] Vaidya, J., Zhu, Y., and Clifton, C.: Privacy Preserving Data Mining. Advances in Information Security, Vol. 19, Springer Science+ Business Media, 2006.

[127] Van Eycken, E., Haustermans, K., Buntinx, F., Ceuppens, A., Weyler, J., Wauters, E., Van Oyen, H., DeSchaever, M., Van Den Berge, D., and Haelterman, M.: Evaluation of the encryption procedure and record linkage in the Belgian National Cancer Registry. Archives of Public Health, 58, pp. 281-294, 2000.

[128] Vatsalan, D., and Christen, P.: Sorted nearest neighborhood clustering for efficient private blocking. Advances in Knowledge Discovery and Data Mining, LNCS 7819. Springer Berlin Heidelberg, pp. 341-352, 2013.

[129] Vatsalan, D., Christen, P., O'Keefe, C. M., and Verykios, V.S.: An Evaluation Framework for Privacy-Preserving Record Linkage. Journal of Privacy and Confidentiality, 6(1), 3, 2014.

[130] Vatsalan, D., Christen, P., and Verykios, V.S.: An efficient two-party protocol for approximate matching in private record linkage. AusDM'11 Proc. Ninth Australasian Data Mining Conference - Volume 121, Australian Computer Society, pp. 125-136, 2011.

[131] Vatsalan, D., Christen, P., and Verykios, V.S.: A taxonomy of privacy-preserving record linkage techniques. Information Systems, 38(6), pp. 946-969. 2013.

[132] Vatsalan, D., Christen, P., and Verykios, V.S.: Efficient two-party private blocking based on sorted nearest neighborhood clustering. Proc. 22nd ACM international conference on information & knowledge management. ACM, pp. 1949-1958, 2013.

[133] Verykios, V.S., Karakasidis, A., and Mitrogiannis, V.K.: Privacy preserving record linkage approaches. International Journal of Data Mining, Modelling and Management, 1(2), pp. 206-221, 2009.

[134] Weber, S.C., Lowe, H., Das, A., and Ferris, T.: A simple heuristic for blindfolded record linkage. Journal of the American Medical Informatics Association, 19(e1), e157-e161, 2012.

[135] Wen, Z., and Dong, C.: Efficient Protocols for Private Record Linkage. SAC'14 Proceedings of the 29th Annual ACM Symposium on Applied Computing. ACM, pp. 1688-1694, 2014.

[136] Winkler, W.: Approximate string comparator search strategies for very large administrative list. Proceedings of the Section on Survey Methods. American Statistical Association, pp. 4595-4602, 2005.

[137] Yakout, M., Atallah, M.J., and Elmagarmid, A.: Efficient and practical approach for private record linkage. Journal of Data and Information Quality (JDIQ), 3(3), 5, 2012.

[138] Yakout, M., Atallah, M.J., and Elmagarmid, A.K.: Efficient private record linkage. IEEE ICDE. IEEE CS, pp. 1283-1286, 2009.

[139] Zanini, G.M., and Marone, C.: A new job: research volunteer? Swiss Med Wkly., 135(21-22), pp. 315-317, 2005.

Appendix A – Prevalence of concurrent enrollment in phase 1 clinical trials

Colleagues from the *Electronic Health Information Laboratory* have gathered the articles and references below to show the reality and severity of the concurrent enrolment problems in phase 1 clinical trials.

Boyar and Goldfarb [15] looked at a tracking system deployed by five clinical research sites in South Florida, USA. The system that was developed tracked patients using demographic and biometric identifiers. During the study period, 2081 research volunteers were registered. All participants were informed of the registry prior to enrollment. They found that 21.8% of the volunteers enrolled in a second study within the 18 month period, with 50 (2.4%) trying to enroll within 30 days of a previous trial and 186 (8.9%) within 60 days [15][105]. Researchers reported that study sites rejected an additional 10 to 20 subjects who tried to enroll in a second study before dosing commenced in the first trial. Furthermore, the researchers believed that knowledge of the registry may have discouraged some participants from attempting to enroll in overlapping trials.

Zanini and Marone [139] piloted a registry system that included all healthy volunteers participating in drug studies approved by the central Research Ethics Committee in Southern Switzerland within a three-year period. The total number of volunteers involved in clinical trials during this period was 1436. All trial participants were informed of the registry at the time of enrollment. The researchers identified 192 (13.4%) habitual or regular volunteers who participated in 6-8 successive trials during the period (with a mandated 30 day waiting period in between). Three subjects attempted to enroll in overlapping trials during the study period.

Hermann, et al. conducted a survey of healthy research volunteers in Germany [55]. Four hundred and forty people participated in the survey, 47.1% who were employees of the research centers and 52.9% who were outside volunteers. A small number of those surveyed, 2.7%, reported having considered participating in two studies within the restricted period of time (two

months), while 0.9% reported that they had participated in two overlapping trials. Also, 3% of participants reported that they had provided incorrect answers regarding their medical histories in order to be accepted by recruiters.

In preparation for developing our secure tracking protocol, we interviewed four phase 1 site managers and study coordinators for phase 1 trials to understand the issue of volunteers participating in multiple trials concurrently. During these interviews, we asked for the interviewees' estimate of the proportion of volunteers who attempt to enroll in multiple trials concurrently based on their experiences. The interviewees were recruited by sending invitations to ACRO (Association of Clinical Research Organizations), CRAC (Clinical Research Association of Canada), and CRO's (Contract Research Organizations) in Canada doing phase 1 trials contacted through an industry list. Our interviewees indicated that, based on their experiences, between 2% and 7% of screened volunteers may be participating in other trials.

Based on the articles and references above, we conclude that the dual enrolment in phase 1 clinical trials is indeed a real problem. In our experimentation, we used **2% as the prevalence value** between the dataset containing the participants that ask to join a trial and the central database that contains all the registered participants at the various clinical trial sites. Note that we experimented with various prevalence values between 1% and 10% and discovered that the accuracy of the protocol was higher if the prevalence value was higher so using the lower 2% value ensures that we are considering the lower part of the reported values and reporting the accuracy of our protocol properly and objectively.

Appendix B – TrialGuardian.com overview

We built a website in ASP.NET and C# to understand and demo to contract research organizations the usage of our protocol in a real-world scenario. The system can be run on an everyday computer able to host the Microsoft IIS server. We developed the system on Windows Vista on an Intel Centrino 2 computer with 4GB of RAM. This hardware is both cheap and accessible so it does not add much overhead to the cost of managing a trial. The following is the main page of a system to be deployed at a clinical trial site:

The system allows the user to add a study, view the studies at the clinical trial site and to modify the information about a study including the start date, end date, and washout period. For each study, the user can add participants, check the eligibility of a participant or modify the information about a participant. The *Alerts / Notifications* section informs the user of any dual enrollers as soon as they login to the system. The *Reports* section shows statistics about the studies and the participants and the dual enrollers.

After logging in, the user can select one of the options from the menu on the left. If the user selects to check the eligibility of a participant, the following page is displayed:

The user selects the study for which the participant wants to participate and fills in the participant's info (note that the shown participant information is **fake**; this is not a real person):

The user clicks on the *Check Eligibility* button and a popup dynamically displays the eligibility result to the user:

Appendix C – Online resources

In the following, we share several resources that we created throughout this thesis in order to assist other researchers to perform experiments and evaluations.

- Datasets: containing four fields (last name, gender, date of birth, health card number) with prevalence values between 1% and 10%
    - http://www.eecs.uottawa.ca/~damyot/pub/farah/datasets
- Frequency files: containing the name frequencies that can be used to generate simulated populations
    - http://www.eecs.uottawa.ca/~damyot/pub/farah/frequency
- Tools
    - DataRealtor
        - http://www.eecs.uottawa.ca/~damyot/pub/farah/datarealtor
    - ProEvalNet
        - http://www.eecs.uottawa.ca/~damyot/pub/farah/proevalnet