BMC Medical Informatics and Decision Making

BioMed Central

Technical advance

Some methods for blindfolded record linkage

Tim Churches*[1] and Peter Christen[2]

Address: [1]Centre for Epidemiology and Research, Population Health Division, New South Wales Department of Health, Locked Mail Bag 961, North Sydney NSW 2059, Australia and [2]Department of Computer Science, Australian National University, Canberra ACT 0200, Australia

Email: Tim Churches* - tchur@doh.health.nsw.gov.au; Peter Christen - peter.christen@anu.edu.au

* Corresponding author

Published: 28 June 2004

BMC Medical Informatics and Decision Making 2004, **4**:9 doi:10.1186/1472-6947-4-9

Received: 17 January 2004
Accepted: 28 June 2004

This article is available from: http://www.biomedcentral.com/1472-6947/4/9

Abstract

Background: The linkage of records which refer to the same entity in separate data collections is a common requirement in public health and biomedical research. Traditionally, record linkage techniques have required that all the identifying data in which links are sought be revealed to at least one party, often a third party. This necessarily invades personal privacy and requires complete trust in the intentions of that party and their ability to maintain security and confidentiality. Dusserre, Quantin, Bouzelat and colleagues have demonstrated that it is possible to use secure one-way hash transformations to carry out follow-up epidemiological studies without any party having to reveal identifying information about any of the subjects – a technique which we refer to as "blindfolded record linkage". A limitation of their method is that only exact comparisons of values are possible, although phonetic encoding of names and other strings can be used to allow for some types of typographical variation and data errors.

Methods: A method is described which permits the calculation of a general similarity measure, the *n*-gram score, without having to reveal the data being compared, albeit at some cost in computation and data communication. This method can be combined with public key cryptography and automatic estimation of linkage model parameters to create an overall system for blindfolded record linkage.

Results: The system described offers good protection against misdeeds or security failures by any one party, but remains vulnerable to collusion between or simultaneous compromise of two or more parties involved in the linkage operation. In order to reduce the likelihood of this, the use of last-minute allocation of tasks to substitutable servers is proposed. Proof-of-concept computer programmes written in the Python programming language are provided to illustrate the similarity comparison protocol.

Conclusion: Although the protocols described in this paper are not unconditionally secure, they do suggest the feasibility, with the aid of modern cryptographic techniques and high speed communication networks, of a general purpose probabilistic record linkage system which permits record linkage studies to be carried out with negligible risk of invasion of personal privacy.

Background

Introduction

The ability to link records in separate data collections which refer to the same entity is a common requirement in public health and biomedical research [1]. Where records in separate data collections share a common,

unique entity key or identifier, such linkage is easy to perform. Where a common key is not available, the task is more difficult, and typically an ensemble of non-unique, partially-identifying attributes such as name, date of birth and address are used to probabilistically infer which records refer to the same entity. Often these attributes are time-variant (for example, residential address) or recorded with some degree of error.

In order to carry out record linkage using traditional methods, it is necessary for someone (or some organisation) to be given access to the ensemble of non-unique identifying attributes for all of the relevant records in the data collections to be linked. Good practice dictates that medical and other substantive attributes should be removed from the records before passing them to a person or organisation undertaking the record linkage operation (the record linker). A number of protocols for achieving this have been described [2-4]. A defect of these protocols is that they do little to obfuscate the source of those records. In many circumstances knowledge of the data source permits significant and highly confidential information to be inferred about individuals who are identified in the candidate records to be linked. Furthermore, the record linker necessarily requires access to all relevant records in all the data collections to be linked because there is no way of knowing prospectively which records will match – thus these methods require the disclosure of confidential information about large numbers of individuals, albeit to a small number of people who actually undertake the linkage.

For example, a research project may need to determine which individuals who appear in a population-based register of people suffering from hepatitis C also appear in a separate population-based register of cases of hepatocellular carcinoma (which may be part of a wider cancer registry). If the research project requires that individuals whose records are present in both data collections be contacted or followed up, then the researcher clearly needs to know their identities. If no follow-up is planned, then the researcher need not know their identities. Either way, the identities of all individuals in each of the two data collections, many or most of whom will not appear in both, need to be provided to the person or organisation performing the record linkage. This disclosure of identities represents a loss of privacy for all of the individuals on both databases, regardless of whether they ultimately become subjects of the study in question. The loss of privacy can be minimised by strict observance of procedures designed to maintain confidentiality, but it cannot be eliminated entirely when conventional record linkage techniques are employed. In addition, there remains a small but finite risk that the linkage operation might be compromised by external attackers, by errant internal staff or even through administrative or political malfeasance. The consequences of compromise – such as misappropriation and publication of the names and addresses of individuals on the Internet – are potentially devastating and, although unlikely, must nevertheless be contemplated.

Related work

One method of mitigating such risks is to establish a multi-party system in which all linkage is undertaken by a central linkage bureau which is supplied with only the identifying attributes of individuals. The sources of these partially identifying attributes are hidden by passing them through a proxy. Messages containing identifying information from many sources are mixed by the proxy and delayed by random amounts before being forwarded to the central linkage bureau, thus obfuscating the source of each message. Such a framework is described in detail by Churches [5]. If such frameworks are used, a compromise of the central linkage bureau is much less catastrophic because no information is held about the source of the names, addresses and other partially identifying data items in its database – instead, this information is held by the proxy. Simultaneous compromise of both the record linkage agency and the proxy would be required for a serious breach of confidentiality to occur. However, it may still be possible for the centralised linkage bureau (or an attacker who gains access to the data held by the linkage bureau) to infer information about the source of each message. This could be done through analysis of missing data items or of the way in which data items are formatted – both of which may be characteristic of particular data sources. In addition, such systems require many parties to participate in order to be successful, and participation requires adaptation of existing information systems. Thus, they are not easy to establish.

Given these problems, methods which allow records in separate data collections to be linked for the purposes of specific research projects, without requiring identifying information to be revealed to anyone who does not already have access to it, are desirable.

One approach is to severely limit the identifying data items which are disclosed to the record linker – an approach which is often termed "anonymous record linkage" [6]. This approach has the disadvantage that as the number and details of the (partially) identifying data items which are disclosed to the record linker are reduced, the specificity and overall efficiency of the linkage operation are also diminished. Of course, truly anonymous data does not contain sufficient partially identifying information to permit any useful record linkage, by definition. Thus, it may be better to call such approaches "partially-identified record linkage", or perhaps, "partially de-identified record linkage".

However, methods are available which allow the party undertaking the record linkage operation to use all of the partially-identifying data items, such as names and dates of birth, which are available from the data collections being linked, but without the need to know the actual values of those data items. These methods, which use keyed one-way hash functions, were described in the mid-to-late 1990s by a team of French researchers [7-13]. These researchers have also described their method as "anonymous record linkage", but the approach is quite different to the aforementioned usage. We feel that "blindfolded record linkage" is a better term, because a number of identifying data items are, in fact, used in the linkage but the party undertaking the linkage is unable to see their values. The adjective "blindfolded" rather than "blinded" is suggested in order to remind users that such methods still assumes that two or more parties involved in the linkage operation will not collude in order to "remove the blindfold". However, unlike traditional record linkage techniques, the methods provide good protection against a single party, acting alone, attempting to invade privacy or breach confidentiality. In this respect the qualifier "blindfolded with both hands tied behind the back" is more accurate, but perhaps rather too wordy.

The French methods target situations in which de-identified patient data needs to be centralised and linked for the purposes of longitudinal (follow-up) studies. Schadow *et al*. have subsequently described a related method for using keyed hash functions to carry out on-the-fly, distributed, deterministic record linkage, that is, without any requirement to centralise data [14]. There is also intense interest in the knowledge discovery and data mining community in "privacy-enhanced" data mining and "secure multi-party computation", a field first introduced by Yao in 1982 [15]. Although it appears that almost any function can be computed securely without revealing its inputs, all of the extant protocols do so at the expense of communications efficiency (as do the protocols proposed in this paper). Hirt *et al*. provide a brief review [16].

In this paper we will first describe the means used in the French protocols to carry out blindfolded equality testing of both normalised and phonetically-encoded strings. We will then describe a new method for performing blindfolded similarity comparisons between strings or other short sequences. We will illustrate how these methods can be used with asymmetrical (public key) encryption to create blindfolded record linkage protocols suitable for various research scenarios.

Many of the assumptions and terminology used in this paper are borrowed from the discipline of information system security engineering, which is concerned with "...building systems to remain dependable in the face of malice, error, or mischance" [17]. This is a discipline which has been likened to "programming Satan's computer" [18]. In particular, there is a deliberate attempt to avoid the unrealistic assumptions that individuals or organisations will always behave as they ought, or as expected, and that computer security mechanisms are foolproof and will always function perfectly. Where such assumptions are unavoidable, they are identified explicitly and the probability and impact of violations evaluated. The names Alice, Bob and so on are traditionally used in the cryptology literature to refer to parties who participate in transactions or data operations. It should be born in mind that these parties may represent the rightful or intended recipients or custodians of some data, but could also represent an external or internal attacker who has gained unauthorised access to that data. Thus, it is not only the fairly remote possibility that a researcher or computer system administrator might behave unethically (or be induced to do so) which must be considered, but also that untrustworthy parties may gain unauthorised access to Alice's or Bob's data.

The following protocol descriptions assume that data communication between parties is secured through the use of public key cryptography [19,20]. Public key cryptography relies on the difficulty of factoring the key to produce pairs of very large prime numbers to encrypt data using a pair of complementary keys belonging to each party wishing to exchange information securely. These keys are known as the public key and the private key. The public key is published and can be used by anyone wishing to encrypt information in such a way that it can only be decrypted (read) by the holder of the matching private key, and by no-one else. In practice, for reasons of computational efficiency, public key encryption and decryption algorithms are used to pass random "session keys" securely between parties and these session keys are used with conventional encryption algorithms to protect the actual data. The effect is the same as if the entire message were encrypted or decrypted using public or private keys. Each party's private key can also be used to digitally "sign" messages to prove to the recipient that the party sending the message is in fact whom they claim to be and that the message has not been altered during the transmission process. Usually a trusted agency known as a certificate authority handles the distribution of public keys and vouches for the authenticity of these keys and the *bona fides* of the parties to whom they belong. Collectively these technologies are often referred to as "public key infrastructure" (PKI).

Minimal-knowledge value comparisons

The fundamental operation at the core of all forms of record linkage is the comparison of partially-identifying data items from two (or more) data sources. Typically

these values are character strings (such as names, or address components), but they may be scalar quantities such as age or date of birth. The comparison may be as simple as determining whether the values match exactly, or it may involve some form of similarity comparison, such as the Levenshtein edit distance [21], or the bigram similarity function, which is discussed in more detail below.

Usually, the values being compared must be known to the person or machine undertaking the comparison. However, cryptographically secure one-way hash functions can be used to allow values to be tested for equality while hiding the actual values from the entity performing the test.

A one-way hash function, such as the Secure Hash Algorithm (SHA) [22], transforms a string or other series of bytes into a new, fixed-length string (a hash value) in such as way that it is impossible, or at least extremely difficult, to discover the value of the original string from the hash value. A given source string will always produce the same hash value (assuming that little-endian/big-endian differences are accounted for, as they usually are), and the probability that two different source strings will produce the same hash value (a "hash collision") is approximately $1.2\sqrt{n}$ where n is the number of possible hash values. The SHA algorithm produces a 160 bit hash value, which means that the likelihood of hash collisions is still very small – approximately 1 in 10^{24} operations. Even tiny changes in the original string cause major changes in the resulting hash value – with the SHA algorithm change of a single character will result in at least half of the bits in the resulting hash value changing [23].

Problem statement and principal actors

Before setting out any protocols, it is useful to define the problem statement and principal actors.

Alice holds a data collection (database), A, which contains one or more attributes (also often referred to as data elements, variables, fields or columns), denoted A.a, A.b and so on, each, containing confidential character strings (such as names) or some other sequences or values.

Bob holds a similar but quite separate data collection, B, also containing one or more confidential attributes, B.a, B.b and so on.

Alice and Bob wish to determine whether any of the values in A.a match any of the values in B.a without revealing to each other or to any other party what the actual values in A.a and B.a are.

Carol is a third party who is trusted by Alice and Bob to perform the processing specified in the protocols described in this paper, and not to divulge the data which she receives from Alice and from Bob to any other party, including Alice and Bob.

Basic protocol

In order to perform a minimal-knowledge value comparison using one-way hash functions, Alice and Bob first agree on a particular secure hash function to use. There is a limited number of such functions to choose from, and in most circumstances the SHA algorithm would be used. Alice then computes the hash of a value in her data collection and sends it to Carol. Bob does the same. Carol then compares the two hash values which she has received. If they are the same, then the source values in Alice's and Bob's databases must match.

Extension to comparison of sets of values is straightforward – all that is required is some secure means of associating each hash value with the original source values in Alice's and Bob's data collections. To do this, Alice and Bob could attach an arbitrary, random ID number (a nonce) to each hash digest being compared.

Although Carol is unable to reverse the hashing process and discover the original values, she is able to mount a dictionary attack against the list of hashes which she receives from Alice and Bob [24]. For example, Carol may know (or guess) that the hash values she is given are derived from surnames. If she computes the hashes for all surnames in an exhaustive list (for example, derived from a telephone book or electoral roll), and compares these with the hashes she has received from Alice and Bob, she may be able to discover the original values. In order to thwart such attacks by Carol, Alice and Bob need to agree on a secret key (ideally a random value chosen from a large key space) which is used to transform the source values in an agreed manner before they are hashed. The HMAC (Hashed Message Authentication Code) algorithm provides a suitable transformation which is both well studied and widely supported by mature software implementations [25]. We will refer to the resulting value as a "keyed hash digest" or simply a "digest".

It should be noted that even when keyed hash digests are used to thwart a dictionary attack, there is still a theoretical risk that Carol may be able to infer some information about the source data values by mounting a statistical attack in which she examines the relative frequencies of the hash digests which she receives. For such an attack to succeed, large numbers of digests must be available to Carol, and she must know or be able to guess a great deal about the frequency distribution of values in the source data collection, and some of those values need to be rare. The risk of a successful statistical attack against the keyed hash digests increases if Carol has access to digests derived

from more than one data item (such as surname, given name, residential street address and suburb) so that she is able to correlate the relative frequencies of the hash digests for a particular record. Mechanisms for avoiding this situation, such as "chaffing" as suggested by Rivest [26], are described later in this paper.

Extension to phonetically-encoded character strings

If a value in Alice's data collection is "victoria" and a value in Bob's data collection is "Victoria ", then the resulting keyed hash digest will be different and the comparison will fail. Thus, Alice and Bob must also agree on a set of pre-processing rules and transformations to render the values to be compared as alike as possible prior to the application of the keyed hash transformation.

The values to be compared may also contain errors – for example, the value in Bob's data collection may be "victora". A phonetic encoding function such as Soundex or Metaphone may be used to make the match between the two strings more robust to spelling mistakes and typographical errors, at the expense of a greater number of false matches (homonym errors) [27,28].

However, Soundex, Metaphone and other phonetic transformations are not perfect – in particular they are not robust with respect to errors in the initial character, or to truncation differences. For example, "Christopher", "Christine" and "Cristina" all have a Soundex code of C623 whereas "Chris" has a code of C620; and "Kristine" has a code of K623. In some cases it may be possible to mitigate these problems by creating an array of alternative digest values for known alternatives, such as "Liz" and "Beth" for "Elizabeth", in an attempt to get a match, but there are limits to the generality of this approach and again the rate of "false" matches (homonym errors) will inevitably increase.

Ideally, a protocol is required which permits the blindfolded calculation by Carol and/or other trusted third parties of a more general measure of similarity between the pairs of secret strings.

n-*gram similarity comparators*

The method proposed here involves the use of an *n*-gram similarity score, which is often referred to as the Dice coefficient in the information retrieval field [29,30]. An *n*-gram (also sometimes called an *n*-graph) is the set of sub strings of length *n* in a word string. For example, the 2-grams (bigrams) in the word "peter" are "pe", "et", "te" and "er". The Dice co-efficient is defined as

$$Dice\,co-efficient = 2\times\left(\frac{|\,bigrams(x)\cap bigrams(y)\,|}{|\,bigrams(x)\,|+|\,bigrams(y)\,|}\right)$$

where bigrams() is a function which reduces a word to its set of bigrams, and x and y the two character strings to be compared. The Dice co-efficient (or bigram score) for "peter" and "pete" is $2 \times 3/(4 + 3) \cong 0.86$ because "pete" contains three bigrams, "peter" contains four, and they have three bigrams in common. Bigram scores are not reliable and thus generally not calculated if one of the strings contains fewer than two bigrams. Unigram scores can be used in such circumstances.

In order to be able to compare hashed values of different bigram sets (e.g. ("pe", "et", "te", "er") and ("pe", "et", "te")), an exhaustive set of subsets (that is, the power set of each bigram set) is calculated, as described in more details below (Protocol 1, step 3). For the example above, the comparison of hashed values would result in the hashed-value for the bigram set ("pe", "et", "te") to match, but not the hashed values of bigram sets ("pe", "et", "te", "er") and ("pe", "et", "te"). A worked example can be found in Tables 1 and 2.

Porter and Winkler examined the effects of various string similarity comparators on the performance of a modern probabilistic record linkage system [31]. They concluded that although information-theoretic comparators such as the Jaro comparator gave slightly better performance, the Dice bigram co-efficient still produces satisfactory results [32].

For short strings, Dice's co-efficient can also be calculated for unigrams (single characters). Extension to higher-order *n*-grams and other variations are also possible. For example, Brew and McKelvie found that addition of "extended bigrams", which are formed by deleting the middle letter from each three letter sub string of a word, decreased the sensitivity of the Dice bigram co-efficient to transpositions and substitutions in shorter words [33].

Methods

A protocol for minimal-knowledge n-*gram similarity comparisons*

In the following description, bigrams (2-grams) are used, but the extension to unigrams (single letters), trigrams and higher order *n*-grams is direct. We will refer to this protocol as Protocol 1.

As previously, Alice and Bob wish to use the services of Carol to determine whether any of the values in A.a match any of the values in B.a without revealing to each other or to Carol what the actual values in A.a and B.a are. Note that there may be more than one instance of Carol, each used to handle one of the attributes which are common to both Alice's and Bob's data collections. Each such instance of Carol is functionally equivalent, but they do not (and should not) communicate with each other.

Table 1: Example encrypted tuple for value "peter" created by Alice. For illustrative purposes the record key as well as the bigram subset values are shown unencrypted. The tuple matching with Bob's value "pete" (with highest bigram score) is shown bold-faced (see also Table 2).

A record key	*A.a bigram subset*	*A.a_bigram_combination_digest*	*A.a_bigram_ combination_length*	*A.a_length*
10	('er')	0a3be282870998fe7332ae0fecff68cc0d370152	1	4
10	('et')	8898f53d6225f464bb2640779cb17b9378237149	1	4
10	('pe')	6fc83a87ee04335a58aa576cb5157625b1b5c51b	1	4
10	('te')	f2bcfb3d76d7fc010e3adc08663090f29c5e928a	1	4
10	('er', 'et')	f86abb0c84889d004b817e86199b3837708d70e9	2	4
10	('er', 'pe')	df99d8658d8165af4552f60ade3662ba98006298	2	4
10	('er', 'te')	edfb618d37ecfafc9735e6ad4675245a4071aa9d	2	4
10	('et', 'pe')	bd7ada000c2b9004b7519b989bfcfdff7ad36678	2	4
10	('et', 'te')	fdcb71db96d2da9b1d19b62944c5f36448cb2668	2	4
10	('pe', 'te')	71322eeebabff9828aeed3281a86577163e16a78	2	4
10	('er', 'et', 'pe')	8bf2788ef28443b7a0298f19defa5532db40f63a	3	4
10	('er', 'et', 'te')	c7e9a32e54ba33d3769c4813616fdfcc6306459c	3	4
10	('er', 'pe', 'te')	33287ce86aa02af0f31d4857a79671c1f4645277	3	4
10	**('et', 'pe', 'te')**	**ecd7b15129lf1612595c9f8f385e9f71119a1ae0**	**3**	**4**
10	('er', 'et', 'pe', 'te')	65e568493a08a3428595b8be35f6ae2a0f48d170	4	4

Table 2: Example encrypted tuple for value "pete" created by Bob. For illustrative purposes the record key as well as the bigram sub set values are shown unencrypted. The tuple matching with Alice's value "peter" (with highest bigram score) is shown bold-faced (see also Table 1).

B record key	*B.a bigram subset*	*B.a_bigram_combination_digest*	*B.a_bigram_ combination_length*	*B.a_length*
42	('et')	8898f53d6225f464bb2640779cb17b9378237149	1	3
42	('pe')	6fc83a87ee04335a58aa576cb5157625b1b5c51b	1	3
42	('te')	f2bcfb3d76d7fc010e3adc08663090f29c5e928a	1	3
42	('et', 'pe')	bd7ada000c2b9004b7519b989bfcfdff7ad36678	2	3
42	('et', 'te')	fdcb71db96d2da9b1d19b62944c5f36448cb2668	2	3
42	('pe', 'te')	71322eeebabff9828aeed3281a86577163e16a78	2	3
42	**('et', 'pe', 'te')**	**ecd7b15129lf1612595c9f8f385e9f71119a1ae0**	**3**	**3**

Protocol 1 assumes that Carol is trusted by Alice and Bob to a) adhere to the protocol; b) not reveal information to other parties except where permitted by the protocol – in other words, not to collude with either Alice or Bob against the other, and; c) not try to determine the values of Alice's or Bob's source strings using cryptanalytic techniques. The effects of violations of these assumptions are considered below. There is no assumption that Alice trusts Bob or *vice versa*. Alice and Bob do need to share with each other metadata about the nature of the information contained in their databases – in order to decide which attributes can be validly compared – but they do not need to share the actual values of those attributes, nor summary measures (such as frequency counts) derived from those values. Alice and Bob do not need to share metadata with Carol.

In the following description, a "set" refers to a collection of items which has no inherent order and in which each item is unique, and a "list" is a sequence of items which have an inherent order and which are not necessarily unique. A "tuple" is a collection or set containing two or more values [34]. It is assumed that Alice and Bob process the records in their respective data collections in a random order in order to prevent any information which might be contained in the record order from inadvertently being transmitted to Carol or other parties. It is also assumed that each record in Alice's and Bob's databases has a unique record identifier (key).

Protocol 1 consists of the following steps:

1. Alice and Bob mutually agree on:

• A secret random key, K_{AB}, which they share only with each other. For example, Alice might generate K_{AB} using a random number generator operating over a large number space, and then transform the result into an SHA hash value which she encrypts with Bob's public key before sending it to him. This requires that Bob trusts Alice not to re-use the same key. Alternatively, Alice and Bob could arrive at a shared, random key by repeated use of the "honest coin flip" protocol or, more efficiently, via the Diffie-Hellman protocol [35-37].

• A particular keyed hash transformation function – this would usually be the HMAC algorithm.

• A standard protocol for pre-processing strings to render them in a standard form (such as converting all characters to lower case, removal or substitution of punctuation and extraneous white space, and so on).

• A mechanism for adding "chaff" which to Carol (or an attacker) is indistinguishable from real data, but which subsequently can be "winnowed" by Alice and Bob. Rivest describes a suitable mechanism [38]. The purpose of such "chaffing" is to thwart a statistical attack against the keyed hashes sent to Carol.

2. Alice pre-processes the values in attribute A.a in the agreed manner and computes the set of bigrams for each of the resulting values. For example if the value of $A.a_1$ is "peter" then the set of bigrams is ("pe","et","te","er").

3. Alice then computes the power set (i.e. the subsets of the original bigram set) of each set of bigrams. Using the subsets (key hashed as described below) will allow for partial matching between values in Alice's and Bob's databases. The power set of ("pe","et","te","er") is as follows:

(), ("pe"), ("et"), ("te"), ("er"),

("pe","et"), ("pe","te"), ("pe","er"), ("et","te"), ("et", "er"), ("te","er"),

("pe","et","te"), ("pe","et","er"), ("pe","te","er"), ("et","te","er"),

("pe","et","te","er")

4. Each set in the power set is converted into a sorted list, with the exception of the null (empty) set, which is discarded. Each of these sorted lists are then transformed using the agreed keyed hash algorithm and key K_{AB}. The resulting vector of digests is stored in an attribute called A.a_bigram_combination_digest.

5. Using a secret key, K_A, known only to herself, Alice creates keyed hash digests of the record identifiers (keys) for strings from which each value in attribute A.a_bigram_combination_digest was derived – she stores these in A.record_key_digest. She also places the count of bigrams in each combination used to create each A.a_bigram_combination_digest in an attribute called A.a_bigram_combination_length, and the count of bigrams in each of the original values in A.a in an attribute called A.a_length.

6. Alice then creates a set of tuples of the form (A.record_key_digest, A.a_bigram_combination_digest, A.a_bigram_combination_length, A.a_length), encrypts this set using Carol's public key, and sends the result to Carol. An example of tuples created for the value "peter" is given in Table 1 (partially unencrypted for illustrative purposes).

7. Alice sends Carol a list of A.record_key_digest values for those records in which the value of a, the attribute in question, is missing or otherwise unavailable.

8. Bob also carries out steps 2 to 7 using his data collection attribute B.a, and sends the encrypted set of tuples to Carol. An example of tuples for the value "pete" is given in Table 2 (partially un-encrypted for illustrative purposes).

9. Carol performs an inner equi join of the tuples, using a_bigram_combination_digest as the join key – in other words, she determines the intersection of A.a_bigram_combination_digest and B.a_bigram_combination_digest. For each row in the inner join product, Carol calculates the bigram_score (the Dice co-efficient), which is 2 × A.a_bigram_combination_length / (A.a_length + B.a_length), and selects the maximum bigram_score for each unique tuple (A.encrypted_record_key, B.encrypted_record_key) – in other words, the highest bigram_score for each pair of records from A and B which share some bigrams in common.

10. Carol reports the similarity scores (including a predefined value used to represent comparison with a missing value) and the respective encrypted record keys to another third party who uses the information as input to a blindfolded record linkage procedure which involves the minimal-knowledge comparison of a number of different attributes from Alice's and Bob's data collections – that is, blindfolded similarity comparisons of not only A.a and B.a, but also A.b and B.b, A.c and B.c and so on, with each attribute comparison undertaken by a different instance of Carol. This will be described in more detail below.

Results

Efficiency considerations

The first question to ask is whether the above protocol is feasible, given the combinatorial explosion in data volume caused by having to generate the power set of bigrams for the values being compared: a string which contains x bigrams gives rise to 2^x-1 keyed hash values. For example, instead of simply having to send a string value "peter" (5 characters), the hashed values (160 bits or 20 bytes each, assuming we use SHA) for the 15 bigram subsets (shown in step 3 above) have to be communicated

from Alice to Carol (in total 300 bytes). Additionally, the hash key values for the A.record_key_digest, and the values of A.a_bigram_combination_length and A.a_length need to be communicated as well.

This requirement places a practical limit on the length of strings which can be compared, and makes the method unsuitable for use with long sequences such as those used in genomics or proteomics. However, we believe that using this method with typical name and address character strings is quite feasible using high speed research networks which are intended for the efficient transmission of large volumes of data.

To test this, we derived lists of approximately 2.3 million surnames and suburb names from the residential telephone directories for the Australian state of New South Wales (NSW), which has a population of approximately 6.5 million, with a large, polyglot immigrant component. The mean surname and suburb name lengths were 6.4 and 9.3 characters respectively, giving rise to an average of 166 and 2,521 bigram subsets. Each bigram subset is represented by a 160 bit keyed hash value (using SHA), which means that the hashed values of all the bigram subsets occupy approximately 520 and 5,400 times more space than the original surname and suburb values, respectively. Although this means that the protocol is very inefficient from a data transmission perspective, the degree of data inflation is not so great as to render it infeasible when used with small- to medium-sized data collections on dedicated research networks.

One method for improving the efficiency of the protocol is to avoid creating shorter bigram subsets. In general, we will only be interested in pairs of values whose bigram score is greater than some threshold, say 0.7. Alice does not know the number of bigrams in each of Bob's values, but she does know the number of bigrams in each of her own values. She also knows that as the number of bigrams in Bob's string decreases, her string needs to have fewer bigrams in common with it in order for the resulting bigram score to be above a given threshold. The shortest of Bob's values will have (by definition) a bigram length of 2. Substituting these values into the equation for the bigram score, Alice can calculate the minimum number of bigrams which her value must have in common with Bob's, regardless of the length of Bob's value. Thus, for each of her values, Alice can limit the bigram subsets which she produces to those which have a length equal to or greater than the minimum length required to reach a bigram score of 0.7 (or some other arbitrary cut-off). For example, for a value "peter" – with bigrams ("pe","et","te","er") – at least three bigrams need to be in common with another value in order to get a bigram score larger than 0.7. The bigram score for comparing, for example, "peter" with "pet" – with bigrams ("pe","et") – will be $2 \times 2/(4 + 2) \cong 0.66$. Thus, for value "peter", Alice only needs to communicate bigram subsets of lengths 3 and 4 to Carol (i.e. 5 out of 15 bigram subsets).

Using the same NSW telephone directory data, the reductions in the overall overhead (that is, communication volume requirements compared to the original, unencrypted, surname and suburb values) for thresholds between 0.0 and 1.0 are shown in Table 3.

Security weaknesses in Protocol 1, and some remedies

Protocol 1 meets its design aims of blindfolding Carol to the values of Alice's and Bob's secret character strings, while still permitting her to calculate a widely-used measure of similarity between those strings. The keyed hash digests of the bigram combinations which are sent to Carol do not, *per se*, carry any information about the secret strings from which they were derived. Nevertheless, further consideration of the protocol in the light of two obvious threat models is warranted.

The first threat is the possibility that Carol might mount a frequency analysis attack against the keyed hash digests which she receives. Frequency analysis, which was first described by the ninth-century Arab scientist al-Kindi, involves the comparison of the relative frequency of each symbol (in this case, bigram combination hash values) with the relative frequencies of equivalent symbols derived from data which is likely to have a similar frequency distribution of values to the secret source data [39]. For example, if a particular column contains the surnames of patients drawn from a known population, then the frequency distribution of their values is likely to be somewhat similar to the distribution of, say, surnames contained in a telephone directory for that same population. This similarity can be used to obtain some information about the original values from which hash values are derived. Such an attack is more likely to succeed if Carol receives a large number of digests, thus providing her with better statistical power. In the case of Protocol 1, Carol has access to information about the number of bigrams which each hash digest represents, as well as the number of bigrams in the original strings. In addition, she can easily determine which bigram hash values were derived from the same original string, because a hashed record key is provided as part of each tuple which she receives from Alice and Bob. Taken together, this information provides Carol with a large number of clues or "cribs" (to use the cryptanalytic term) to assist her attack. It should be noted that the one-way hashing of original values or phonetic transformations thereof, as described by Dusserre, Quantin, Bouzelat and colleagues, is also vulnerable to such frequency analysis [7-13].

Table 3: Keyed hash bigram subset communication volume. Overhead for various minimum bigram score thresholds compared to the unencrypted communication of the original values. The average (unencrypted) surname and suburb name lengths were 6.4 and 9.3 characters, giving rise to an average of 166 and 2,521 bigram subsets respectively. A total of 2,323,355 records were processed.

Minimum bigram score threshold	*Surnames*		*Suburb names*	
	Megabytes communicated	Overhead	Megabytes communicated	Overhead
0	7540	520	114399	5435
0.1	7540	520	114384	5435
0.2	7484	516	113787	5406
0.3	7132	492	109679	5211
0.4	6287	434	97300	4623
0.5	4238	292	63892	3036
0.6	2836	196	35091	1667
0.7	1242	86	12154	577
0.8	511	35	3056	145
0.9	185.0	12.7	442.4	21.0
1	45.4	3.13	45.4	2.16
Original values	14.5	1	21.1	1

One response to this threat is for Alice and Bob just to trust Carol not to undertake such cryptanalytic attacks. This means that not only must Carol be trusted by Alice and Bob, but also she must implement mechanisms to protect the data to which she has access from misuse by both insiders (Carol's own staff) and external attackers. Protection against the former is particularly difficult. Fortunately, the processing which Carol is required to carry out can be both amnestic and completely automated. All data can be manipulated entirely in volatile random access memory and never needs to be written to any form of persistent storage for all but the largest data sets. If implemented on a secure operating system which provides mandatory access control, such as NSA Security Enhanced Linux [40], it should be possible to make it extremely difficult for even insiders to misappropriate Carol's data or suborn her processing of it. It is worth noting that a frequency-based cryptanalytic attack against the hash digest and bigram length data to which Carol has access would still require considerable skill and resources, for rather uncertain gains, and thus an attacker would need to be highly motivated and well resourced – but it nevertheless remains a theoretical possibility.

An alternative response is to use further separation of duties in order to reduce the amount of information available to Carol for use in a frequency-based cryptanalytic attack. Protocol 2, described below, sets out how this can be achieved. Additionally, "chaffing" can be used to obfuscate the frequency information contained in the hash digest values. This is also discussed further below, in the context of Protocol 2.

The other threat which needs to be considered is that of Alice colluding with Carol in an attempt to discover Bob's secret values, or vice versa. If Carol passes the information given to her by Bob on to Alice, then it is straightforward for Alice to mount a dictionary attack against Bob's keyed hash digests and thus discover the underlying bigram combinations. With these in hand, reconstruction of Bob's original, secret values is trivial.

For these reasons, it is important that Carol, her staff and her data processing system are as trustworthy as possible, and that requests to collude from Alice or Bob are refused. Of course, it is unwise to assume that Carol or her staff will always behave correctly, or that her processing system is eternally secure. Thus mechanisms for limiting the consequence of Carol colluding with Alice or Bob need to be considered.

In the context of record linkage, one method for achieving this is to employ multiple, independent instances of Carol, each one receiving and processing information derived from only one attribute of Alice's and Bob's data collections. In that way, a security failure or collusion involving one of these instances of Carol would result in the compromise or disclosure of only a single data element. Provided that the compromised attribute was not a unique or near unique identifier (such as a social security number), the risk that individuals might be identified is quite small.

Another method for reducing the risk of collusion is to allocate specific tasks to instances of Carol and other parties only at the very last moment. This is explored in more detail below. However, we will first describe Protocol 2.

An improved protocol for minimal-knowledge n*-gram similarity comparisons*

In this variation, we introduce a fourth party, David, who, like Carol, assists Alice and Bob in carrying out blindfolded *n*-gram similarity comparisons of their secret strings. Carol and David are required to not exchange data with each other except as set out in the protocol. If they do collude they are able to assemble the same information as is available to Carol in Protocol 1 – in other words, they would still need to mount a frequency-based cryptanalytic attack in order to discover anything about Alice's or Bob's original secret values.

Protocol 2 consists of the following steps:

Steps 1 to 4 are identical to Protocol 1.

5. Alice prepares a set of tuples comprising (A.a_bigram_combination_digest, $\{(R_{rand}$, A.record_key_digest, A.a_bigram_combination_length, A.a_length)$\}_{PublicKeyD}$), where R_{rand} is an arbitrary random number and $\{...\}_{PublicKeyD}$ denotes that the contents are encrypted with David's public key.

6. Alice encrypts this set of tuples with Carol's public key and sends the result to her.

7. Bob also carries out steps 1 to 6, using the values in his attribute B.a, and also sends the result to Carol.

8. Carol determines which values in the attribute A.a_bigram_combination_digest also appear in B.a_bigram_combination_digest – in other words, the intersection of the two attributes of keyed hash digest values. Carol then sends David a set of tuples which correspond to the intersecting bigram combination keyed hash digest values, each tuple comprised of ($\{(R_{rand}$, A.record_key_digest, A.a_bigram_combination_length, A.a_length)$\}_{PublicKeyD}$, $\{(R_{rand}$, B.record_key_digest, B.a_bigram_combination_length, B.a_length)$\}_{PublicKeyD}$).

9. Using his private key, David decrypts the bigram length values and calculates a bigram score for tuples he has been sent. The value R_{rand} is ignored. David then determines the maximum bigram score for each unique pairing of A.record_key_digest and B.record_key_digest.

In Protocol 2, the only information which David is able to glean about Alice's and Bob's original values is a frequency distribution of the lengths of those values. No information about the frequencies of the actual values is available to him. In addition, Carol has much less information which she can use to mount a frequency attack against the bigram hash digests which she receives. Nevertheless, information about the frequency of each bigram digest is still available to Carol. This information can be removed by hiding Alice's and Bob's real data amongst dummy data, analogous to the practice of homophonic substitution used to prevent frequency analysis of monoalphabetic ciphers, and related to the technique of "chaffing and winnowing" suggested by Rivest [41].

For example, keyed hash digests for shorter bigram combinations, such as ("pe","te"), will be much less common than keyed hash digests for longer combinations such as ("er","et","pe","te"). Similarly, bigram combinations which contain common bigrams, such as "et", will be correspondingly more common overall. In practice, most bigram combinations will be rare, and only relatively few will occur more than a few times in any set derived from real-life names. Alice can dramatically increase the difficulty of frequency analysis by including dummy records, with the same bigram digest, for those bigram digests which are relatively common. There is no need to include dummy records for the vast majority of bigram digests which occur only once or a few times. Of course, Alice needs to include an indicator that a record is a dummy in the information associated with the bigram digest. Carol is unable to determine the value of this indicator because the associated information is encrypted with David's public key. Thus, the information which Alice prepares in step 5 of Protocol 2 becomes (A.a_bigram_combination_digest, $\{(R_{rand}$, A.record_key_digest, dummy_record_flag, A.a_bigram_combination_length, A.a_length)$\}_{PublicKeyD}$). Bob mirrors these steps. In step 9, David simply discards any tuples in which the dummy_record_flag is set to true.

The other strategies discussed above for improving the security of Protocol 1, such as "hardening" Carol's and David's systems to make misuse of the information which they are sent much more difficult, can also be used with Protocol 2.

A proof-of-concept implementation of Protocol 2 is given in an Appendix 1 (see additional file 1) as a series of programs written in the free, open-source Python language [42]. The programs simulate the exchange of information between parties by simply writing files to shared directories in the computer's file system. Encryption of these message files has been omitted from this proof-of-concept implementation in order to facilitate examination of the information which is passed between parties. Instructions for obtaining Python and running the demonstration programs are included in the appendix.

Additional protection through last-minute election of third parties

A further means of reducing the risk of collusion between parties would be to have many Carols and Davids availa-

ble, all functionally equivalent, and for Alice and Bob to mutually agree on which instances of Carol and David to use for each data element (attribute) only at the very last moment. If the number of instances of Carol and David available on the network was considerably larger than the number of data elements in Alice's and Bob's data collections, an attacker would need to compromise the security of a large number of computers (or subvert their administrators) in order to have a reasonable chance of gaining access to some of the information provided by Alice and Bob to the chosen instances of Carol or David. The fully automated nature of the processing carried out by Carol and David lends itself to replication on multiple hosts on a network, with each host capable of acting in either rôle. Such dynamic allocation of tasks to substitutable computing resources is consistent with the increasingly popular "Grid" model of computing [43]. It may also be possible to use the model of public-resource distributed computing pioneered by the Search for Extraterrestrial Intelligence at Home (SETI@home) project [44].

Some modifications to Protocol 2 would be required if last minute election were to be used. An additional, initial step would be required in which Alice and Bob mutually agree on which instance of Carol and David will be responsible for processing each of the data elements which Alice's and Bob's data collections have in common. A trusted third party should be used to undertake this allocation – because Alice wants to be sure that Bob has no knowledge of the allocation in advance, and vice versa. Alternatively, Alice and Bob could use the honest coin flip or Diffie-Hellman protocols mentioned earlier to perform the allocation [35,37]. Additionally, because the information which Alice and Bob send to a particular instance of Carol contains data encrypted with the public key of a particular instance of David, Alice and Bob must indicate to Carol to which instance of David she should send her results.

Minimal-knowledge similarity using other string similarity functions

Although bigram similarity functions have been shown to be adequate comparators for record linkage purposes, several other string similarity comparators are known to perform better, in particular the Jaro comparator and the Winkler modification of it [31,32,45]. Is it possible to design minimal-knowledge versions of any of these comparison functions?

In fact, designing minimum-knowledge protocols to compute the Jaro or Winkler comparators is relatively straightforward, using chaffing and winnowing to hide the real data from Carol and David. Although such protocols are computationally feasible for very small data sets, in practice there is a combinatorial explosion which renders them impractical for almost all record linkage applications. One such protocol is given in Appendix 2 (see additional file 2) in order to stimulate thought.

In order to avoid combinatorial problems, "dynamic programming" techniques are often used to calculate similarity measures such as the Levenshtein edit distance when the original values are known – that is, in non-blindfolded settings. Related methods are used to calculate the Jaro and Winkler similarity metrics. Suzuki and Yokoo have recently demonstrated, in the context of secret combinatorial auctions, that secure dynamic programming is possible using properties of polynomial expressions [46]. Further work is needed to determine whether it is possible to devise a practical method for calculating the Jaro, Winkler and other similarity measures using these techniques.

Minimal-knowledge comparison of scalar values

So far, we have only considered the comparison of character strings. Equality comparisons of scalar values such as numbers or dates can also be performed using methods similar to those of Protocol 2 – the scalar value is treated as a sequence of bytes and is transformed by a keyed hash algorithm in the same way as a character string. Interval or "delta" comparisons of discrete scalars, such as integers, is also straightforward, using the following "bracketing" method.

An attribute of Alice's data collection contains integer values, perhaps ages in years, and she wishes to determine which values are within 1 year of values held by Bob. For each of her records, Alice generates keyed hash digests for all possible values within the target comparison range. For a value of 35, Alice sends Carol three tuples each containing a hash digest of the relevant record key plus hash digests for the numbers 34, 35 and 36. Bob sends Carol a single tuple for each value for the equivalent attribute in his data collection, containing the hash digest of the relevant record key plus the hash digest for the relevant number – say 36. Carol compares the hash digests in order to determine which of Alice's and Bob's values are within the agreed tolerance of each other.

Clearly this technique requires that continuous values are first converted to discrete values using binning or some other quantisation method negotiated between Alice and Bob beforehand. Date and time values can also be compared in a similar fashion by representing them as a count of time units since the start of an arbitrary epoch.

Comparisons of dates can also take into account common typographical or representational errors, such as transposition of day and month, by using a set of rules to generate likely permutations of a particular date or other value.

It should be noted that Carol does not need to be informed of the name or nature of the data collection attribute from which the hash digests which she processes are derived. Nevertheless, some attributes such as age have rather characteristic frequency distributions, and it is therefore important that the "winnowing and chaffing" described above is used to prevent such frequency information from leaking to Carol.

A protocol for blindfolded record linkage

The essence of modern record linkage techniques is the comparison of a number of partial identifiers (data elements) between pairs of records from two data collections, A and B. Each pair of records forms an agreement pattern (or vector), γ in a comparison space Γ, which is equivalent to all possible agreement patterns of the partially-identifying attributes a, b,..*i* which are common to both A and B. The classical model of probabilistic record linkage, as formalised in 1969 by Fellegi and Sunter [47], assumes statistical independence of the comparisons, although more recent methods may model some dependencies between the data elements.

Thus, the first task is to compare each of the partially-identifying data elements and return a similarity score (bigram_score) for each pair. We have demonstrated how this can be done without revealing the actual values involved, albeit at the cost of significantly increased communication overhead. In the following discussion, we will assume the use of Protocol 2 to undertake the minimal-knowledge similarity comparisons of individual attributes, and thus refer to Carol and David as the third and fourth parties. We will refer to the following record linkage protocol as Protocol 3.

We introduce two new parties: Edith, whose role is to combine the results of minimal-knowledge comparisons for individual data items, and Freddy, a putative researcher who is to receive linked but de-identified records from Alice's and Bob's data collections.

Protocol 3 consists of the following steps:

1. For each of the partially-identifying data elements, a, b...*i*, in their data collections A and B, Alice and Bob dispatch the information needed for the similarity comparison task defined in Protocol 2 to different instances of Carol, which we will denote as $Carol_a$, $Carol_b$ and so on. A different shared secret hashing key, K_{AB}, should be used for the information sent to each instance of Carol, because each of these comparison tasks is independent. As prescribed in Protocol 2, each instance of Carol forwards the necessary information in encrypted form to the designated instance of David, who computes the similarity score (bigram_score) for each pair of Alice's and Bob's records.

2. Each instance of David then sends the results of his computations to Edith as a set of tuples of the form (A.record_key_digest, B.record_key_digest, data_item_identifier, similarity_score), where data_item_identifier is the name or some other identifier of the attributes a,b,...*i* present in both data collections, A and B. It may be desirable for each instance of David to slightly perturb and/or quantise the similarity_score values before sending them to Edith, in order to reduce the frequency information present in these scores – information which could assist Edith in a cryptanalytic attack.

Note that not every possible combination of A.record_key_digest and B.record_key_digest will be present in the data sent to Edith by each instance of David -only those record pairs for which the similarity score was greater than some threshold score. It is a common practice in many record linkage systems to disregard similarity scores below a threshold, such as 0.8 [31]. Reduction of the record comparison space by this means has been found to have good coverage properties compared to traditional record linkage "blocking" using compound indexes [48].

Edith also receives record pairs for comparisons in which one or both values were missing.

3. Edith then performs an outer join of these data using the tuple (A.record_key_digest, B.record_key_digest) as the join key, to form tuples of the form:

(A.record_key_digest, B.record_key_digest, a.similarity_score,

b.similarity_score, ..., *i*.similarity_score)

If Edith receives no similarity score for a particular data item in a particular row in this matrix, the similarity score is taken to be zero, indicating disagreement for that data item. The number of rows in the matrix assembled by Edith will be considerably less than the product of the number of rows in A and B, for the reason discussed above.

4. Edith now multiplies this matrix by vectors of individual comparison weights – one each for agreement, disagreement and missing values. Comparison weights for various degrees of similarity, derived from the similarity score, can also be used. The resulting values are summed row-wise and the ratio of the summed agreement weights to the summed disagreement weights is calculated – this value is known as the "matching weight". The matching

weights are compared to a pair of criteria, known as the matching parameters, in order to classify records as matches, possible matches and non-matches. A more complete exposition of the Fellegi-Sunter model can be found in [49]. In many circumstances adequate results (in terms of acceptable numbers of false matches and missed matches) can be obtained using individual comparison weights and matching parameters which are determined heuristically by inspection of candidate pairs of records. Such an approach is unacceptable in the context of blindfolded record linkage. It may be possible to use weights and parameters derived from other matching exercises. However, as Winkler has pointed out, comparison weights for individual data items are strongly influenced by the error rates for those data items in each data collection, or in particular subsets of data collections [50]. Thus the re-use of weights and parameters which have been found to work well in one setting may be quite suboptimal in other settings.

Fortunately, it is possible to estimate individual comparison weights and the matching parameters directly from the data. Fellegi and Sunter described a method for doing this in certain constrained circumstances when conditional independence of each data item was assumed. Winkler demonstrated that the EM (expectation maximisation) algorithm could be used to calculate match weights and parameters with fewer assumptions, including relaxation of the conditional independence assumption [51]. "Scoring" and Markov chain Monte Carlo methods can also be used [52,53]. An unresolved problem is that although these techniques produce an optimal partitioning of pairs of records based on the match weights, the resulting partitions may not correspond exactly to matches and non-matches. Nevertheless, Quantin *et al.* reported that in a trial of automated record linkage using weights determined by EM, a sensitivity of 0.97 and specificity of 0.93 was achieved compared to traditional, manual linkage of the same data [12].

We will assume that Edith uses one of these automatic methods to estimate matching weights and parameters, and thus arrives at a set of linked records in A and B – or more precisely, pairs of (A.record_key_digest, B.record_key_digest). Researcher Freddy now needs to be supplied with anonymous (or at worst de-identified) records containing the required data from Alice and Bob's data collections – but only for these linked records. In doing so, Alice must not learn which of her records match Bob's, and vice versa. There is a straightforward solution to this problem using public key encryption.

5. Alice assembles the data required by Freddy for every record in her database, and encrypts each record with Freddy's public key, and associates the resulting cipher text with the corresponding record_key_digest. Alice sends these data to Edith. Bob does the same. Edith now performs an inner join of these data with her list of matching records, using the record_key_digest values as the join key. Edith forwards the encrypted data records from this inner join to Freddy, who decrypts the data for each record using his private key.

Thus, Freddy obtains the data items he needs, but only for the linked records. Edith learns the number of linked records but is unable to inspect the actual data (because it is encrypted with Freddy's key), and Alice and Bob learn nothing.

It must be recognised that although the linked information which Freddy receives lacks any directly identifying data items such as names or dates of birth, the conjunction of other less specific attributes from Alice's and Bob's data collection may render the data readily re-identifiable in the presence of external information. Freddy therefore needs to ensure that the data he receives is stored securely and used ethically – in other words, that no attempt is made to re-identify the data unless the study protocol specifically permits this. Alice and Bob may, quite reasonably, require that Freddy only uses the data he has received in a special-purpose secure analysis facility, which has tight controls on the information which may enter and leave the facility – analogous to the special facilities in which microbiologists and virologists are required to work when handling hazardous organisms. Indeed, Edith could be instructed to forward the linked data file, encrypted with Freddy's public key, to a suitable secure analysis facility. Freddy would then need to bring his private key to that facility in order to decrypt and analyse the data.

Extension to frequency-based matching

Protocol 3 does not provide for the use of information about the relative frequency of particular names or other data items as part of the linkage process. Newcombe *et al.* introduced the idea of using such information [54], based on the intuition that two records with the surname "Stockhausen" are more likely to represent the same person than two records with the surname "Smith". Fellegi and Sunter gave two methods for estimating frequency-based weights [47]. Winkler subsequently extended this work to use the EM algorithm for frequency-based parameter estimation [55].

In the context of blindfolded record linkage, it is important that Alice or Bob do not provide precise information about the relative frequencies of particular values in their data collections to any other party, because such information provides the basis for a frequency-based cryptanalytic attack against the values. However, if Alice and Bob trust

Edith sufficiently, it may be acceptable for them to provide her with approximate relative frequency information for each of the attributes in the comparison vector to Edith. Such a step needs to be taken advisedly, because it provides Edith with quite a lot of information which she can use in a cryptanalytic attack against Alice's and Bob's data. In addition, Winkler has noted that frequency-based linkage does not always improve matching completeness and accuracy, and that it may not be needed if there is sufficient redundancy in the data items available for matching [56]. More work is required to quantitatively assess the marginal gain afforded by frequency information in situations in which blindfolded record linkage might be used.

Discussion

Other uses for blindfolded record linkage protocols

So far we have concentrated on the use of blindfolded record linkage protocols for epidemiological studies. However, there are a number of other potential uses.

One example is the provision of privacy-preserving geocoding services. "Geocoding" is the process of converting an address or place name into a pair of geographical co-ordinates, such as latitude and longitude. This is done by matching a target address or place name against a comprehensive database of geo-referenced addresses and place names, and returning the co-ordinates associated with the matching address (or addresses if the match is not singular). Because typographical errors in addresses and place names are common, it is important that the matching algorithm employed by the geocoding engine uses some form of similarity comparator in order to achieve good match rates.

The geo-referenced database can be distributed *in toto* to parties wishing to geocode their data, but this may pose problems due to the size, volatility or proprietary nature of the database. Alternatively, parties can transmit addresses or place names to a centralised geocoding service, thus avoiding the need to distribute the reference database and associated geocoding engine [57]. A problem with the latter model is the unavoidable loss of privacy associated with the need to reveal addresses to the geocoding service.

However, it is easy to design a privacy-preserving geocoding service using a minor variation of the blindfolded record linkage protocol described above.

Alice is a client of the service. She has one or more addresses which she wishes to geocode without revealing what those addresses actually are. Bob has a geo-referenced database of addresses which he does not wish to distribute to other parties for business or logistical reasons. Using standardised parsing methods, such as those described in [58], Bob preprocesses his entire database, separating each address into attributes, such as street number, street name, street type, suburb and so on (Bob may already have his addresses stored like this). Alice preprocesses her address(es) in the same manner. Alice indicates to Bob that she wishes to make use of the geocoding service, and with the assistance of multiple instances of Carol and David, the pair carry out the steps described in Protocol 2. One instance of Carol and David is used for each address attribute.

The similarity information for each address attribute is then assembled and processed by Edith as described in Protocol 3, perhaps with some minor differences. In step 4, simpler, deterministic linkage methods using predefined match weights and parameters might be employed, rather than probabilistic techniques. In step 5, there is no need for Alice to provide encrypted data to Edith. Rather, Bob would provide Edith with a set of tuples comprising the encrypted record key for each address record in his reference database, together with the co-ordinates for that address, encrypted using Alice's public key. Based on the results of the matching process, Edith simply forwards the relevant record(s) from this set of tuples to Alice, who then decrypts the co-ordinates for each of her original addresses. No-one other than Alice knows what these addresses or their co-ordinates are, and Bob only needs to reveal an encrypted (via keyed hashes) version of his geo-referenced address database to external parties. This scheme provides good protection against attacks by any one of the parties involved but, as discussed above, is vulnerable to collusion between either Alice or Bob and the various instances of Carol, David and Edith. Last-minute election of substitutable services can be used to mitigate this risk.

Berman [59] has described a "threshold" method for annotating confidential medical records. Although the method dissociates information from particular patient records, it does not guarantee that confidential or identifying information does not leak to the annotating entity. Protocol 2 could be used to secure the annotation. It could also be used by pharmaceutical companies who wish to determine whether candidate names for new drugs which they have under development are similar to new drug names being considered by their competitors, without having to reveal the actual names. *n*-grams can also be formed from words (or other tokens), rather than individual letters. Thus, minimum-knowledge *n*-gram similarity comparisons can be used to compare record-or line-oriented text or data, such as poetry or computer program code, or any other information comprising sequences of tokens. In the case of computer program code, the output of the tokeniser built into the language compiler or interpreter could be used as input to detect

potentially plagiarised program code, without either party revealing their intellectual property. The only proviso is that the average number of words or tokens in each line must be reasonably small. For this reason the technique is unlikely to be suitable for the comparison of genomic or proteomic sequences.

Conclusion

In this paper we have demonstrated that it is possible to perform minimal-knowledge similarity comparisons between strings, using keyed hash combinations of *n*-grams, albeit at the cost of considerable computing and communications overhead. Minimal-knowledge comparison of scalar quantities is also described, using a related "bracketing" technique. We have outlined how these techniques can be combined with public key cryptography and current probabilistic record linkage techniques to create a generalised framework for blindfolded record linkage.

Traditional methods of record linkage are vulnerable to misdeeds by or compromise of single parties, particular the party undertaking the linkage. Although the protocols described in this paper are not unconditionally secure, they do make it much harder for any one party involved in a linkage operation to determine any useful information about the data being linked. The protocols remain vulnerable to collusion between, or simultaneous compromise of, two or more parties. However, because most of the computation involved can be fully automated, it is possible to use last-minute allocation of tasks to substitutable service providers to minimise the likelihood of such collusion or compromise.

An immediate task is the creation of a generic blindfolded record linkage application framework, in which different minimal-knowledge comparison protocols can be trialled. This work is under way, as part of the freely-available, open-source *Febrl* research project [60,61], and some results obtained with real-life data sets will be reported subsequently. Further exploration of the utility of variations of *n*-gram comparators would also be valuable, as well as evaluation of the impact of limiting the number of *n*-gram combinations used in Protocol 2.

The potential use of blindfolded record linkage to preserve privacy also poses some interesting ethical and philosophical questions. Most record linkage studies require either individual consent by the subjects whose records are to be linked, or more usually, a determination by an ethics committee or institutional review board that the public good which is likely to accrue from the study outweighs the invasion of personal privacy posed by the study. However, it could be argued from the perspective of Berkeley's idealism that if record linkage can be undertaken without revealing any identifying information to anyone who does not already have access to that information, then there is no invasion of privacy [62]. Despite the imperfections in the protocols presented here, we hope that this paper stimulates further work on blindfolded record linkage.

Competing Interests

None declared.

Authors' contributions

The protocols described in this paper were originally developed by TC. PC undertook the efficiency analysis and suggested efficiency improvements. TC drafted the original manuscript to which PC made edits and improvements. TC wrote the original Python proof-of-concept programs, which PC improved substantially. Both authors read and approved the final manuscript.

Additional material

Additional File 1

Annotated copy of Python programs which illustrate Protocol 2, and their output.
Click here for file
[http://www.biomedcentral.com/content/supplementary/1472-6947-4-9-S1.pdf]

Additional File 2

An impractical protocol for minimum-knowledge calculation of the Jaro and Winkler metrics.
Click here for file
[http://www.biomedcentral.com/content/supplementary/1472-6947-4-9-S2.pdf]

Additional File 3

Installable copy of Python prgrammes and associated data files for Posix (Unix, Linux and Apple Mac OS X) computers. Instructions for the installation and use of these programmes appear in Appendix 1 (additional file 1).
Click here for file
[http://www.biomedcentral.com/content/supplementary/1472-6947-4-9-S3.gz]

Additional File 4

Installable copy of Python prgrammes and associated data files for Microsoft Windows computers. Instructions for the installation and use of these programmes appear in Appendix 1 (additional file 1).
Click here for file
[http://www.biomedcentral.com/content/supplementary/1472-6947-4-9-S4.zip]

Acknowledgements

None.

References

1. Armstrong BK, Kricker A: **Record linkage – a vision renewed.** *Aust N Z J Public Health* 1999, **23:**451-452.
2. Boruch R, Cecil J: **Assuring the Confidentiality of Social Research Data.** Philadelphia, University of Philadelphia Press; 1979.
3. Pommerening K, Miller M, Schidtmann I, Michaelis J: **Pseudonyms for cancer registries.** *Methods Inf Med* 1996, **35:**112-121.
4. Kelman CW, Bass AJ, Holman CD: **Research use of linked health data-a best practice protocol.** *Aust N Z J Public Health* 2002, **26:**251-255.
5. Churches T: **A proposed architecture and method of operation for improving the protection of privacy and confidentiality in disease registers.** *BMC Med Res Methodol* 2003, **3:**1 [http://www.biomedcentral.com/1471-2288/3/1].
6. Blakely T, Woodward A, Salmond C: **Anonymous linkage of New Zealand mortality and Census data.** *Aust N Z J Public Health* 2000, **24:**92-5.
7. Dusserre L, Quantin C, Bouzelat H: **A one way public key cryptosystem for the linkage of nominal files in epidemiological studies.** *Medinfo* 1995, **8:**644-7.
8. Bouzelat H, Quantin C, Dusserre L: **Extraction and anonymity protocol of medical file.** *Proc AMIA Annu Fall Symp* 1996:323-27.
9. Quantin C, Bouzelat H, Dusserre L: **A computerized record hash coding and linkage procedure to warrant epidemiological follow-up data security.** *Stud Health Technol Inform* 1997, **43:**339-42.
10. Quantin C, Kerkri E, Allaert FA, Bouzelat H, Dusserre L: **Security aspects of medical file regrouping for the epidemiological follow-up.** *Medinfo* 1998, **9:**1135-7.
11. Quantin C, Bouzelat H, Allaert FA, Benhamiche AM, Faivre J, Dusserre L: **How to ensure data security of an epidemiological follow-up: quality assessment of an anonymous record linkage procedure.** *Int J Med Inf* 1998, **49:**117-22.
12. Quantin C, Bouzelat H, Allaert FA, Benhamiche AM, Faivre J, Dusserre L: **Automatic record hash coding and linkage for epidemiological follow-up data confidentiality.** *Methods Inf Med* 1998, **37:**271-7.
13. Bouzelat H: **Anonymat et chaînage de fichiers médicaux en vue d'études épidémiologiques.** *Thèse de Docteur d'Université specialiste en Informatique Médicale. Université de Bourgogne* 1998.
14. Schadow G, Grannis SJ, McDonald CJ: **Privacy-preserving distributed queries for a clinical case research network.** In *Proc IEEE International Conference on Data Mining, Workshop on Privacy, Security and Data Mining, Maebashi City, Japan. In Confernces in Research and Practice in Information Technology* 2002, **14:**43-54 [http://crpit.com/Vol14.html]. Australian Computer Society
15. Yao AC: **Protocols for secure communication.** In *Proc of 23rd IEEE Symposium on the Foundations of Computer Science (FOCS).IEEE* 1982:160-6.
16. Hirt M, Maurer U, Przydatek B: **Efficient Secure Multi-Party Computation.** In *Proc Asiacrypt Lecture Notes in Computer Science Volume 1976*. Springer-Verlag; 2000:143-161.
17. Anderson RJ: *Security Engineering: A Guide to Building Dependable Distributed Systems* New York: Wiley Computer Publishing; 2001:3.
18. Anderson RJ, Needham R: **Programming Satan's computer.** In *Computer Science Today – Recent Trends and developments, of Lecture Notes In Computer Science Volume 1000*. Edited by: van Leeuwen J. Springer-Verlag; 1995:426-441.
19. Schneier B: *Applied Cryptography* 2nd edition. New York: John Wiley and Sons; 1996:31-32.
20. Etheridge Y: **PKI (public key infrastructure) – how and why it works.** *Health Manag Technol* 2001, **22:**20-21.
21. Levenshtein VI: **Binary codes capable of correcting deletions, insertions and reversals.** *Sov Phys Dokl* 1966.
22. National Institute of Standards and Technology: *Digital signature standard. NIST FIPS PUB 186, US Department of Commerce* 1994.
23. Schneier B: *Applied Cryptography* 2nd edition. New York: John Wiley and Sons; 1996:30-31.
24. **ibid. 52.** .
25. Bellare M, Canetti R, Krawczyk H: **Message authentication using hash functions – the HMAC construction.** *RSA Laboratories' CryptoBytes* 1996, **2:**1-5 [http://www.cs.ucsd.edu/users/mihir/papers/hmac.html].
26. Rivest RL: **Chaffing and Winnowing: Confidentiality without Encryption.** *MIT Lab for Computer Science* 1998 [http://theory.lcs.mit.edu/~rivest/chaffing.txt].
27. Hall PAV, Dowling GR: **Approximate string comparison.** *Computing Surveys* 1980, **12:**381-42.
28. Philips L: **Hanging on the Metaphone.** *Computer Language* 1990, **7:**39-43.
29. Adamson GW, Boreham J: **The use of an association measure based on character structure to identify semantically related pairs of words and document titles.** *Information and Storage Retrieval* 1974, **10:**253-60.
30. van Rijsbergen RCJ: *Information retrieval* Second edition. London: Butterworths; 1981.
31. Porter EH, Winkler WE: **Approximate String Comparison and its Effect on an Advanced Record Linkage System.** In *Record Linkage Techniques – 1997: Proceedings of an International Workshop and Exposition, March 1997, Arlington VA* 1997 [http://www.census.gov/srd/papers/pdf/rr97-2.pdf]. Washington DC: Federal Committee on Statistical Methodology, Office of Management and Budget
32. Jaro MA: **Advances in Record-Linkage Methodology as Applied to Matching the 1985 Ce nsus of Tampa, Florida.** *Journal of the American Statistical Association* 1989, **89:**414-420.
33. Brew C, McKelvie D: **Word-pair extraction for lexicography.** In *Proceedings of the Conference on New Methods in Natural Language Processing (NeMLaP-2), Bilkent University, Ankara, Turkey, September 1996* 1996:45-55 [http://citeseer.nj.nec.com/brew96wordpair.html].
34. Howe D: *Free On-Line Dictionary of Computing* 1994 [http://foldoc.doc.ic.ac.uk]. London: Department of Computing, Imperial College
35. Blum M: **Coin flipping by telephone.** In *Advances in Cryptology: A Report on CRYPTO 81, IEEE Workshop on Communications Security, Santa Barbara, August 24–26, 1981. U.C. Santa Barbara, Dept. of Elec. and Computer Eng., ECE Report No 82-04* 1982:11-15 [http://www-2.cs.cmu.edu/~mblum/research/pdf/coin/].
36. Kilian J: *Uses of Randomness in Algorithms and Protocols* Cambridge, Massachusetts: MIT Press; 1990.
37. Diffie W, Hellman ME: **New directions in cryptography.** *IEEE Trans. Inform. Theory IT-22* 1976, **6:**644-654.
38. Rivest RL: **Chaffing and Winnowing: Confidentiality without Encryption.** *MIT Lab for Computer Science* 1998 [http://theory.lcs.mit.edu/~rivest/chaffing.txt].
39. Singh S: *The Code Book: The Secret History of Codes and Codebreaking* London: Fourth Estate; 1999:17.
40. National Security Agency: *Security Enhanced Linux* [http://www.nsa.gov/selinux/].
41. Rivest RL: **Chaffing and Winnowing: Confidentiality without Encryption.** *MIT Lab for Computer Science* 1998 [http://theory.lcs.mit.edu/~rivest/chaffing.txt].
42. Python Software Foundation: *Python (computer programming language)* [http://www.python.org].
43. Foster I, Kesselman C: **Computational grids.** In *The Grid: Blueprint for a New Computing Infrastructure* Edited by: Foster I, Kesselman C. San Francisco: Morgan Kaufmann; 1986:15-52.
44. Anderson DP, Cobb J, Korpela E, Lebofsky M, Werthimer D: **SETI@home: An Experiment in Public-Resource Computing.** *Communications of the Association for Computing Machinery* 2002, **45:**56-61 [http://setiathome.ssl.berkeley.edu/cacm/cacm.html].
45. Budzinsky CD: *Automated spelling correction* Ottawa: Statistics Canada; 1991.
46. Suzuki K, Yokoo M: **Secure combinatorial auctions by dynamic programming with polynomial secret sharing.** In *Proceedings of Sixth International Financial Cryptographic Conference (FC-02), Bermuda, March 2002. Lecture Notes in Computer Science* 2002, **2357:** [http://www.kecl.ntt.co.jp/csl/ccrg/members/yokoo/PDF/fc02.pdf]. Springer-Verlag
47. Fellegi IP, Sunter AB: **A Theory for Record Linkage.** *Journal of the American Statistical Association* 1969, **64:**1183-1210.
48. Baxter R, Christen P, Churches T: **A comparison of fast blocking methods for record linkage.** *Proc ACM Workshop on Data Cleaning, Record Linkage and Object Identification* 2003 [http://www.act.cmis.csiro.au/rohanb/PAPERS/kdd03clean.pdf]. Washington DC
49. Winkler WE: **Advanced Methods for Record Linkage.** *Technical report* 1994 [http://www.census.gov/srd/papers/pdf/rr94-5.pdf]. Washington, DC: Statistical Research Division, U.S. Bureau of the Census
50. Winkler WE: **Methods for Record Linkage and Bayesian Networks.** *Research Report Series Statistics #2002-05* 2002 [http://

www.census.gov/srd/papers/pdf/rrs2002-05.pdf]. Washington, DC: Statistical Research Division, U.S. Bureau of the Census
51. Winkler WE: **Using the EM algorithm for weight computation in the Fellegi-Sunter model of record linkage.** *Proceedings of the Section on Survey Research Methods, American Statistical Association* 1988:667-71 [http://www.census.gov/srd/papers/pdf/rr2000-05.pdf].
52. Thibaudeau Y: **The discrimination power of dependency structures in record linkage.** *Survey Methodology* 1993, **19:**31-38.
53. Larsen MD, Rubin DB: **Iterative automated record linkage using mixture models.** *Journal of American Statistical Association* 2001, **79:**32-41.
54. Newcombe HB, Kennedy JM, Axford SJ, James AP: **Automatic linkage of vital records.** *Science* 1959, **130:**954-959.
55. Winkler WE: **Frequency-based matching in the Fellegi-Sunter model of record linkage.** *Proceedings of the Section on Survey Research Methods, American Statistical Association* 1989:778-783 [http://www.census.gov/srd/papers/pdf/rr2000-06.pdf].
56. Winkler WE: **The state of record linkage and current research problems.** *Statistical Society of Canada, Proceedings of the Survey Methods Section* 1999:73-80 [http://www.census.gov/srd/papers/pdf/rr99-04.pdf].
57. OpenGIS Consortium Inc: *OpenGIS Location Services (OpenLS) Core Services. OpenGIS Project Document OGC 03-006r3. OpenGIS Consortium* 2003.
58. Churches T, Christen P, Lim K, Zhu JX: **Preparation of name and address data for record linkage using hidden Markov models.** *BMC Medical Informatics and Decision Making* 2002, **2:**9 [http://www.biomedcentral.com/1472-6947/2/9].
59. Berman JJ: **Threshold protocol for the exchange of confidential medical data.** *BMC Medical Research Methodology* 2002, **2:**12 [http://www.biomedcentral.com/1471-2288/2/12].
60. Christen P, Churches T: *Febrl – Freely extensible biomedical record linkage. ANU Computer Science Technical Reports TR-CS-02-05* 2002 [http://datamining.anu.edu.au/linkage.html]. Canberra: Australian National University
61. Christen P, Churches T, Hegland M: *A Parallel Open Source Data Linkage System. Proceedings of the 8th PAKDD'04 (Pacific-Asia Conference on Knowledge Discovery and Data Mining), Sydney, Springer Lecture Notes in Artificial Intelligence (3056)* 2004.
62. Berkeley G: **A treatise concerning the principles of human knowledge.** :1710 [http://www.gutenberg.net/etext03/prhkn10.txt].

Pre-publication history

The pre-publication history for this paper can be accessed here:

http://www.biomedcentral.com/1472-6947/4/9/prepub