

# Anonymity for Continuous Data Publishing\*

Benjamin C. M. Fung\*

Ke Wang†

Ada Wai-Chee Fu§

Jian Pei†

\* Concordia University, Montreal, QC, Canada, fung@ciise.concordia.ca

† Simon Fraser University, Burnaby, BC, Canada, {wangk,jpei}@cs.sfu.ca

§ The Chinese University of Hong Kong, adafu@cse.cuhk.edu.hk

## ABSTRACT

$k$ -anonymization is an important privacy protection mechanism in data publishing. While there has been a great deal of work in recent years, almost all considered a single static release. Such mechanisms only protect the data up to the first release or first recipient. In practical applications, data is published continuously as new data arrive; the same data may be anonymized differently for a different purpose or a different recipient. In such scenarios, even when all releases are properly  $k$ -anonymized, the anonymity of an individual may be unintentionally compromised if recipient cross-examines all the releases received or colludes with other recipients. Preventing such attacks, called *correspondence attacks*, faces major challenges. In this paper, we systematically characterize the correspondence attacks and propose an efficient anonymization algorithm to thwart the attacks in the model of continuous data publishing.

## 1. INTRODUCTION

$k$ -anonymization [11] is a promising approach to data publishing while protecting the identity of individuals. The data holder has a table of the form [1]

$D(\text{Explicit\_identifier}, \text{Quasi\_identifier}, \text{Sensitive\_attribute})$ .

Explicit\\_identifier consists of identifying information (such as SSN and names). Quasi\\_identifier (such as date of birth, gender, and zip code) does not reveal identity, but can be used to link to a person or an explicit identity in some external sources [11]. Sensitive\\_attribute consists of other person-specific information (such as medical and DNA entries). Instead of publishing the original table  $D$ , the data holder publishes an anonymized release

$R(QID, \text{Sensitive\_attribute})$ ,

\*The work was supported by research grants from the Natural Sciences and Engineering Research Council of Canada and ENCS faculty start-up funds from Concordia University.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

EDBT'08, March 25–30, 2008, Nantes, France.

Copyright 2008 ACM 978-1-59593-926-5/08/0003 ...\$5.00.

where  $QID$  is a  $k$ -anonymized version of Quasi\\_identifier [11]: each record belongs to an equivalence class of size at least  $k$  and all records in an equivalence class are made indistinguishable with respect to  $QID$  by hiding some details. In a  $k$ -anonymized release, if an individual is linked to a record through  $QID$ , it is also linked to at least  $k - 1$  other records. Sensitive\\_attribute is not modified because the data usefulness depends on the exact information of this attribute.

$k$ -anonymization has been primarily studied for a single static release of data [3, 4, 6, 9, 11, 13]. In practice, however, new data arrives continuously and up-to-date data has to be published to researchers in a timely manner. One approach is to anonymize and publish new records *separately* each time they arrive. This naive approach suffers from severe data distortion because small increments are anonymized independently. Moreover, it is difficult to analyze a collection of independently anonymized data sets. For example, if the country name (e.g., Canada) is used in the release for first month records and if the city name (e.g., Toronto) is used in the release for second month records, counting the total number of persons born in Toronto is not possible. Another approach is to enforce the later release to be no more specialized than the previous releases [13]. The major drawback is that each subsequent release gets increasingly distorted, even if more new data are available.

### 1.1 A Practical Publishing Model

Suppose that the data holder previously collected a set of records  $D_1$  timestamped  $T_1$ , and published a  $k$ -anonymized version of  $D_1$ , denoted release  $R_1$ . Then the data holder collects a new set of records  $D_2$  timestamped  $T_2$  and wants to publish a  $k$ -anonymized version of all records collected so far,  $D_1 \cup D_2$ , denoted release  $R_2$ . Note,  $D_i$  contains the “events” that happened at time  $T_i$ . An event, once occurred, becomes part of the history, therefore, cannot be deleted. This *publishing scenario* is different from *update scenario* in standard data management where deletion of records can occur.  $R_i$  simply publishes the “history”, i.e., the events that happened up to time  $T_i$ . A real-life example can be found in California where the hospitals are required to submit specific demographic data of all discharged patients every six months.<sup>1</sup> The above publishing model directly serves the following scenarios.

**Continuous data publishing.** Publishing the release  $R_2$  for  $D_1 \cup D_2$  would permit an analysis on the data over the combined time period of  $T_1$  and  $T_2$ . It also takes the advantage of data abundance over a longer period of time

<sup>1</sup><http://www.oshpd.ca.gov/HQAD/PatientLevel/>

to reduce data distortion required by anonymization.

**Multi-purpose publishing.** With  $D_2$  being empty,  $R_1$  and  $R_2$  can be two releases of  $D_1$  anonymized differently to serve different information needs, such as correlation analysis vs clustering analysis, or different recipients, such as a medical research team vs a health insurance company, who may collude together by sharing their received data.

We first describe the publishing model with two releases and then show the extension beyond two releases in Section 7. Following the convention of  $k$ -anonymity [13], we assume that each individual has at most one record in  $D_1 \cup D_2$ . This assumption holds in many real-life databases. For example, in a normalized customer data table, each customer has only one profile. In the case that an individual has a record in both  $D_1$  and  $D_2$ , there will be two duplicates in  $D_1 \cup D_2$  and one of them can be removed in a preprocessing. As in [17], we assume that the records of an individual remain the same in  $D_1$  and  $D_2$ .

## 1.2 Correspondence Attacks

We show, by an example, that the traditional  $k$ -anonymization is insufficient for preventing correspondence attacks.

**EXAMPLE 1.1.** Consider Tables 1-2, where  $QID$  is [Birthplace, Job] and the sensitive attribute is Disease. The data holder (e.g., a hospital) published the 5-anonymized  $R_1$  for 5 records  $a_1$ - $a_5$  collected in the previous month (i.e., timestamp  $T_1$ ). The anonymization was done by generalizing UK and France into Europe; the original values in the brackets are not released. In the current month (i.e., timestamp  $T_2$ ), the data holder collects 5 new records (i.e.,  $b_6$ - $b_{10}$ ) and publishes the 5-anonymized  $R_2$  for all 10 records collected so far. Records are shuffled to prevent mapping between  $R_1$  and  $R_2$  by their order. The recipients know that every record in  $R_1$  has a “corresponding record” in  $R_2$  (that originates from the same record in  $D_1$ ). One recipient, the *attacker*, tries to identify his neighbor Alice’s record from  $R_1$  or  $R_2$ , knowing that Alice was admitted to the hospital, as well as, Alice’s  $QID$  and timestamp. Consider the following scenarios.

**Scenario I.** Alice has  $QID$ =[France, Lawyer] and timestamp  $T_1$ . The attacker seeks to identify Alice’s record in  $R_1$ . Examining  $R_1$  alone, Alice’s  $QID$  matches all 5 records in  $R_1$ . However, examining  $R_1$  and  $R_2$  together, the attacker learns that the records  $a_1, a_2, a_3$  cannot all originate from Alice’s  $QID$ ; otherwise  $R_2$  would have contained at least three records of [France, Professional, Flu] since every record in  $R_1$  has a corresponding record in  $R_2$ . Consequently, the attacker excludes one of  $a_1, a_2, a_3$  as possibility; the choice among  $a_1, a_2, a_3$  does not matter as they are identical.

**Scenario II.** Alice has  $QID$ =[France, Lawyer] and timestamp  $T_1$ . Knowing that  $R_2$  contains all records at  $T_1$  and  $T_2$ , the attacker seeks to identify Alice’s record in  $R_2$ . The attacker infers that, among the matching records  $b_4$ - $b_8$  in  $R_2$ , at least one of  $b_4, b_5, b_6$  must have timestamp  $T_2$ ; otherwise  $b_4, b_5, b_6$  would have timestamp  $T_1$ , in which case there would have been at least three (corresponding) records of the form [Europe, Lawyer, HIV] in  $R_1$ . In this case, the attacker excludes at least one of  $b_4, b_5, b_6$  since Alice has timestamp  $T_1$ .

**Scenario III.** Alice has  $QID$ =[UK, Lawyer] and timestamp  $T_2$  and the attacker seeks to identify Alice’s record in  $R_2$ . The attacker infers that, among the matching records  $b_1, b_2, b_3, b_9, b_{10}$  in  $R_2$ , at least one of  $b_1, b_2, b_3$  must have timestamp  $T_1$ ; otherwise one of  $a_1, a_2, a_3$  would have no corresponding record in  $R_2$ . In this case, at least one of  $b_1, b_2, b_3$

**Table 1: 5-anonymized  $R_1$**

RID	Birthplace	Job	Disease
$(a_1)$	Europe (UK)	Lawyer	Flu
$(a_2)$	Europe (UK)	Lawyer	Flu
$(a_3)$	Europe (UK)	Lawyer	Flu
$(a_4)$	Europe (France)	Lawyer	HIV
$(a_5)$	Europe (France)	Lawyer	HIV

**Table 2: 5-anonymized  $R_2$**

RID	Birthplace	Job	Disease
$(b_1)$	UK	Professional (Lawyer)	Flu
$(b_2)$	UK	Professional (Lawyer)	Flu
$(b_3)$	UK	Professional (Lawyer)	Flu
$(b_4)$	France	Professional (Lawyer)	HIV
$(b_5)$	France	Professional (Lawyer)	HIV
$(b_6)$	France	Professional (Lawyer)	HIV
$(b_7)$	France	Professional (Doctor)	Flu
$(b_8)$	France	Professional (Doctor)	Flu
$(b_9)$	UK	Professional (Doctor)	HIV
$(b_{10})$	UK	Professional (Lawyer)	HIV

is excluded since Alice has timestamp  $T_2$ .

In each scenario, at least one matching record is excluded, so the 5-anonymity of Alice is compromised. ■

All these attacks “crack” some matching records in  $R_1$  or  $R_2$  by inferring that they either do not originate from Alice’s  $QID$  or do not have Alice’s timestamp. Such cracked records are not related to Alice, thus, excluding them allows the attacker to focus on a smaller set of candidates. Since cracked records are identified by cross-examining  $R_1$  and  $R_2$  and by exploiting the knowledge that every record in  $R_1$  has a “corresponding record” in  $R_2$ , such attacks are called *correspondence attacks*.

Having access to only  $R_1$  and  $R_2$ , not  $D_1$  and  $D_2$ , cracking a record is not straightforward. For example, to crack a record in  $R_1$  for Alice having  $QID$ =[France, Lawyer], the attacker must show that the *original* birthplace in the record is not France, whereas the published Europe may or may not originate from France. Similarly, it is not straightforward to infer the timestamp of a record in  $R_2$ . For example, any three of  $b_1, b_2, b_3, b_7, b_8$  can be the corresponding records of  $a_1, a_2, a_3$ , so none of them must have timestamp  $T_1$ . In fact, observing only the published records, there are many possible assignments of corresponding records between  $R_1$  and  $R_2$ . For example, one assignment is

$$(a_1, b_1), (a_2, b_2), (a_3, b_3), (a_4, b_4), (a_5, b_5),$$

where the original record represented by  $(a_1, b_1)$  is [UK, Lawyer, Flu]. Another assignment is

$$(a_1, b_7), (a_2, b_2), (a_3, b_8), (a_4, b_6), (a_5, b_9).$$

In this assignment, the original record represented by  $(a_1, b_7)$  is [France, Lawyer, Flu]. All such assignments are possible to the attacker because they all produce the same “view”, i.e.,  $R_1$  and  $R_2$ . Detecting correspondence attacks assuming this view of the attacker is non-trivial.

## 1.3 Contributions

In this paper, we formalize the notion of correspondence attacks and present an approach to prevent such attacks. We focus on answering several key questions:

- *Given that there are many possible ways of assigning corresponding pairs, and each may lead to a different inference, what should the attacker assume while cracking a record?* We present a model of correspondence attacks to address this issue and show that the continuous data publishing problem subsumes the case with multiple including recipients (Section 3).

- *What are exactly the records that can be cracked based on  $R_1$  and  $R_2$ ?* We systematically characterize the set of cracked records by correspondence attacks and propose the notion of *BCF-anonymity* to measure anonymity assuming this power of the attacker (Section 4).
- *Can  $R_2$  be anonymized such that  $R_2$  satisfies BCF-anonymity yet remains useful?* We show that the optimal BCF-anonymization is NP-hard. Then, we develop a practically efficient algorithm to determine a BCF-anonymized  $R_2$  (Section 5), study its data quality (Section 6), and extend the proposed approach to deal with more than two releases and other privacy notions (Section 7).

## 2. RELATED WORK

Anonymizing continuous data is a challenging problem because the attacker has increasingly more knowledge about the previously published data. Some anonymization methods [2, 10, 14, 17] have been proposed recently.

[2] is an early study which investigates the continuous data publishing scenario, but the anonymization method relies on delaying records release and the delay can be unbounded. Consequently, records could not be released in a timely fashion. In our method, records collected at timestamp  $T_i$  are *always* published in the corresponding release  $R_i$  without delay. Moreover, [2] guarantees a relatively weak privacy notion, which requires each equivalence class to contain at least  $\ell$  distinct sensitive values. This privacy notion is not safe if some sensitive values dominate an equivalence class. In contrast, our proposed approach can achieve anonymity, *confidence bounding* [15],  $(\alpha, k)$ -anonymity [16], and other safer notions of  $\ell$ -diversity [8].

[17] presents the first study to address both record insertions and deletions in data re-publication. It proposes a new privacy notion called *m-invariance*: if a record  $r$  has been published in releases  $R_i, \dots, R_j$  where  $i < j$ , then all *QID* groups containing  $r$  must have the same set of sensitive values, called the *signature* [17]. This will ensure the intersection of sensitive values over all such groups does not reduce the set of sensitive values. To maintain *m-invariance*, their method adds “counterfeit records” corresponding to infrequent sensitive values to make those equivalence classes have the same signature. Counterfeits, however, may not be acceptable in some cases. Suppose a pharmaceutical company wants to analyze patient reaction to certain drugs. Infrequent sensitive values such as the negative reactions are the most interesting ones and the target for research. However, with many counterfeit negative reactions which correspond to no real-life entities, it is difficult to deploy the results obtained from such data. Note that, even in the “insertion only” case, adding counterfeits is still necessary, for example, when a record with a new sensitive value is added. In contrast, our method guarantees data truthfulness at record level: each published record corresponds to a real-life entity.

Our previous work [14] studies the problem of anonymizing sequential releases where each subsequent release publishes a different subset of attributes for the same set of records. In contrast, this paper considers each release that combines new records with previously collected records over the same set of attributes. The attack and prevention mechanisms are very different in these two publishing models.

[10] considers the scenario that the Case-ID of records must be published. In our work, we consider the scenario that the data holder has removed the Case-ID of records, so

the attack based on Case-ID [10] does not occur. Instead, we deal with a new type of attacks even if no Case-ID is published. [5] proposes an efficient index structure to incrementally  $k$ -anonymize each individual release, but it does not address the correspondence attacks studied in this paper.

## 3. PROBLEM STATEMENTS

Every attribute has a finite set of *domain values*. Numeric values are pre-discretized into intervals. A *raw record* contains only domain values. Every attribute in *QID* has a taxonomy (tree) in which leaf nodes represent domain values and high level values generalize low level values. To generalize a table, we employ the global recoding scheme, i.e., globally replace all instances of some domain values with their ancestor value in the taxonomy of some attributes  $A_j$  in *QID*. A generalized attribute  $A_j$  can be represented by a “cut” through its taxonomy tree. A cut of a tree is a subset of values in the tree which contains exactly one value on each root-to-leaf path [4].  $R_1$  and  $R_2$  in Tables 1-2 are examples.

There are some other generalization schemes, such as multidimensional [7] and local recoding [10, 16, 17], that cause less data distortion, but these schemes make data analysis difficult. In local recoding, for example, some instances of Lawyer can be generalized to Professional while some instances of Lawyer remain ungeneralized. As a result, counting the number of lawyers becomes impossible because some lawyers are represented by professionals. Furthermore, many standard data mining methods, e.g., decision tree analysis, treat Lawyer and Professional as two independent values. Consequently, the decision tree may contain two branches, (Lawyer  $\rightarrow$  class1) and (Professional  $\rightarrow$  class2). It is unclear which branch should be used to classify a new lawyer.

In a  $k$ -anonymized table, records are partitioned into equivalence classes of size (i.e., the number of records) at least  $k$ . Each equivalence class contains all records having the same value on *QID*. We use *qid* to denote both a value on *QID* and the corresponding equivalence class.  $|qid|$  denotes the size of the equivalence class. A *group*  $g$  in an equivalence class *qid* consists of the records in *qid* that have the same value on the sensitive attribute. In other words, a group contains all records in the table that are indistinguishable wrt *QID* and the sensitive attribute. A person *matches* a (generalized) record in a table if her *QID* is either equal to or more specific than the record on every attribute in *QID*.

### 3.1 Correspondence Attacks

The data holder previously collected some data  $D_1$  timestamped  $T_1$  and published a  $k$ -anonymized version of  $D_1$ , called release  $R_1$ . Then the data holder collects new data  $D_2$  timestamped  $T_2$  and publishes a  $k$ -anonymized version of  $D_1 \cup D_2$ , called release  $R_2$ . An *attacker*, one of the recipients of  $R_1$  and  $R_2$ , attempts to identify the record of some *target* person, denoted by  $P$ , from  $R_1$  or  $R_2$ . We assume that the attacker is aware of  $P$ 's *QID* and timestamp. In addition, the attacker has the following *correspondence knowledge*: (1) Every record timestamped  $T_1$  (i.e., from  $D_1$ ) has a record in  $R_1$  and a record in  $R_2$ , called *corresponding records*. (2) Every record timestamped  $T_2$  (i.e., from  $D_2$ ) has a record in  $R_2$ , but not in  $R_1$ . Below is an intuition of the three possible attacks based on such knowledge.

**Forward-attack**, denoted by F-attack( $R_1, R_2$ ).  $P$  has timestamp  $T_1$  and the attacker tries to identify  $P$ 's record

in the *cracking release*  $R_1$  using the *background release*  $R_2$ . Since  $P$  has a record in  $R_1$  and a record in  $R_2$ , if a matching record  $r_1$  in  $R_1$  represents  $P$ , there must be a corresponding record in  $R_2$  that matches  $P$ 's *QID* and agrees with  $r_1$  on the sensitive attribute. If  $r_1$  fails to have such a corresponding record in  $R_2$ , then  $r_1$  does not originate from  $P$ 's *QID*, and therefore,  $r_1$  can be excluded from the possibility of  $P$ 's record. Scenario I is an example.

**Cross-attack**, denoted by  $C\text{-attack}(R_1, R_2)$ .  $P$  has timestamp  $T_1$  and the attacker tries to identify  $P$ 's record in the *cracking release*  $R_2$  using the *background release*  $R_1$ . Similar to F-attack, if a matching record  $r_2$  in  $R_2$  represents  $P$ , there must be a corresponding record in  $R_1$  that matches  $P$ 's *QID* and agrees with  $r_2$  on the sensitive attribute. If  $r_2$  fails to have such a corresponding record in  $R_1$ , then  $r_2$  either has timestamp  $T_2$  or does not originate from  $P$ 's *QID*, and therefore,  $r_2$  can be excluded from the possibility of  $P$ 's record. Scenario II is an example.

**Backward-attack**, denoted by  $B\text{-attack}(R_1, R_2)$ .  $P$  has timestamp  $T_2$  and the attacker tries to identify  $P$ 's record in the *cracking release*  $R_2$  using the *background release*  $R_1$ . In this case,  $P$  has a record in  $R_2$ , but not in  $R_1$ . Therefore, if a matching record  $r_2$  in  $R_2$  has to be the corresponding record of some record in  $R_1$ , then  $r_2$  has timestamp  $T_1$ , and therefore,  $r_2$  can be excluded from the possibility of  $P$ 's record. Scenario III is an example. Note that it is impossible to single out the matching records in  $R_2$  that have timestamp  $T_2$  but do not originate from  $P$ 's *QID* since all records at  $T_2$  have no corresponding record in  $R_1$ .

Table 3 summarizes all four possible combinations of cracking release ( $R_1$  or  $R_2$ ) and target  $P$ 's timestamp ( $T_1$  or  $T_2$ ). Note that if a target  $P$  has timestamp  $T_2$ ,  $P$  does not have a record in  $R_1$ , so it is impossible to crack  $P$ 's record in  $R_1$  in such a case and there are only three types of attacks.

All these attacks are based on making some inferences about corresponding records. There are many possible assignments of corresponding records and each assignment implies a possibly different underlying data ( $D'_1, D'_2$ ), not necessarily *the* underlying data ( $D_1, D_2$ ) collected by the data holder. Since all such underlying data ( $D'_1, D'_2$ ) generate the same published  $R_1$  and  $R_2$ , they are all possible to the attacker who knows about the data only through the published  $R_1$  and  $R_2$ . This observation motivates us to consider only the inferences that do not depend on a particular choice of a candidate ( $D'_1, D'_2$ ). First, let us define the space of such candidates underlying data for the published  $R_1$  and  $R_2$ .

Consider a record  $r$  in  $R_1$  or  $R_2$ . An *instantiation* of  $r$  is a raw record that agrees with  $r$  on the sensitive attribute and specializes  $r$  or agrees with  $r$  on *QID*. A *generator* of ( $R_1, R_2$ ) is an assignment, denoted by  $I$ , from the records in  $R_1 \cup R_2$  to their instantiations such that: for each record  $r_1$  in  $R_1$ , there is a distinct record  $r_2$  in  $R_2$  such that  $I(r_1) = I(r_2)$ ;  $(r_1, r_2)$  is called *buddies* under  $I$ . Duplicate records are treated as distinct records. The buddy relationship is *injective*: no two records have the same buddy. Every record in  $R_1$  has a buddy in  $R_2$  and exactly  $|R_1|$  records in  $R_2$  have a buddy in  $R_1$ . If  $r_2$  in  $R_2$  has a buddy in  $R_1$ ,  $I(r_2)$  has timestamp  $T_1$ ; otherwise  $I(r_2)$  has timestamp  $T_2$ . Intuitively, a generator represents an underlying data for ( $R_1, R_2$ ) and each pair of buddies represent corresponding records in the generator.

EXAMPLE 3.1. Refer to  $R_1$  and  $R_2$  in Tables 1-2. One generator has the buddies:

**Table 3: Types of correspondence attacks**

	Target $P$ 's timestamp	Cracking release	Background release
<b>F-attack</b>	$T_1$	$R_1$	$R_2$
<b>C-attack</b>	$T_1$	$R_2$	$R_1$
<b>B-attack</b>	$T_2$	$R_2$	$R_1$
<b>No attack</b>	$T_2$	$R_1$	$R_2$

$(a_1, b_1)$ , assigned to [UK, Lawyer, Flu]

$(a_2, b_2)$ , assigned to [UK, Lawyer, Flu]

$(a_3, b_3)$ , assigned to [UK, Lawyer, Flu]

$(a_4, b_4)$ , assigned to [France, Lawyer, HIV]

$(a_5, b_5)$ , assigned to [France, Lawyer, HIV].

$I(b_1)$ - $I(b_5)$  have timestamp  $T_1$ .  $b_6$ - $b_{10}$  can be assigned to any instantiation. Another generator has the buddies:

$(a_1, b_7)$ , assigned to [France, Lawyer, Flu]

$(a_2, b_2)$ , assigned to [UK, Lawyer, Flu]

$(a_3, b_8)$ , assigned to [France, Lawyer, Flu]

$(a_4, b_6)$ , assigned to [France, Lawyer, HIV]

$(a_5, b_9)$ , assigned to [UK, Lawyer, HIV].

$I(b_2)$ ,  $I(b_6)$ - $I(b_9)$  have timestamp  $T_1$ . Note that these generators give different underlying data. ■

Consider a record  $r$  in  $R_1$  or  $R_2$ . Suppose that for *some* generator  $I$ , the instantiation  $I(r)$  matches  $P$ 's *QID* and timestamp. In this case, excluding  $r$  means information loss for the purpose of attack because there is some underlying data for ( $R_1, R_2$ ) (given by  $I$ ) in which  $r$  is  $P$ 's record. On the other hand, suppose that for *no* generator  $I$  the instantiation  $I(r)$  can match  $P$ 's *QID* and timestamp. Then  $r$  *definitely* cannot be  $P$ 's record, so excluding  $r$  losses no information to the attacker. Our attack model is based on excluding such non-representing records.

For a target  $P$  with timestamp  $T_1$ , if  $P$  has a record in  $R_1$ ,  $P$  must match some  $qid_1$  in  $R_1$  and some  $qid_2$  in  $R_2$ . Therefore, we assume such a matching pair ( $qid_1, qid_2$ ) for  $P$ . Recall that a group  $g_i$  in an equivalence class  $qid_i$  consists of the records in  $qid_i$  that agree on the sensitive attribute. For a generator  $I$ ,  $I(g_i)$  denotes the set of records  $\{I(r_i) \mid r_i \in g_i\}$ . Below, we present the formal definition for each type of attacks.  $r_i, g_i, qid_i$  refer to records, groups and equivalence classes from  $R_i$ ,  $i = 1, 2$ .

### 3.1.1 F-attack

The F-attack seeks to identify as many as possible records in an equivalence class  $qid_1$  that do not represent  $P$  in *any* choice of the generator. Such *cracked records* definitely cannot be  $P$ 's record, and therefore, can be excluded. Since all records in a group are identical, the choice of cracked records in a group does not make a difference and determining the number of cracked records (i.e., the crack size) is sufficient to define the attack.

DEFINITION 3.1 (CRACK SIZE). Assume that a target  $P$  has timestamp  $T_1$  and matches ( $qid_1, qid_2$ ). A group  $g_1$  in  $qid_1$  has *crack size*  $c$  wrt  $P$  if  $c$  is maximal such that for every generator  $I$ , at least  $c$  records in  $I(g_1)$  do not match  $P$ 's *QID*. ■

If  $g_1$  has crack size  $c$ , at least  $c$  records in  $g_1$  can be excluded from the possibility of  $P$ 's record. On the other hand, with  $c$  being maximal, excluding more than  $c$  records will result in excluding some record that can possibly be  $P$ 's record. Therefore, the crack size is both the minimum and

the maximum number of records that can be excluded from  $g_1$  without any information loss for the purpose of attack.

EXAMPLE 3.2. Consider Scenario I in Example 1.1. Alice has  $QID=[\text{France, Lawyer}]$  and matches  $(qid_1, qid_2)$ , where  $qid_1 = [\text{Europe, Lawyer}] = \{a_1, a_2, a_3, a_4, a_5\}$   $qid_2 = [\text{France, Professional}] = \{b_4, b_5, b_6, b_7, b_8\}$ .  $qid_1$  has two groups:  $g_1 = \{a_1, a_2, a_3\}$  for Flu and  $g'_1 = \{a_4, a_5\}$  for HIV.  $g_1$  has crack size 1 wrt Alice because, for any generator  $I$ , at least one of  $I(a_1), I(a_2), I(a_3)$  does not match Alice's  $QID$ : if all of  $I(a_1), I(a_2), I(a_3)$  match Alice's  $QID$ ,  $R_2$  would have contained three buddies of the form  $[\text{France, Professional, Flu}]$ . The crack size is maximal since the second generator  $I$  in Example 3.1 shows that only one of  $I(a_1), I(a_2), I(a_3)$  (i.e.,  $I(a_2)$ ) does not match Alice's  $QID$ . ■

Definition 3.1 does not explain *how* to effectively determine the crack size, which is the topic in Section 4. For now, assuming that the crack size is known, we want to measure the anonymity after excluding the cracked records from each equivalence class. The F-anonymity below measures the minimum size of an equivalence class in  $R_1$  after excluding all records cracked by F-attack.

DEFINITION 3.2 (F-ANONYMITY). Let  $F(P, qid_1, qid_2)$  be the sum of the crack sizes for all groups in  $qid_1$  wrt  $P$ .  $F(qid_1, qid_2)$  denotes the maximum  $F(P, qid_1, qid_2)$  for any target  $P$  that matches  $(qid_1, qid_2)$ .  $F(qid_1)$  denotes the maximum  $F(qid_1, qid_2)$  for all  $qid_2$  in  $R_2$ . The *F-anonymity* of  $(R_1, R_2)$ , denoted by  $FA(R_1, R_2)$  or  $FA$ , is the minimum  $(|qid_1| - F(qid_1))$  for all  $qid_1$  in  $R_1$ . ■

### 3.1.2 C-attack

DEFINITION 3.3 (CRACK SIZE). Assume that a target  $P$  has timestamp  $T_1$  and matches  $(qid_1, qid_2)$ . A group  $g_2$  in  $qid_2$  has *crack size*  $c$  wrt  $P$  if  $c$  is maximal such that for every generator  $I$ , at least  $c$  records in  $I(g_2)$  do not match either  $P$ 's timestamp or  $P$ 's  $QID$ . ■

EXAMPLE 3.3. Consider Scenario II in Example 1.1. Alice has  $QID=[\text{France, Lawyer}]$  and matches  $(qid_1, qid_2)$  where  $qid_1 = [\text{Europe, Lawyer}] = \{a_1, a_2, a_3, a_4, a_5\}$   $qid_2 = [\text{France, Professional}] = \{b_4, b_5, b_6, b_7, b_8\}$ .  $qid_2$  has two groups:  $g_2 = \{b_7, b_8\}$  for Flu, and  $g'_2 = \{b_4, b_5, b_6\}$  for HIV.  $g'_2$  has crack size 1 wrt Alice since, for any generator  $I$  at least one of  $I(b_4), I(b_5), I(b_6)$  does not match Alice's timestamp or  $QID$ ; otherwise,  $R_1$  would have contained three buddies of the form  $[\text{Europe, Lawyer, HIV}]$ . The crack size is maximal since the first generator  $I$  in Example 3.1 shows that  $I(b_4)$  and  $I(b_5)$  match Alice's timestamp and  $QID$ . ■

DEFINITION 3.4 (C-ANONYMITY). Let  $C(P, qid_1, qid_2)$  be the sum of crack sizes of all groups in  $qid_2$  wrt  $P$ .  $C(qid_1, qid_2)$  denotes the maximum  $C(P, qid_1, qid_2)$  for any target  $P$  that matches  $(qid_1, qid_2)$ .  $C(qid_2)$  denotes the maximum  $C(qid_1, qid_2)$  for all  $qid_1$  in  $R_1$ . The *C-anonymity* of  $(R_1, R_2)$ , denoted by  $CA(R_1, R_2)$  or  $CA$ , is the minimum  $(|qid_2| - C(qid_2))$  for all  $qid_2$  in  $R_2$ . ■

### 3.1.3 B-attack

A target  $P$  for B-attack has timestamp  $T_2$ , thus, does not have to match any  $qid_1$  in  $R_1$ .

DEFINITION 3.5 (CRACK SIZE). Assume that a target  $P$  has timestamp  $T_2$  and matches  $qid_2$  in  $R_2$ . A group  $g_2$  in  $qid_2$  has *crack size*  $c$  wrt  $P$  if  $c$  is maximal such that for every generator  $I$ , at least  $c$  records in  $I(g_2)$  have timestamp  $T_1$ . ■

EXAMPLE 3.4. Consider Scenario III in Example 1.1. Alice has timestamp  $T_2$  and  $QID=[\text{UK, Lawyer}]$ .  $qid_2=[\text{UK, Professional}]$  consists of  $g_2 = \{b_1, b_2, b_3\}$  for Flu and  $g'_2 = \{b_9, b_{10}\}$  for HIV.  $g_2$  has crack size 1 wrt Alice. For every generator  $I$ , at least one of  $I(b_1), I(b_2), I(b_3)$  has timestamp  $T_1$ ; otherwise, one of  $a_1, a_2, a_3$  would have no buddy in  $R_2$ . The crack size is maximal since the second generator  $I$  in Example 3.1 shows that only  $I(b_2)$ , among  $I(b_1), I(b_2), I(b_3)$ , has timestamp  $T_1$ . ■

DEFINITION 3.6 (B-ANONYMITY). Let  $B(P, qid_2)$  be the sum of the crack sizes of all groups in  $qid_2$  wrt  $P$ .  $B(qid_2)$  denotes the maximum  $B(P, qid_2)$  for any target  $P$  that matches  $qid_2$ . The *B-anonymity* of  $(R_1, R_2)$ , denoted by  $BA(R_1, R_2)$  or  $BA$ , is the minimum  $(|qid_2| - B(qid_2))$  for all  $qid_2$  in  $R_2$ . ■

## 3.2 Two Problems

A *BCF-anonymity requirement* states that *all* of  $BA$ ,  $CA$  and  $FA$  are equal to or larger than some data-holder-specified threshold. We study two problems. The first problem checks whether a BCF-anonymity requirement is satisfied, assuming the input (i.e.,  $R_1$  and  $R_2$ ) as viewed by the attacker.

DEFINITION 3.7 (DETECTION). Given  $R_1$  and  $R_2$ , as described above, the *BCF-detection problem* is to determine whether a BCF-anonymity requirement is satisfied. ■

The second problem is to produce a generalized  $R_2$  that satisfies a given BCF-anonymity requirement and remains useful. This problem assumes the input as viewed by the data holder, that is,  $R_1, D_1$  and  $D_2$ , and uses an information metric to measure the usefulness of the generalized  $R_2$ . Examples are *discernibility cost* [12] and *data distortion* [11].

DEFINITION 3.8 (ANONYMIZATION). Given  $R_1, D_1$  and  $D_2$ , as described above, the *BCF-anonymization problem* is to generalize  $R_2 = D_1 \cup D_2$  so that  $R_2$  satisfies a given BCF-anonymity requirement and remains as useful as possible wrt a specified information metric. ■

In the special case of empty  $D_1$ , F-attack and C-attack do not happen and B-anonymity coincides with  $k$ -anonymity of  $R_2$  for  $D_2$ . Since the optimal  $k$ -anonymization is NP-hard [9], the optimal BCF-anonymization is NP-hard.

So far, we have assumed that both  $R_1$  and  $R_2$  are received by one recipient. In the special case of empty  $D_2$ ,  $R_1$  and  $R_2$  are two different generalized versions of the same data  $D_1$  to serve different information requirements or different recipients. In this case, there are potentially multiple attackers. What happens if the attackers collude together? The collusion problem may seem to be very different. Indeed, Definitions 3.7-3.8 subsume the collusion problem. Consider the worst-case collusion scenario in which *all* recipients collude together by sharing *all* of their received data. This scenario is equivalent to publishing all releases to one attacker.

## 4. DETECTION

The key to the BCF-detection problem is computing the crack size in Definitions 3.1, 3.3, 3.5. We present a method

for computing the crack size of a group. Our insight is that if a record  $r$  represents the target  $P$  for some generator, its buddy in the other release (i.e., the corresponding record) must satisfy some conditions. If such conditions fail,  $r$  does not represent  $P$  for that generator. One of the conditions is the following “comparable” relationship.

**DEFINITION 4.1.** For  $qid_1$  in  $R_1$  and  $qid_2$  in  $R_2$ ,  $(qid_1, qid_2)$  are *comparable* if for every attribute  $A$  in  $QID$ ,  $qid_1[A]$  and  $qid_2[A]$  are on the same path in the taxonomy of  $A$ . For a record in  $r_1$  (or a group) in  $qid_1$  and a record  $r_2$  (or a group) in  $qid_2$ ,  $(r_1, r_2)$  are *comparable* if they agree on the sensitive attribute and  $(qid_1, qid_2)$  are comparable. For comparable  $(qid_1, qid_2)$ ,  $CG(qid_1, qid_2)$  denotes the set of group pairs  $\{(g_1, g_2)\}$ , where  $g_1$  and  $g_2$  are groups in  $qid_1$  and  $qid_2$  for the same sensitive value and there is one pair  $(g_1, g_2)$  for each sensitive value (unless both  $g_1$  and  $g_2$  are empty). ■

Essentially, being comparable means sharing a common instantiation. For example,  $qid_1 = [\text{Europe, Lawyer}]$  and  $qid_2 = [\text{UK, Professional}]$  are comparable, but  $qid_1 = [\text{Europe, Lawyer}]$  and  $qid_2 = [\text{Canada, Professional}]$  are not. It is easy to see that if a target  $P$  matches  $(qid_1, qid_2)$ ,  $(qid_1, qid_2)$  are comparable; if  $(r_1, r_2)$  are buddies (for some generator),  $(r_1, r_2)$  are comparable; comparable  $(r_1, r_2)$  can be assigned to be buddies (because of sharing a common instantiation). The following fact can be verified:

**THEOREM 4.1.** Suppose that  $P$  matches  $(qid_1, qid_2)$  and that  $(r_1, r_2)$  are buddies for a generator  $I$ . If  $I(r_1)$  and  $I(r_2)$  match  $P$ 's  $QID$ , then  $r_1$  is in  $g_1$  if and only if  $r_2$  is in  $g_2$ , where  $(g_1, g_2)$  is in  $CG(qid_1, qid_2)$ . ■

Theorem 4.1 follows because buddies agree on the sensitive attribute and  $I(r_1)$  and  $I(r_2)$  matching  $P$ 's  $QID$  implies that  $r_1$  is in  $qid_1$  and  $r_2$  is in  $qid_2$ . The next two lemmas are used to derive an upper bound on crack size. The first states some transitivity of the “comparable” relationship, which will be used to construct a required generator in the upper bound proof.

**LEMMA 4.1 (3-HOP TRANSITIVITY).** Let  $r_1, r'_1$  be in  $R_1$  and  $r_2, r'_2$  be in  $R_2$ . If each of  $(r'_1, r_2)$ ,  $(r_2, r_1)$  and  $(r_1, r'_2)$  is comparable,  $(r'_1, r'_2)$  is comparable.

**PROOF.** The three comparable pairs imply that  $r'_1$  and  $r'_2$  agree on the sensitive attribute. We show that  $r'_1[A]$  and  $r'_2[A]$  are on the same path of the taxonomy for every attribute  $A$  in  $QID$ . Recall that the generalization in  $A$  forms a cut of the taxonomy of  $A$ . Therefore, with both  $r_1$  and  $r'_1$  being comparable to  $r_2$ , either  $r_1[A] = r'_1[A]$  or both are more specific than  $r_2[A]$ . Similarly, either  $r_2[A] = r'_2[A]$  or both are more specific than  $r_1[A]$ . At least one of these equalities must hold; otherwise  $r_1[A]$  is both more general than and more specific than  $r_2[A]$ , which is impossible. If  $r_1[A] = r'_1[A]$  holds, the proof is done because  $r_1[A]$  and  $r'_2[A]$  are on the same path. If  $r_2[A] = r'_2[A]$  holds, the proof is done because  $r'_1[A]$  and  $r_2[A]$  are on the same path. □

The following is the key lemma for proving the upper bound of crack size.

**LEMMA 4.2.** Let  $(g_1, g_2)$  be in  $CG(qid_1, qid_2)$ . There exists a generator in which exactly  $\min(|g_1|, |g_2|)$  records in  $g_1$  have a buddy in  $g_2$ .

**PROOF.** Consider any generator  $I$ . Since the buddy relationship is injective, at most  $\min(|g_1|, |g_2|)$  records in  $g_1$  have a buddy in  $g_2$ . Let  $X$  contain all such records and possibly other records in  $g_1$  so that  $|X| = \min(|g_1|, |g_2|)$ . For each  $r_1$  in  $X$  that has a buddy  $r'_2$  not in  $g_2$ , we create a new buddy  $r_2$  that is in  $g_2$ . Since less than  $\min(|g_1|, |g_2|)$  records in  $g_1$  have a buddy in  $g_2$ , some record  $r_2$  in  $g_2$  does not have a buddy in  $g_1$ . If  $r_2$  has no buddy, let  $r_2$  be the new buddy of  $r_1$ . Assume that  $r_2$  has a buddy  $r'_1$ . Note that  $r'_1$  is not in  $g_1$ . Each of  $(r'_1, r_2)$ ,  $(r_2, r_1)$  and  $(r_1, r'_2)$  is comparable (because buddies are comparable). From Lemma 4.1,  $(r'_1, r'_2)$  is comparable. Therefore, we can swap the buddies of  $r_1$  and  $r'_1$ , that is, make  $(r_1, r_2)$  and  $(r'_1, r'_2)$  the new buddies. Note that every record in  $X$  that previously has a buddy in  $g_2$  is not affected by the swapping. □

Now we determine the crack size for each type of attack.

## 4.1 F-attack

Assume that  $P$  matches  $(qid_1, qid_2)$ . Consider a group pair  $(g_1, g_2)$  in  $CG(qid_1, qid_2)$ . Since the buddy relationship is injective, if  $g_1$  contains more records than  $g_2$ , i.e.,  $|g_1| > \min(|g_1|, |g_2|)$ , at least  $|g_1| - \min(|g_1|, |g_2|)$  records in  $g_1$  do not have a buddy in  $g_2$  for *any* generator. According to Theorem 4.1, these records do not originate from  $P$ 's  $QID$  for any generator. Thus the crack size of  $g_1$  is at least  $|g_1| - \min(|g_1|, |g_2|)$  (i.e., a lower bound). On the other hand, according to Lemma 4.2, there exists *some* generator in which *exactly*  $|g_1| - \min(|g_1|, |g_2|)$  records in  $g_1$  do not have a buddy in  $g_2$ ; according to Theorem 4.1, these records do not originate from  $P$ 's  $QID$  for any generator. By Definition 3.1,  $|g_1| - \min(|g_1|, |g_2|)$  is the crack size of  $g_1$ .

**THEOREM 4.2.** Suppose that a target  $P$  matches  $(qid_1, qid_2)$ . Let  $(g_1, g_2)$  be in  $CG(qid_1, qid_2)$ . (1)  $g_1$  has crack size  $c$  wrt  $P$ , where  $c = |g_1| - \min(|g_1|, |g_2|)$ . (2)  $F(qid_1, qid_2) = \sum c$ , where  $\sum$  is over  $(g_1, g_2)$  in  $CG(qid_1, qid_2)$  and  $g_1$  has the crack size  $c$  determined in (1). ■

*Remarks.*  $F(P, qid_1, qid_2)$  is the same for all targets  $P$  that match  $(qid_1, qid_2)$ , i.e.,  $F(qid_1, qid_2)$  computed by Theorem 4.2. To compute  $FA$ , we compute  $F(qid_1, qid_2)$  for all comparable  $(qid_1, qid_2)$ . This requires partitioning the records into equivalence classes and groups, which can be done by sorting the records on all attributes. The F-attack happens when  $|g_1| > \min(|g_1|, |g_2|)$ , that is,  $g_1$  contains too many records for their buddies to be contained in  $g_2$ . This could be the case if  $R_2$  has less generalization due to additional records at timestamp  $T_2$ .

**EXAMPLE 4.1.** Continue with Example 3.2. Alice matches  $qid_1 = [\text{Europe, Lawyer}]$  and  $qid_2 = [\text{France, Professional}]$ .  $qid_1$  consists of  $g_1 = \{a_1, a_2, a_3\}$  for Flu and  $g'_1 = \{a_4, a_5\}$  for HIV.  $qid_2$  consists of  $g_2 = \{b_7, b_8\}$  for Flu and  $g'_2 = \{b_4, b_5, b_6\}$  for HIV.  $CG(qid_1, qid_2) = \{(g_1, g_2), (g'_1, g'_2)\}$ .  $|g_1| = 3$ ,  $|g_2| = 2$ ,  $|g'_1| = 2$  and  $|g'_2| = 3$ . So  $g_1$  has crack size 1 and  $g'_1$  has crack size 0. ■

## 4.2 C-attack

By a similar argument, at least  $|g_2| - \min(|g_1|, |g_2|)$  records in  $g_2$  do not have a buddy in  $g_1$  in *any* generator, so the crack size of  $g_2$  is at least  $|g_2| - \min(|g_1|, |g_2|)$ . According to Lemma 4.2, for some generator exactly  $\min(|g_1|, |g_2|)$  records in  $g_1$  have a buddy in  $g_2$ , therefore, exactly  $\min(|g_1|, |g_2|)$

records in  $g_2$  have a buddy in  $g_1$  due to the injective buddy relationship. So, exactly  $|g_2| - \min(|g_1|, |g_2|)$  records in  $g_2$  do not have a buddy in  $g_1$ . According to Theorem 4.1, these records in  $g_2$  do not represent  $P$ . By Definition 3.3, we have

**THEOREM 4.3.** Suppose that a target  $P$  matches  $(qid_1, qid_2)$ . Let  $(g_1, g_2)$  be in  $CG(qid_1, qid_2)$ . (1)  $g_2$  has crack size  $c$  wrt  $P$ , where  $c = |g_2| - \min(|g_1|, |g_2|)$ . (2)  $C(qid_1, qid_2) = \sum c$ , where  $\sum$  is over  $(g_1, g_2)$  in  $CG(qid_1, qid_2)$  and  $g_2$  has the crack size  $c$  determined in (1). ■

The condition  $|g_2| > \min(|g_1|, |g_2|)$  says that  $g_2$  contains too many records even if all the records in  $g_1$  have a buddy in  $g_2$ . In this case, some record in  $g_2$  either comes from  $D_2$  or has a buddy not in  $g_1$ . Such records either do not match  $P$ 's timestamp (i.e.,  $T_1$ ) or do not originate from  $P$ 's  $QID$ . It is interesting to note the condition  $|g_1| - \min(|g_1|, |g_2|) > 0$  for F-attack and the condition  $|g_2| - \min(|g_1|, |g_2|) > 0$  for B-attack are exactly opposite. More interestingly, in Section 4.4 we will show that  $FA \geq k$  is violated exactly when  $CA \geq k$  is violated.

**EXAMPLE 4.2.** Continue with Example 3.3. Alice matches  $qid_1 = [\text{Europe, Lawyer}]$  and  $qid_2 = [\text{France, Professional}]$ .  $qid_1$  consists of  $g_1 = \{a_1, a_2, a_3\}$  and  $g'_1 = \{a_4, a_5\}$ .  $qid_2$  consists of  $g_2 = \{b_7, b_8\}$  and  $g'_2 = \{b_4, b_5, b_6\}$ .  $CG(qid_1, qid_2) = \{(g_1, g_2), (g'_1, g'_2)\}$ .  $|g_2| = 2$ ,  $|g_1| = 3$ ,  $|g'_2| = 3$  and  $|g'_1| = 2$ . Thus  $g_2$  has crack size 0 and  $g'_2$  has crack size 1. ■

### 4.3 B-attack

Suppose that  $P$  matches some  $qid_2$  in  $R_2$ . Let  $g_2$  be a group in  $qid_2$ . The crack size of  $g_2$  is related to the number of records in  $g_2$  that have a buddy in  $R_1$  (thus timestamp  $T_1$ ). Let  $G_1$  denote the set of records in  $R_1$  comparable to  $g_2$ . So  $G_1$  contains all the records in  $R_1$  that *can* have a buddy in  $g_2$ . Let  $G_2$  denote the set of records in  $R_2$  comparable to some record in  $G_1$ . The next lemma implies that all records in  $G_1$  *can* have a buddy in  $G_2 - g_2$ .

**LEMMA 4.3.** Every record in  $G_2$  is comparable to all records in  $G_1$  and only those records in  $G_1$ .

**PROOF.** Each record  $r_2$  in  $G_2$  is comparable to some record  $r'_1$  in  $G_1$  and both  $r'_1$  and  $r_1$  are comparable to  $g_2$ , where  $r_1$  is *any* record in  $G_1$ . It then follows from Lemma 4.1 that  $(r_1, r_2)$  are comparable. For the second part, suppose that a record  $r_2$  in  $G_2$  is comparable to some record  $r_1$  in  $R_1$ . Note that  $r_2$  is comparable to some record  $r'_1$  in  $G_1$ , which is comparable to  $g_2$ . Lemma 4.1 then implies that  $r_1$  is comparable to  $g_2$ . By the definition of  $G_1$ ,  $r_1$  is in  $G_1$ . □

From Lemma 4.3, all records in  $G_1$  and only those records in  $G_1$  can have a buddy in  $G_2$ . Each record in  $G_1$  has its buddy either in  $g_2$  or in  $G_2 - g_2$ , but not in both. If  $|G_1| > |G_2| - |g_2|$ , the remaining  $c = |G_1| - (|G_2| - |g_2|)$  records in  $G_1$  must have their buddies in  $g_2$ , or equivalently,  $c$  records in  $g_2$  must have their buddies in  $R_1$  (therefore, timestamp  $T_1$ ). The next theorem follows from this observation. Note that for multisets,  $|G_2| - |g_2| = |G_2 - g_2|$ . Also,  $c \leq |g_2|$  because  $|G_1| \leq |G_2|$  and all records in  $G_1$  must have its buddy in  $G_2$  due to the injective buddy relationship.

**THEOREM 4.4.** Suppose that a target  $P$  has timestamp  $T_2$  and matches  $qid_2$  in  $R_2$ . Let  $g_2$  in  $qid_2$ . (1) If  $|G_2| < |g_2|$ ,  $g_2$  has crack size 0 wrt  $P$ . (2) If  $|G_2| \geq |g_2|$ ,  $g_2$  has

crack size  $c$ , where  $c = \max(0, |G_1| - (|G_2| - |g_2|))$ . (3)  $B(qid_2) = \sum c$ , where  $\sum$  is over  $g_2$  in  $qid_2$  and  $g_2$  has the crack size  $c$  determined in (1) and (2).

**PROOF.** (3) follows from Definition 3.6. For (1),  $|G_2| < |g_2|$  implies  $|G_1| = 0$  (otherwise  $G_2$  contains at least all the records in  $g_2$  and  $|G_2| \geq |g_2|$ ). This means that the records in  $g_2$  have no buddy (for any generator), thus, have timestamp  $T_2$ . In this case,  $g_2$  has crack size 0 wrt  $P$ . We prove (2): From the discussion above, at least  $c$  (as in the theorem) records in  $g_2$  have timestamp  $T_1$  (for any generator). To show that  $c$  is also an upper bound, we construct a generator in which exactly  $c$  records in  $g_2$  have timestamp  $T_1$ . In this generator, exactly  $c$  records in  $G_1$  have their buddies in  $g_2$  and the remaining  $|G_1| - c$  records in  $G_1$  have their buddies in  $G_2 - g_2$ . According to Lemma 4.3, any record in  $G_2$  can be the buddy of any record in  $G_1$ . So we only need to show that  $g_2$  contains at least  $c$  records and that  $G_2 - g_2$  contains at least  $|G_1| - c$  records. The first part follows from  $|G_1| \leq |G_2|$ . For the second part, if  $c > 0$ ,  $c = |G_1| - (|G_2| - |g_2|)$ , so  $|G_1| - c = |G_2| - |g_2|$ ; if  $c = 0$ ,  $|G_1| - (|G_2| - |g_2|) \leq 0$ , so  $|G_2| - |g_2| \geq |G_1| - c$ . In both cases,  $G_2 - g_2$  contains at least  $|G_1| - c$  records. □

$|G_1| - (|G_2| - |g_2|) > 0$  occurs if  $G_2 - g_2$  is so small that some records in  $G_1$  have to find their buddies in  $g_2$ . This condition is likely to occur for a small  $D_2$  where most records in  $R_2$  have timestamp  $T_1$ , thus, are buddies of the records in  $G_1$ . Our experiments also confirmed this observation. In the special case of an empty  $D_2$ , all records in  $g_2$  have timestamp  $T_1$  and  $g_2$  has crack size  $|g_2|$ . This can be derived from Theorem 4.4(2). In this special case,  $|G_1| = |G_2|$  because every record in  $G_2$  has a buddy in  $R_1$ , in particular, in  $G_1$ , according to Lemma 4.3.

**EXAMPLE 4.3.** Continue with Example 3.4. Alice has  $[\text{UK, Lawyer}]$  and timestamp  $T_2$ .  $qid_2 = [\text{UK, Professional}]$  has  $g_2$  for Flu and  $g'_2$  for HIV:

$$\begin{aligned} g_2 &= \{b_1, b_2, b_3\}, G_1 = \{a_1, a_2, a_3\}, G_2 = \{b_1, b_2, b_3, b_7, b_8\} \\ |G_1| - (|G_2| - |g_2|) &= 3 - (5 - 3) = 1 \\ g'_2 &= \{b_9, b_{10}\}, G_1 = \{a_4, a_5\}, G_2 = \{b_4, b_5, b_6, b_9, b_{10}\} \\ |G_1| - (|G_2| - |g'_2|) &= 2 - (5 - 2) = -1 \end{aligned}$$

Thus  $g_2$  has crack size 1 and  $g'_2$  has crack size 0. ■

### 4.4 Equivalence of F-attack and C-attack

F-attack and C-attack are motivated under different scenarios and have a different characterization of crack size. Despite such differences, we show that these attacks are not independent of each other at all; in fact,  $FA = CA$ . First, we show the following lemma as a stepping stone.

**LEMMA 4.4.** For comparable  $(qid_1, qid_2)$ , where  $qid_i$  is in  $R_i$ ,  $|qid_1| - F(qid_1, qid_2) = |qid_2| - C(qid_1, qid_2)$ .

**PROOF.**  $F(qid_1, qid_2) = \sum (|g_1| - \min(|g_1|, |g_2|))$ , following Theorem 4.2, where  $\sum$  is over  $(g_1, g_2)$  in  $CG(qid_1, qid_2)$ . For any  $(g_1, g_2)$ , if  $|g_1| > |g_2|$ , let  $(g_1^a, g_2^a)$  denote the pair; if  $|g_1| \leq |g_2|$ , let  $(g_1^b, g_2^b)$  denote the pair. Note that  $\sum (|g_1| - \min(|g_1|, |g_2|)) = \sum (|g_1^a| - |g_2^a|)$ , and  $|qid_1| = \sum |g_1^a| + \sum |g_1^b|$ . So we have:

$$\begin{aligned} |qid_1| - F(qid_1, qid_2) &= |qid_1| - \sum (|g_1| - \min(|g_1|, |g_2|)) = |qid_1| - \sum (|g_1^a| - |g_2^a|) \\ &= \sum |g_1^a| + \sum |g_1^b| - \sum |g_1^a| + \sum |g_2^a| = \sum |g_1^b| + \sum |g_2^a|. \end{aligned}$$

From Theorem 4.3,  $C(qid_1, qid_2) = \sum (|g_2| - \min(|g_1|, |g_2|))$ , where  $\sum$  is over  $(g_1, g_2)$  in  $CG(qid_1, qid_2)$ . By a similar rewriting, we have  $|qid_2| - C(qid_1, qid_2) = \sum |g_2^a| + \sum |g_1^b|$ . □

THEOREM 4.5.  $FA = CA$ .

PROOF. Refer to Definition 3.2 of  $FA$  and Definition 3.4 of  $CA$ . Let  $qid'_1$  and  $qid'_2$  be such that  $FA = |qid'_1| - F(qid'_1, qid'_2)$ . From Lemma 4.4,  $FA = |qid'_2| - C(qid'_1, qid'_2)$ . By the definition of  $CA$ ,  $CA \geq |qid'_2| - C(qid'_1, qid'_2) = FA$ . Let  $qid^*_1$  and  $qid^*_2$  be such that  $CA = |qid^*_2| - C(qid^*_1, qid^*_2)$ . From Lemma 4.4,  $CA = |qid^*_1| - F(qid^*_1, qid^*_2)$ . By the definition of  $FA$ ,  $FA \geq |qid^*_1| - F(qid^*_1, qid^*_2) = CA$ . Thus  $FA = CA$ .  $\square$

At first glance,  $FA = CA$  seems contradicting Theorem 4.2 and Theorem 4.3 that suggest exactly opposite conditions for F-attack and C-attack, that is, (1)  $|g_1| - |g_2| > 0$  for F-attack and (2)  $|g_2| - |g_1| > 0$  for C-attack. A closer look tells that there is no contradiction. Consider how condition (1) affects  $FA$  and  $CA$ . Recall that  $FA$  is the minimum ( $|qid_1| - F(qid_1, qid_2)$ ) and  $CA$  is the minimum ( $|qid_2| - C(qid_1, qid_2)$ ). The occurrence of condition (1) decreases  $FA$  by increasing  $F(qid_1, qid_2)$ . However, the occurrence of condition (1) also decreases  $|qid_2|$ , thus  $CA$ . So condition (1) causes both  $FA$  and  $CA$  to decrease. Similarly, condition (2) causes both  $FA$  and  $CA$  to decrease. In fact, from the proof of Lemma 4.4, both  $FA$  and  $CA$  are contributed by the same source, i.e.,  $\sum |g_2^i| + \sum |g_1^i|$ .

## 5. ANONYMIZATION

We now present an algorithm for anonymizing  $R_2$  for  $D_1 \cup D_2$  to satisfy all of  $FA \geq k, CA \geq k, BA \geq k$ . Our approach iteratively specializes  $R_2$  starting from the *most generalized state* of  $R_2$ . In the most generalized state, all values for each attribute  $A_j \in QID$  are generalized to the top most value in the taxonomy. Each *specialization*, for some attribute in  $QID$ , replaces a parent value with an appropriate child value in every record containing the parent value. Section 5.1 shows that  $FA, CA$ , and  $BA$  are non-increasing in this specialization process. Therefore, all further specializations can be pruned once any of the above requirements is violated. Section 5.2 presents an efficient algorithm for producing such a maximally specialized  $R_2$ .

### 5.1 Anti-Monotonicity of BCF-Anonymity

THEOREM 5.1. Each of  $FA, CA$  and  $BA$  is non-increasing with respect to a specialization on  $R_2$ .

PROOF. From Theorem 4.5,  $FA = CA$ , so we show the theorem only for  $FA$  and  $BA$ . Consider a specialization on  $R_2$  in which  $qid_2$  is specialized into  $qid_2^1, \dots, qid_2^z$ , each corresponding to a child value. Each group  $g_2$  in  $qid_2$  is also specialized into  $g_2^1, \dots, g_2^z$ , thus,  $g_2^j \leq g_2$ . Note that the specialization does not affect  $|qid_1|$  for any  $qid_1$  in  $R_1$ .

$FA$ : Recall that  $FA$  is the minimum ( $|qid_1| - F(qid_1)$ ) over all  $qid_1$  in  $R_1$ .  $F(qid_1) = \max(F(qid_1, qid_2))$  over all  $qid_2$  comparable to  $qid_1$ .  $F(qid_1, qid_2) = \sum c$ , where  $\sum$  is over  $(g_1, g_2)$  in  $CG(qid_1, qid_2)$  and  $c = |g_1| - \min(|g_1|, |g_2|)$  given by Theorem 4.2. We show that  $F(qid_1)$  is non-decreasing wrt the above specialization. If  $qid_2$  is not comparable to  $qid_1$ ,  $qid_2^i$ 's remain not comparable to  $qid_1$ . Assume that  $qid_2$  is comparable to  $qid_1$ . In this case, the term  $F(qid_1, qid_2)$  in  $\max(F(qid_1, qid_2))$  is replaced with the terms  $F(qid_1, qid_2^i)$  for those  $qid_2^i$  comparable to  $qid_1$ . For each such  $qid_2^i$ ,  $F(qid_1, qid_2) \leq F(qid_1, qid_2^i)$  since  $|g_1| - \min(|g_1|, |g_2|) \leq |g_1| - \min(|g_1|, |g_2^i|)$ , where  $(g_1, g_2)$  is in  $CG(qid_1, qid_2)$

and  $(g_1, g_2^i)$  is in  $CG(qid_1, qid_2^i)$ . Therefore,  $F(qid_1)$  is non-decreasing wrt the specialization.

$BA$ : Recall that  $BA$  is the minimum ( $|qid_2| - B(qid_2)$ ) over all  $qid_2$  in  $R_2$ .  $B(qid_2) = \sum c$ , where  $\sum$  is over the groups  $g_2$  in  $qid_2$  and  $c$  is the crack size of  $g_2$ . Unlike  $FA$ , the specialization on  $R_2$  decreases both  $|qid_2|$  and  $B(qid_2)$ . We show that the decrease of  $|qid_2|$  is *at least as much as* the decrease of  $B(qid_2)$ , which implies that  $|qid_2^i| - B(qid_2^i) \leq |qid_2| - B(qid_2)$ . Consider obtaining the specialized equivalence class  $qid_2^i$  from  $qid_2$  by discarding all groups  $g_2^j$  of  $qid_2^i$ ,  $j \neq i$ . Discarding  $g_2^j$  decreases  $|qid_2|$  by  $|g_2^j|$  and decreases  $B(qid_2)$  by at most  $c^j$ , where  $c^j$  is the crack size of  $g_2^j$ . Note that  $c^j \leq |g_2^j|$ . Therefore, the decrease of  $|qid_2|$  is at least as much as that of  $B(qid_2)$ .  $\square$

COROLLARY 5.1. For a given requirement on BCF-anonymity, there exists a generalized  $R_2$  that satisfies the requirement if and only if the most generalized  $R_2$  does.  $\blacksquare$

### 5.2 Algorithm

Finding an optimal BCF-anonymized  $R_2$  is NP-hard. Our approach *BCF-anonymizer*, summarized in Algorithm 1, aims at producing a maximally specialized (suboptimal) BCF-anonymized  $R_2$  which any further specialization leads to a violation. It starts with the most generalized  $R_2$ . At any time,  $R_2$  contains the generalized records of  $D_1 \cup D_2$  and  $Cut_{2j}$  gives the generalization cut for  $A_j \in QID$ . Each equivalence class  $qid_2$  in  $R_2$  is associated with a set of groups  $g_2$  with stored  $|g_2|$ . Each group  $g_2$  is associated with the set of raw records in  $D_1 \cup D_2$  generalized to the group.  $R_1$  is represented similarly with  $|qid_1|$  and  $|g_1|$  stored, except that no raw record is kept for  $g_1$ .  $Cut_{1j}$  contains the generalization cut  $A_j \in QID$  in  $R_1$ .  $Cut_{1j}$  never change once created.

Initially,  $\cup Cut_{2j}$  and the candidate list contain the most general value  $ANY_j$  for every  $A_j \in QID$  (Lines 1-3). In each iteration, we examine the first valid candidate specialization ranked by a criterion *Score*. If the candidate  $w$  is *valid*, that is, not violating the BCF-anonymity after its specialization, we specialize  $w$  on  $R_2$  (Lines 6-10); otherwise, we remove  $w$  from the candidate list (Line 12). This iteration is repeated until there is no more candidate. From Theorem 5.1, the returned  $R_2$  is maximal (suboptimal). *Score* ranks the candidates by their ‘‘information worth.’’ We employ the discernibility cost [12] which charges a penalty to each record for being indistinguishable from other records. For each record

---

#### Algorithm 1 BCF-anonymizer

---

**Input:**  $R_1, R_2 = D_1 \cup D_2, k$ , taxonomy for each  $A_j \in QID$ .  
**Output:** a BCF-anonymized  $R_2$ .

- 1: generalize every value for  $A_j \in QID$  in  $R_2$  to  $ANY_j$ ;
  - 2: let candidate list =  $\cup Cut_{2j}$  containing all  $ANY_j$ ;
  - 3: sort candidate list by *Score* in descending order;
  - 4: **while** the candidate list is not empty **do**
  - 5:   **if** the first candidate  $w$  in candidate list is valid **then**
  - 6:     specialize  $w$  into  $w_1, \dots, w_z$  in  $R_2$ ;
  - 7:     compute *Score* for all  $w_i$ ;
  - 8:     remove  $w$  from  $\cup Cut_{2j}$  and the candidate list;
  - 9:     add  $w_1, \dots, w_z$  to  $\cup Cut_{2j}$  and the candidate list;
  - 10:    sort the candidate list by *Score* in descending order;
  - 11:    **else**
  - 12:     remove  $w$  from the candidate list;
  - 13:    **end if**
  - 14: **end while**
  - 15: output  $R_2$  and  $\cup Cut_{2j}$ ;
-



in an equivalence class  $qid_2$ , this penalty is  $|qid_2|$ . To minimize the discernibility cost, we choose the specialization  $w$  that maximizes  $Score(w)$ , where  $Score(w) = \sum_{qid_w} |qid_w|^2$  over all  $qid_w$  containing  $w$ .

In general, Lines 5-7 require scanning all pairs  $(qid_1, qid_2)$  and all records in  $R_2$ , which is highly inefficient for a large data set. We present an incremental computation of  $FA$ ,  $CA$ , and  $BA$  that examines only comparable pairs  $(qid_1, qid_2)$  and raw records in  $R_2$  that are *involved* in the current specialization. Let  $\{w_1, \dots, w_z\}$  be the set of child values of the winning candidate  $w$  on Line 6. Let  $qid_w$  be a  $qid_2$  containing  $w$  and let  $qid_{w_1}, \dots, qid_{w_z}$  be the new qids resulting from specializing  $qid_w$ . Let  $g_w$  be a group in  $qid_w$  and  $g_{w_i}$  be a group in  $qid_{w_i}$ .

**Line 6:** To perform the specialization of  $w$  efficiently, for each value  $v$  in  $\cup Cut_{2j}$ , we create  $Link[v]$  to link up all  $qid_2$  containing  $v$ .  $Link[w]$  provides a direct access to all equivalence classes  $qid_w$  and associated records to be specialized. After the specialization, we create new  $Link[w_i]$  to link up new  $qid_{w_i}$  containing  $w_i$ . In addition, we add  $qid_{w_i}$  to every existing  $Link[v]$  to which  $qid_w$  was previously linked, except for  $Link[w]$ . The overhead for maintaining these links is negligible. [4] shows that the time complexity of this specialization procedure is linear in  $|R_2|$ .

**Line 7:** In the same scan of  $Link[w_i]$ ,  $Score(w_i)$  is computed by  $\sum |qid_{w_i}|^2$  for all  $qid_{w_i}$  on  $Link[w_i]$ .

**Line 5:** To check the validity of  $w$ , we introduce a data structure, called  $GTree1$ , to index all comparable  $qid_1$  in  $R_1$  for a given  $qid_2$  in  $R_2$ .  $GTree1$  has one level per attribute in  $QID$  followed by the leaf level for the sensitive attribute. Each root-to-leaf path in  $GTree1$  represents a group  $g_1$  in  $R_1$ . To find all  $qid_1$  in  $GTree1$  that are comparable to  $qid_2$ , we traverse  $GTree1$  in a depth-first manner. At the level for an attribute  $A \in QID$ , if  $qid_2[A]$  and  $qid_1[A]$  are not on the same path in the taxonomy of  $A$ , the entire subtree below  $qid_1[A]$  is pruned. On reaching a node on the level  $|QID|$ , the  $qid_1$  at the node is comparable to  $qid_2$ .  $GTree1$  is static. Below, we update  $FA$  and  $BA$  using  $GTree1$ .

---

### Procedure 2 Updating $FA$

---

```

1: for all new  $qid_{w_i}$  on  $Link[w_i]$  do
2:   for all comparable  $qid_1$  do
3:     let  $F(qid_1) = \max(F(qid_1), F(qid_1, qid_{w_i}))$ ;
4:     let  $FA = \min(FA, |qid_1| - F(qid_1))$ ;
5:   end for
6: end for

```

---

Updating  $FA$ :  $FA$  is the minimum  $(|qid_1| - F(qid_1))$  over all  $qid_1$  in  $R_1$ , where  $F(qid_1) = \max(F(qid_1, qid_2))$  over all  $qid_2$  comparable to  $qid_1$ .  $F(qid_1, qid_2)$  is a function of  $|g_1|$  and  $|g_2|$  for  $(g_1, g_2)$  in  $CG(qid_1, qid_2)$ . We maintain  $|g_2|$ ,  $FA$ , and  $F(qid_1)$  on performing each specialization. For each  $qid_{w_i}$  on  $Link[w_i]$ , find all comparable  $qid_1$  (using  $GTree1$ ) and update  $F(qid_1)$  and  $FA$  as shown in Procedure 2. The update on Line 3 exploits the property that  $F(qid_1)$  is non-decreasing wrt a specialization on  $R_2$  (Theorem 5.1). The computation is proportional to the number of comparable  $(qid_1, qid_{w_i})$  for all new  $qid_{w_i}$  added in the current iteration.

Updating  $BA$ :  $BA$  is the minimum  $(|qid_2| - B(qid_2))$  over all  $qid_2$  in  $R_2$ , where  $B(qid_2)$  is a function of  $|G_1(g_2)|$ ,  $|G_2(g_2)|$  and  $|g_2|$  for the groups  $g_2$  in  $qid_2$ .  $G_1(g_2)$  denotes the set of groups in  $R_1$  comparable to  $g_2$ ,  $G_2(g_2)$  denotes the set of groups in  $R_2$  comparable to any group in  $G_1(g_2)$ . We maintain  $|G_1(g_2)|$ ,  $|G_2(g_2)|$  and  $|g_2|$  on specializing  $qid_2$ .

---

### Procedure 3 Updating $BA$ (Case 1)

---

```

1: for all new  $qid_{w_i}$  on  $Link[w_i]$  do
2:   let  $|G_1(g_{w_i})| = 0$ ; let  $|G_2(g_{w_i})| = 0$ ;
3:   let  $compqid1$  be the set of  $qid_1$  comparable to  $qid_{w_i}$ ;
4:   for all  $qid_1$  in  $compqid1$  do
5:     let  $|G_1(g_{w_i})| = |G_1(g_{w_i})| + |g_1|$ , where  $(g_1, g_{w_i}) \in CG(qid_1, qid_{w_i})$ ;
6:   end for
7:   for all new  $qid'_{w_i}$  on  $Link[w_i]$  do
8:     if  $qid'_{w_i}$  is comparable to some  $qid_1$  in  $compqid1$  then
9:       let  $|G_2(g_{w_i})| = |G_2(g_{w_i})| + |g'_{w_i}|$ , where  $g'_{w_i}$  of  $qid'_{w_i}$  has the same sensitive value as  $g_{w_i}$  of  $qid_{w_i}$ ;
10:    end if
11:  end for
12:  let  $BA = \min(BA, |qid_{w_i}| - B(qid_{w_i}))$ ;
13: end for

```

---

We need to compute  $B(qid_{w_i})$  only for new  $qid_{w_i}$  because  $B(qid_2)$  of unspecialized  $qid_2$  remains intact. To compute  $B(qid_{w_i})$ , first we compute  $|G_1(g_{w_i})|$ ,  $|G_2(g_{w_i})|$  and  $|g_{w_i}|$  for the new  $g_{w_i}$  in  $qid_{w_i}$ . Let  $qid_1$  in  $R_1$  be comparable to  $qid_w$ . Assume  $w$  is from attribute  $A_j$ . There are two cases.

*Case 1:*  $w$  is above the cut  $Cut1_j$ . In this case, only  $qid_{w_i}$  with  $w_i$  being equal to or more general than  $qid_1[A_j]$  is comparable to  $qid_1$ . This means that, to compute  $G_2(g_{w_i})$ , we only need to examine  $g'_{w_i}$  of those  $qid'_{w_i}$  on  $Link[w_i]$ . Based on this observation, Procedure 3 updates  $B(qid_{w_i})$  and  $BA$ . Lines 3-6 computes  $|G_1(g_{w_i})|$  and Lines 8-12 computes  $|G_2(g_{w_i})|$ . Each assignment involving a group applies to all groups in an equivalence class.

*Case 2:*  $w$  is on or below the cut  $Cut1_j$ . In this case, every new  $qid_{w_i}$  remains comparable to  $qid_1$ . Therefore, for  $g_{w_i}$  of  $qid_{w_i}$ ,  $|G_1(g_{w_i})| = |G_1(g_w)|$  and  $|G_2(g_{w_i})| = |G_2(g_w)|$ , where  $g_w$  is the corresponding group of  $qid_w$ . Note that  $|G_1(g_w)|$  and  $|G_2(g_w)|$  were stored with  $g_w$ .

## 6. EMPIRICAL STUDY

We studied the threat of correspondence attacks and the usefulness of the BCF-anonymized  $R_2$ . We employed a publicly available census data set, *Adult*, previously used in [2, 4, 6, 8, 14, 15]. After removing all records with missing values, there were 30,162 and 15,060 records in the training and testing sets on 8 categorical attributes. Refer to [4] for the properties and taxonomy of the attributes. We specified  $D_1$  to contain all records in the testing set, and considered the following three cases of  $D_2$  at timestamp  $T_2$ :

- **200D2:**  $D_2$  contains the first 200 records in the training set, modelling a “small” set of new records at  $T_2$ , typically true if the new data is published in a timely manner.
- **2000D2:**  $D_2$  contains the first 2000 records in the training set, modelling a “medium” set of new records at  $T_2$ .
- **allD2:**  $D_2$  contains all 30,162 records in the training set, modelling a “large” set of new records at  $T_2$ .

We specified two choices of sensitive attributes. In **Sen1**, the sensitive attribute is chosen to be the attribute having the most number of distinct values, which is *Native-country*, and  $QID$  contains the remaining 7 attributes. In **Sen3**, the sensitive attribute is chosen to be the three attributes having the most number of distinct values, which is  $\{Native-country, Education, Occupation\}$ , and  $QID$  contains the remaining 5 attributes. **Sen3** has a more restrictive buddy relationship than **Sen1**.

Two sets of experiments were conducted. In the first set,

we studied the violation of BCF-anonymity when both  $R_1$  and  $R_2$  are  $k$ -anonymized individually as proposed in [5]. In the second set, we anonymized  $R_2$  by the BCF-anonymizer and evaluated its quality. All experiments were conducted on Pentium IV 2.4GHz PC with 512MB of RAM. The BCF-anonymizer took less than 7 seconds, including disk I/O operations. This shows that the algorithm is highly effective.

## 6.1 Detecting Violations

For each pair of  $(D_1, D_2)$  described above, we generalized  $D_1$  and  $D_1 \cup D_2$  into  $k$ -anonymized  $R_1$  and  $k$ -anonymized  $R_2$  separately, by modifying the algorithm in Section 5 to enforce the  $k$ -anonymity on a single table without concerning correspondence attacks. We then measured  $FA$ ,  $CA$  and  $BA$  on the generalized  $R_1$  and  $R_2$ . Since  $FA = CA$  (Theorem 4.5), which was also confirmed by our experiments, we present the results for F-attack and B-attack.

**F-attack:** Figures 1a-1b depict  $FA$  for  $40 \leq k \leq 200$  for Sen1 and Sen3 in 200D2, 2000D2, and *allD2*.  $k$  is the threshold for  $k$ -anonymization. The dash line represents the boundary for  $FA = k$ . A data point below the boundary indicates  $FA < k$ , i.e., a violation of F-anonymity by  $k$ -anonymized  $R_1$  and  $R_2$ .

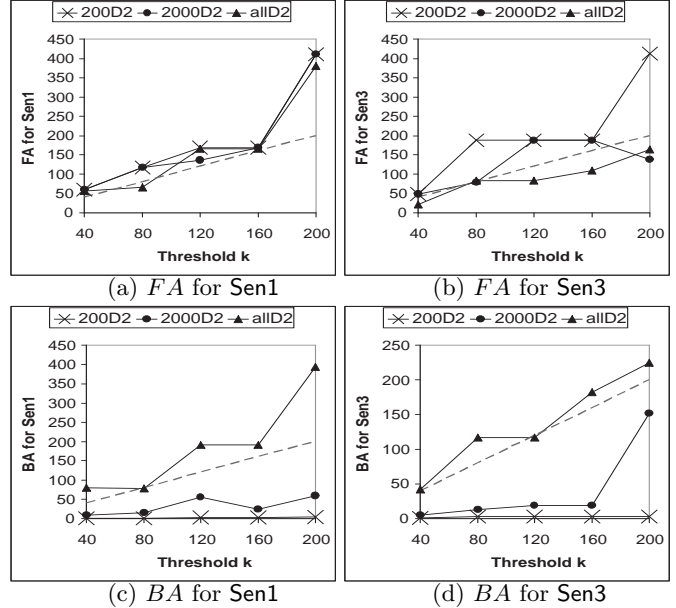
While  $FA$  generally increases as  $k$  increases, there is no guarantee that  $FA \geq k$ . Figure 1b shows that 4 out of the 5 test cases have  $FA < k$  for Sen3 and *allD2*. The most severe case occurs at  $k = 40$  and  $FA = 21$ , where the anonymity of some individuals has been decreased by 48% due to F-attack. Staying above the dash line does not mean no cracked records; rather, it only means that the F-anonymity stays above  $k$ .  $FA$  does not increase monotonically as  $k$  increases. There are two reasons. First, with a larger  $k$ ,  $R_1$  was less specialized, which led to a larger crack size in  $R_1$ . Second, with a larger  $k$ , some specialization on  $R_2$  valid for a smaller  $k$  became invalid and a different sequence of specializations on  $R_2$  was exploited.

A larger size  $|D_2|$  has mixed effects on the crack size  $|g_1| - \min(|g_1|, |g_2|)$ : having more records at  $T_2$  means larger  $|g_2|$ , but also means less generalization in  $R_2$ , thus, smaller  $|g_2|$ . For the more restrictive buddy relationship of Sen3, violations occur more frequently in the case of *allD2* where the decrease of  $|g_2|$  becomes the dominant factor due to less generalization in  $R_2$ . Comparing Figures 1a-1b, Sen3 suffers more violations than Sen1. As the sensitive attribute gets more restrictive for Sen3, both  $|g_1|$  and  $|g_2|$  become smaller. For the larger number of new records in *allD2*, however, the decrease of  $|g_2|$  dominates, therefore causes more violations.

**B-attack:** Figures 1c-1d depict  $BA$  for Sen1 and Sen3. Unlike F-attack, with fewer new records in  $D_2$  (i.e., 200D2 and 2000D2), violations  $BA < k$  occur more frequently and severely since many records in an equivalence class have timestamp  $T_1$ , which are deemed unrelated to the target having timestamp  $T_2$ . The most severe case occurs at  $k = 80$  for Sen1 in 200D2 with  $BA = 1$ . In other words, some target was *uniquely* identified since all other matching records in  $R_2$  were correctly identified as having timestamp  $T_1$ . Another difference from F-attack is that violations  $BA < k$  occur for both Sen1 and Sen3, and are more frequent and more severe for 200D2 and 2000D2 than for *allD2*.

This experiment suggested that a small or medium  $D_2$  is highly vulnerable to B-attack. As the increment  $D_2$  is typically small, the threat of B-attack is very real. A large  $D_2$  can be vulnerable to F-attack due to the reduced general-

Figure 1: Violations of BCF-anonymity



ization in  $R_2$ . This is especially so for a restrictive sensitive attribute. As a result, for neither small nor large  $D_2$  is the  $k$ -anonymity safe from correspondence attacks.

## 6.2 Preventing Violations

This experiment evaluates the quality of BCF-anonymized  $R_2$ . The quality is measured by the discernibility cost [12] for  $R_2$ , which is defined as the square sum of the sizes of all equivalence classes in  $R_2$ , normalized by  $|R_2|^2$ . The normalized cost has the range from 0 to 1, with 0 being the best and 1 being the worst. For each run in the detection experiment, if there is a violation, we compare the discernibility cost of the following three alternatives.

- *BCF-anonymized  $R_2$* :  $R_2$  is generalized by our BCF-anonymizer to satisfy *all* of  $FA \geq k$ ,  $CA \geq k$ ,  $BA \geq k$ .
- *$k$ -anonymized  $R_2$* : This is the  $k$ -anonymized  $R_2$  in the detection experiment and in [5]. This is not safe from correspondence attacks.
- *$k$ -anonymized  $D_2$* : The records in  $D_2$  are anonymized separately from those in  $D_1$ . This alternative is safe because correspondence attacks cannot be applied to data sets representing disjoint sets of individuals.

We evaluate the data quality of a BCF-anonymized  $R_2$  as follows. (1) We measure the penalty to achieve additional protection against all correspondence attacks, relative to the unsafe  $k$ -anonymized  $R_2$ . (2) We measure the benefit to combine  $D_1$  and  $D_2$  into one release  $R_2$  for anonymization, relative to the safe  $k$ -anonymized  $D_2$ . Figure 2 depicts the discernibility cost of the three alternatives. Note that Sen3 has a cost higher than Sen1 since more attributes are exactly preserved through the sensitive attribute and more generalization on the remaining  $QID$  attributes is required to achieve anonymity.

*BCF-anonymized  $R_2$  vs  $k$ -anonymized  $D_2$* . For a smaller  $D_2$ , i.e., 200D2 and 2000D2, BCF-anonymized  $R_2$  has a cost lower than  $k$ -anonymized  $D_2$ . For 200D2, the average cost (over the five test cases) of BCF-anonymized  $R_2$  is 66% and 32% lower than  $k$ -anonymized  $D_2$  for Sen1 and Sen3, shown in Figures 2a-2b. This significant cost reduction ben-

efits from anonymizing a large  $D_1 \cup D_2$ . When  $D_2$  contains a relatively large number of new records, i.e., *allD2* in Figures 2e-2f, the two alternatives have a similar cost since they tend to focus on new records. From this point of view, a separate anonymization of  $D_2$  makes sense if the new data is abundant. Typically, however, the increment  $D_2$  is small.

*BCF-anonymized  $R_2$  vs  $k$ -anonymized  $R_2$ .* BCF-anonymized  $R_2$  generally has a higher cost than  $k$ -anonymized  $R_2$ . The cost increases mainly for 200D2 and 2000D2, as in Figures 2a-2d. A cross-examination of Figures 1c-1d reveals that for 200D2 and 2000D2 there is a very severe violation of  $BA \geq k$  across all  $k$ . The violation was so severe that some target individuals were uniquely identified by the B-attack. To prevent such severe violations, for example, the average increase of cost is 25% in Figure 2d.  $k$ -anonymized  $R_2$  is completely unprotected although it has a lower cost.

In Figure 2c, BCF-anonymized  $R_2$  has a cost lower than  $k$ -anonymized  $R_2$  at  $k = 40$ , though the requirement is more restrictive. In this case, some top ranked specializations used for  $k$ -anonymized  $R_2$  became invalid for BCF-anonymizer due to the additional requirement  $BA \geq k$ , and some lower ranked specializations were selected, which actually resulted in a smaller overall cost. This is possible since the solutions are suboptimal. For a similar reason, a larger  $k$  could possibly produce a lower cost than a smaller  $k$ .

## 7. EXTENSIONS

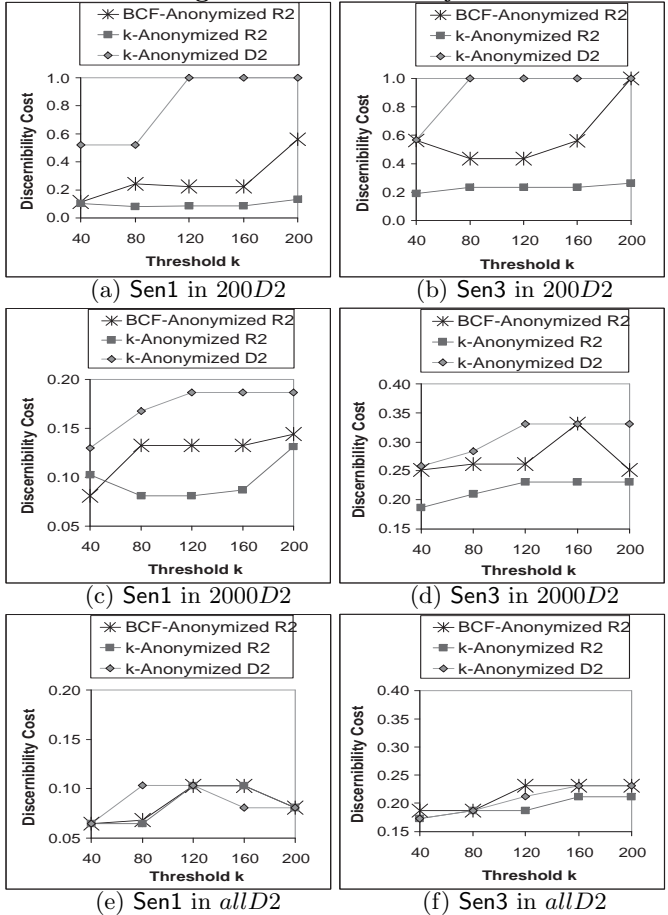
### 7.1 Beyond Two Releases

We extend the two-release case to the general case involving more than two releases. Consider the raw data  $D_1, \dots, D_n$  collected at timestamp  $T_1, \dots, T_n$ . Let  $R_i$  denote the release for  $D_1 \cup \dots \cup D_i$ ,  $1 \leq i \leq n$ . All records in  $R_i$  have the special timestamp, denoted by  $T_i^*$ , that matches *any* timestamp from  $T_1, \dots, T_i$ . The correspondence knowledge now has the form that every record in  $R_i$  (except the last one) has a corresponding record in all releases  $R_j$  such that  $j > i$ . The notion of “generators” can take this into account. Given more releases, the attacker can conduct two additional types of correspondence attacks described below.

**Optimal micro attacks:** The general idea is to choose the “best” background release, yielding the largest possible crack size, *individually* to crack each group. Consider the F-attack on  $R_i$  as an example. The attacker now can choose any  $R_j$  with  $j > i$  as the background release because the correspondence knowledge holds for  $(R_i, R_j)$ . Suppose that the target  $P$  has timestamp  $T_i$  and matches  $(qid_i, \dots, qid_n)$  and that  $(g_i, \dots, g_n)$  are the corresponding groups that have the same sensitive value. Let  $c_{ij}$  be the crack size of a group  $g_i$  computed by Theorem 4.2 wrt  $(g_i, g_j)$  of  $(R_i, R_j)$  instead of  $(g_1, g_2)$  of  $(R_1, R_2)$ . To the attacker, the “optimal” crack size of  $g_i$  is  $\max_j \{c_{ij}\}$  over all  $j > i$ . The number of cracked records in an equivalence class  $qid_i$  wrt  $P$ , i.e.,  $F(P, qid_i, \dots, qid_n)$ , is given by  $\sum_{g_i} c_i$ , where  $g_i$  is a group in  $qid_i$  and  $c_i = \max_j \{c_{ij}\}$  is the optimal crack size of  $g_i$ .

Essentially, the threat of the F-attack on  $g_i$  is determined by the largest crack size  $c_{ij}$  contributed by a collection of “micro attacks” at the group level that each involves (the groups of) two releases at a time, i.e.,  $(g_i, g_j)$ . For such micro attacks, the crack size  $c_{ij}$  can be computed by our method for two releases. Our insight is that it suffices to consider only micro attacks involving two releases. Suppose that a micro attack on  $g_1$  employs the groups  $g_2$  and  $g_3$  from

Figure 2: Discernibility Cost



two background releases  $R_2$  and  $R_3$ . Similar to Theorem 4.2, we can show that the crack size in this case is equal to  $c = |g_1| - \min(|g_1|, |g_2|, |g_3|)$ . But  $c$  is exactly the above optimal crack size of  $g_1$  because  $c = \max(c_{12}, c_{13})$ , where  $c_{12} = |g_1| - \min(|g_1|, |g_2|)$  is the crack size produced by the micro attack using  $(g_1, g_2)$  and  $c_{13} = |g_1| - \min(|g_1|, |g_3|)$  is the crack size produced by the micro attack using  $(g_1, g_3)$ .

The micro attacks for the three types of attacks can be represented as follows. Note that the crack size for these micro attacks can be directly computed by the methods in Section 4. Let  $g_i$  be a group in  $R_i$  and  $g_j$  be a group in  $R_j$ .

- F-attack( $g_i, g_j$ ),  $j > i$ : The target  $P$  has timestamp  $T_i$  and the attacker tries to crack the records in  $g_i$  using  $g_j$  of the background release  $R_j$ . In this case, a cracked record does not originate from  $P$ 's QID.
- C-attack( $g_i, g_j$ ),  $j > i$ : The target  $P$  has timestamp  $T_i$  and the attacker tries to crack the records in  $g_j$  using  $g_i$  of the background release  $R_i$ . A cracked record either does not have timestamp  $T_i^*$  or does not originate from  $P$ 's QID. Note that if a record in  $g_j$  does not have timestamp  $T_i^*$ , it has a timestamp from  $T_{i+1}, \dots, T_j$ , therefore, does not match  $P$ 's timestamp.
- B-attack( $g_i, g_j$ ),  $j > i$ : The target  $P$  has timestamp  $T_j$  and the attacker tries to crack the records in  $g_j$  using  $g_i$  of the background release  $R_i$ . Note that a cracked record has timestamp  $T_i^*$  (because of coming from  $g_i$ ), thus, does not match  $P$ 's timestamp.

**Composition of micro attacks:** Another type of attack is to “compose” multiple micro attacks together (apply *one*

after another) in order to increase the crack size of a group. Composition is possible *only if* all the micro attacks in the composition assume the same timestamp for the target and the correspondence knowledge required for the next attack holds after applying previous attacks. There are two and only two possible cases of composition.

*Case 1:* Apply B-attack( $g_i, g_j$ ) followed by F-attack( $g_j, g_l$ ),  $l > j > i$ . Both attacks assume the target  $P$  at timestamp  $T_j$  and crack records in  $g_j$  wrt  $P$ . First, apply B-attack( $g_i, g_j$ ) to crack  $g_j$ . This excludes some records at timestamp  $T_i^*$ . The crack size,  $x_B$ , is computed by Theorem 4.4. Let  $g'_j$  and  $R'_j$  denote the reduced  $g_j$  and  $R_j$  without cracked records. Note that the following correspondence knowledge holds: every record in  $R'_j$  has a buddy in  $R_l$  since  $R'_j$  is a subset of  $R_j$ . Hence, F-attack( $g'_j, g_l$ ) can be applied to crack  $g'_j$ , which excludes some records from  $g'_j$  that do not originate from  $P$ 's *QID*. The crack size,  $x_F$ , is given by Theorem 4.2. Overall, the crack size of  $g_j$  is  $x_B + x_F$ . Note that if F-attack( $g_j, g_l$ ) is applied first to crack  $g_j$ , B-attack( $g_i, g'_j$ ) cannot be applied since a record in  $R_i$  may not have a buddy in  $R'_j$ .

*Case 2:* Apply B-attack( $g_i, g_j$ ) followed by C-attack( $g_j, g_l$ ),  $l > j > i$ . Both attacks assume a target  $P$  at timestamp  $T_j$ . First, apply B-attack( $g_i, g_j$ ) to crack  $g_j$ , which excludes some records that do not have timestamp  $T_j$  (thus do not represent  $P$ ). Let  $g'_j$  and  $R'_j$  be the reduced  $g_j$  and  $R_j$ . Since every record in  $R'_j$  has a buddy in  $R_l$ , C-attack( $g'_j, g_l$ ) can be applied to crack  $g_l$ . From Theorem 4.3, the crack size of  $g_l$  by this C-attack is  $|g_l| - \min(|g'_j|, |g_l|)$ , which is larger than  $|g_l| - \min(|g_j|, |g_l|)$  by applying C-attack( $g_j, g_l$ ) alone.

All other cases, however, cannot be composed. For example, F-attack( $g_i, g_j$ ) and C-attack( $g_j, g_l$ ) cannot be applied in sequence because the former assumes a target  $P$  at timestamp  $T_i$  whereas the latter assumes  $P$  at timestamp  $T_j$ .

We extend the anonymization algorithm as follows. (1) The notion of BCF-anonymity should be defined based on the optimal crack size of a group wrt micro attacks as well as composed attacks on the group. (2) Each time we anonymize the next release  $R_n$  for  $D_1 \cup \dots \cup D_n$ , we assume that  $R_1, \dots, R_{n-1}$  satisfy BCF-anonymity. Hence, the anonymization of  $R_n$  only needs to ensure that BCF-anonymity is not violated by any attack that *involves*  $R_n$ . (3) The anti-monotonicity of BCF-anonymity, in the spirit of Theorem 5.1, remains valid in this general case. These observations are crucial for maintaining the efficiency.

As the number of releases increases, the constraints imposed on the next release  $R_n$  become increasingly restrictive. However, this does not necessarily require more distortion because the new records  $D_n$  may help reduce the need of distortion. In case that the distortion becomes too severe, the data holder may consider starting a new chain of releases without including previously published records.

## 7.2 Beyond Anonymity

The proposed approach can be extended to incorporate with the requirement of entropy  $\ell$ -diversity and  $(c, \ell)$ -diversity in [8], confidence bounding [15], and  $(\alpha, k)$ -anonymity [16]. First, we modify these measures to take into account the exclusion of cracked records. In this case, the crack size of each group gives all the information needed to exclude sensitive values from an equivalence class. To extend the anonymization algorithm, we argue that anonymity is a *necessary* privacy property because identifying the exact record of an individual from a small set of records is too easy. Thus, BCF-

anonymity is required even if other privacy requirements are desired. Under this assumption, we can still apply the proposed approach to prune unpromising specializations based on the anti-monotonicity of BCF-anonymity.

## 8. CONCLUSION

We considered the anonymity problem for a scenario where the data are continuously collected and published. Each release contains the new data as well as previously collected data. Even if each release is  $k$ -anonymized, the anonymity of an individual can be compromised by cross-examining multiple releases. We formalized this notion of attacks and presented a detection method and an anonymization algorithm to prevent such attacks. Finally, we showed that both the detection and the anonymization methods are extendable to deal with multiple releases and other privacy requirements.

## 9. REFERENCES

- [1] L. Burnett, K. Barlow-Stewart, A. Pros, and H. Aizenberg. The gene trustee: A universal identification system that ensures privacy and confidentiality for human genetic databases. *Journal of Law and Medicine*, 10:506–513, 2003.
- [2] J.-W. Byun, Y. Sohn, E. Bertino, and N. Li. Secure anonymization for incremental datasets. In *VLDB Workshop on Secure Data Management (SDM)*, 2006.
- [3] B. C. M. Fung, K. Wang, and P. S. Yu. Top-down specialization for information and privacy preservation. In *ICDE*, pages 205–216, April 2005.
- [4] B. C. M. Fung, K. Wang, and P. S. Yu. Anonymizing classification data for privacy preservation. *TKDE*, 19(5):711–725, May 2007.
- [5] T. Iwuchukwu, D. J. DeWitt, A. Doan, and J. F. Naughton.  $k$ -anonymization as spatial indexing: Toward scalable and incremental anonymization. In *ICDE*, 2007.
- [6] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Incognito: Efficient full-domain  $k$ -anonymity. In *SIGMOD*, pages 49–60, June 2005.
- [7] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Mondrian multidimensional  $k$ -anonymity. In *ICDE*, 2006.
- [8] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkatasubramanian.  $\ell$ -diversity: Privacy beyond  $k$ -anonymity. In *ICDE*, Atlanta, GA, April 2006.
- [9] A. Meyerson and R. Williams. On the complexity of optimal  $k$ -anonymity. In *PODS*, pages 223–228, 2004.
- [10] J. Pei, J. Xu, Z. Wang, W. Wang, and K. Wang. Maintaining  $k$ -anonymity against incremental updates. In *SSDBM*, Banff, Canada, 2007.
- [11] P. Samarati and L. Sweeney. Protecting privacy when disclosing information:  $k$ -anonymity and its enforcement through generalization and suppression. Technical report, SRI International, March 1998.
- [12] A. Skowron and C. Rauszer. *Intelligent Decision Support: Handbook of Applications and Advances of the Rough Set Theory*, chapter The discernibility matrices and functions in information systems. 1992.
- [13] L. Sweeney.  $k$ -anonymity: A model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5), 2002.
- [14] K. Wang and B. C. M. Fung. Anonymizing sequential releases. In *SIGKDD*, pages 414–423, August 2006.
- [15] K. Wang, B. C. M. Fung, and P. S. Yu. Template-based privacy preservation in classification problems. In *ICDM*, pages 466–473, November 2005.
- [16] R. C. W. Wong, J. Li, A. W. C. Fu, and K. Wang.  $(\alpha, k)$ -anonymity: An enhanced  $k$ -anonymity model for privacy preserving data publishing. In *SIGKDD*, August 2006.
- [17] X. Xiao and Y. Tao.  $m$ -invariance: Towards privacy preserving re-publication of dynamic datasets. In *SIGMOD*, June 2007.