Information Sciences 179 (2009) 2948–2963

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

k-Anonymous data collection

Sheng Zhong [a,*], Zhiqiang Yang [b], Tingting Chen [a]

[a] Department of Computer Science and Engineering, State University of New York, Buffalo, Amherst, NY 14260, USA
[b] Imagine Software, Inc., 233 Broadway, 17th floor, Newyork, NY 10279, USA

ARTICLE INFO

Article history:
Received 12 October 2007
Received in revised form 6 April 2009
Accepted 2 May 2009

Keywords:
Privacy
k-Anonymity
Data collection

ABSTRACT

To protect individual privacy in data mining, when a miner collects data from respondents, the respondents should remain anonymous. The existing technique of Anonymity-Preserving Data Collection partially solves this problem, but it assumes that the data do not contain any identifying information about the corresponding respondents. On the other hand, the existing technique of Privacy-Enhancing *k*-Anonymization can make the collected data anonymous by eliminating the identifying information. However, it assumes that each respondent submits her data through an unidentified communication channel. In this paper, we propose *k*-Anonymous Data Collection, which has the advantages of both Anonymity-Preserving Data Collection and Privacy-Enhancing *k*-Anonymization but does not rely on their assumptions described above. We give rigorous proofs for the correctness and privacy of our protocol, and experimental results for its efficiency. Furthermore, we extend our solution to the fully malicious model, in which a dishonest participant can deviate from the protocol and behave arbitrarily.

1. Introduction

In this information age, huge amounts of data are collected and analyzed every day. Given the unprecedented convenience of collecting and analyzing data, how to protect individual privacy has become a very challenging problem. Consequently, the public have expressed deep concern about data privacy [26,42,43].

Data mining is a powerful tool for finding patterns and knowledge from large amounts of data [66–68]. Naturally, privacy protection in data mining receives a lot of attention [3,10]. Since a considerable amount of data used in data mining is collected over computer networks, it would be very beneficial if we could provide privacy protection at the stage of data collection. In this paper, we consider a typical scenario of online data collection: the miner queries large sets of respondents, each of whom responds with a piece of data. Our question is whether this procedure can be carried out without violating any respondent's privacy.

Specifically, we hope to carry out the data collection procedure in an *anonymous* manner. That is, the miner should be able to collect data from the respondents, but should not be able to link any respondent's data to the respondent. Therefore, each respondent is "hidden" among a number of peers for privacy protection.

Imagine, for example, that a medical researcher is collecting data from respondents. The data he intends to collect may include some privacy information, like the medical histories of the respondents. In order to guarantee that the respondents' privacy is not violated, we would like to have the medical researcher collect the data in such a way that he (or anybody else) cannot link any respondent to any collected private information. Hence, the miner may learn from the collected data that "some respondent" has had, e.g., a stroke, but he should have no idea *which* respondent has had it.

* Corresponding author. Tel.: +1 716 645 3180x107; fax: +1 716 645 3464.
E-mail address: szhong@cse.buffalo.edu (S. Zhong).

0020-0255/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2009.05.004

A partial solution to the above problem is Anonymity-Preserving Data Collection (APDC) [7,65]. However, APDC relies on an assumption that the submitted data does not contain any identifying information. Nevertheless, this may not always be the case in practice. Although identifiers of individuals (like social security number) can be easily excluded from the data collection for privacy protection, it is often necessary to include other types of identifying information in the data collection.

In the example we have mentioned above, in the data collected by the medical researcher, there may be attributes of respondents like gender, age, blood type, zip code, phone number, etc. Clearly, these attributes can be useful in identifying an individual. For example, the data could show that a man aged 23 with zip code 11,000 has had a stroke. Although we do not have the name of this man, there may be just one man at this age having this zip code among all respondents. Hence, the privacy of this respondent has been violated.

Technically, the set of such attributes is called the *quasi-identifier* of an individual. The procedure of processing quasi-identifiers to make the data anonymous is called *k-anonymization*; it has been studied extensively [5,12,18,40,47,50–52,58]. The main idea of *k*-anonymization is to suppress or generalize the values of quasi-identifier such that each value either appears for at least *k* times, or does not appear at all. In this way, each involved individual is hidden among at least *k* peers. In particular, Privacy-Enhancing *k*-Anonymization (PEkA) [18] is a distributed protocol that allows a miner to *k*-anonymize data from a number of respondents. However, PEkA assumes that each respondent uses an unidentified communication channel to submit data. In reality, we may not always have such unidentified channels available to all respondents.

In this paper, we study *k*-Anonymous Data Collection (kADC), which extends APDC to work with data that contains quasi-identifiers. Unlike PEkA, it does not rely on unidentified communication channels. Thus it has the advantages of both APDC and PEkA but does not depend on their assumptions described above. Our target is to build a protocol that collects data but keeps all respondents *k*-anonymous, even if the data collected include quasi-identifiers.

Specifically, in the example we have considered above, we would like to see the medical researcher collect his data, which may include attributes like gender, age, zip code etc. However, the medical researcher should still be unable to link any respondent to any private information. This is achieved by having each respondent hidden among at least *k* peers. The medical researcher may know that a man aged 23 with zip code 11,000 has had a stroke, but there are at least *k* men at this age having this zip code among all respondents. Hence, the privacy of respondents is protected.

1.1. Related work

Privacy-preserving data mining has been studied extensively. There are a number of methods to perturb the respondents' data for privacy protection [4,3,17,45,16,14,34,71,35]. The main advantage of this approach is its good efficiency, but it also has the disadvantage: it generally has a tradeoff between the privacy of respondents and the accuracy of the data mining result. If we want to guarantee more privacy for each respondent, then we have to add more noise to the data, which results in less accuracy of the data mining result. In [11], Canfora et al. reported an empirical evaluation of this tradeoff. The privacy problems of certain perturbation methods were further explored in [34,28].

Another class of techniques for privacy-preserving data mining are based on cryptography [38,32,54,61,2,63,64,70,49]. Unlike the perturbation-based solutions discussed above, some of the cryptographic solutions (e.g., [64]) can achieve full privacy plus full accuracy. Nevertheless, these cryptographic protocol are specifically designed for certain mining tasks. The only known general-purpose cryptographic protocols are from secure multiparty computation (see [22] for a good summary of the results), but these general-purpose solutions are prohibitively expensive and thus cannot be used in practice.

Recently, APDC [65] was proposed as an alternative to privacy-preserving data mining; it allows a miner to collect data anonymously and thus provides protection of respondents' privacy regardless of the mining task. This work has had some impact on the research literature. For example, in [59], Williams and Barker proposed a hierarchy-based approach for privacy-preserving data collection in which data providers can divulge information at any chosen privacy level; in [72], Zhang and Zhang presented a method to allow the respondents to use generalization to anonymously collect data in a client-service-to-user model. However, as we have mentioned, APDC relies on the assumption that the data do not contain any identifying information about the respondents. So, one motivation of our work is to eliminate this assumption of APDC. Technically, APDC is essentially an adaptation of anonymous communication techniques, in particular mix network techniques, to the scenario of online data collection. Here, mix network is a topic that has been extensively studied in cryptography [8,41,46,30,25]. Other ways to provide anonymous communication include dining cryptographer networks [9,57,24] and *k*-anonymous message transmission [56]. (Systems like Crowds [44] and Hordes [37] also provide a form of anonymous communication. But these systems have strong restrictions on the adversary, e.g., the adversary can only eavesdrop communication within one hop.)

Complementary to APDC, research on *k*-anonymization (e.g., [50,47,12,52,51,40,5,58,1,18,39,60,62,36,27], among others) studies how to handle the data containing identifying information. In particular, the PEkA protocol [69] *k*-anonymizes the data in the data collection procedure. However, as we have also mentioned, this protocol depends on the assumption that each respondent has an unidentified communication channel. Consequently, it is another motivation of our work to eliminate this assumption.

There are other types of work on privacy in data mining, for example how to measure privacy guarantee in data mining. Various definitions were given based on confidence intervals [4], based on mutual information [3], based on priori and posterior knowledge [16,13], and based on cryptographic notions of confidentiality [20,15].

1.2. Our contributions

We briefly summarize our contributions as follows:

- We propose the approach of *k-Anonymous Data Collection* (kADC). This approach allows the miner to collect a *k*-anonymized version of the respondents' data in such a way that the miner cannot figure out which respondent submits which piece of sensitive data. Compared with APDC and PEkA, kADC does not rely on the assumption of no identifying information in the submitted data or the availability of unidentifiable communication channels.
- We design a basic protocol of kADC and give rigorous proofs for its correctness and anonymity properties in the semi-honest model, i.e., the model that every party follows the protocol but may attempt to find out the private information which they are supposed not to learn. We discuss how to improve the efficiency of our protocol and provide experimental data to measure the efficiency.
- We extend our basic protocol to the fully malicious model, in which a malicious party can deviate from the protocol and behave arbitrarily. We use digital signatures and zero-knowledge proofs to prevent and detect malicious behaviors.
- Since our basic protocol suppresses a good amount of information in quasi-identifiers, we further develop an improved protocol that *significantly reduces the amount of suppressed information*. This improved protocol is very useful for applications that can not afford much information loss in quasi-identifiers.

The target of this paper is to *demonstrate the possibility* of collecting data with good privacy protection and good efficiency, and without unnecessary assumptions. We emphasize that even our improved protocol still suppresses more information than the theoretically minimum amount of information suppressed by a *k*-anonymization algorithm. Combining our research with that of minimizing suppressed information in *k*-anonymization is an interesting topic; but that is out of the scope of our paper and thus we leave it to future research.

1.3. Paper organization

The rest of this paper is organized as follows. In Section 2, we give a formal description of our problem and a formal definition of anonymity. In Section 3, we present a basic solution in the semi-honest model. This section also includes proofs of the correctness and anonymity properties, discussion on improving the efficiency, and experimental results of the efficiency. In Section 4, we extend our protocol to protect respondents' privacy against malicious behavior. In Section 5, we present our improved protocol, which suppresses significantly less information than the basic protocol.

2. Technical preliminaries

The kADC problem can be modeled as follows: there are N respondents $1,\ldots,N$; each respondent i has a piece of data d_i. Each d_i consists of two parts: $d_i^+ = (d_i^1, d_i^2, \ldots, d_i^m)$, the quasi-identifier, which contains identifying information about respondent i; d_i^-, other attributes, which does not contain any identifying information. Without loss of generality, we assume that all d_i^+s and d_i^-s are of the same length. The objective of our protocol is that the miner should learn $((d^\star_{\pi(1)}, d^-_{\pi(1)}), \ldots, (d^\star_{\pi(N)}, d^-_{\pi(N)}))$, where π is a random permutation of $(1,\ldots,N)$ and $(d_1^\star,\ldots,d_N^\star)$ is a *k-anonymized version* of $(d_1^+,\ldots,d_N^+)$. By saying "*k*-anonymized version", we mean each $d_i^\star = (d_i^{\star,1}, d_i^{\star,2}, \ldots, d_i^{\star,m})$ satisfies the following two requirements:

- Each $d_i^{\star,j}$ is equal to either d_i^j or a special symbol $\star$ (which denotes that d_i^+ has been suppressed for privacy protection).[1]
- If a value appears in $(d_1^\star,\ldots,d_N^\star)$, it must appear for at least k times.

Any kADC protocol satisfying the above requirements is said to be *correct*. Furthermore, for privacy protection, we require that the miner should not be able to find out which user submitted $(d^\star_{\pi(i)}, d^-_{\pi(i)})$. Formally, we have the following definition.

Definition 2.1. A kADC protocol is private against the miner in the semi-honest model if the protocol is correct and for all $(d_1,\ldots,d_N)$, for all $i \in \{1,\ldots,N\}$, there exists $I \subseteq \{1,\ldots,N\}$ such that $i \in I |I| \geqslant k$, and that for all permutation σ on I,

$$\{\mathsf{view}_{\mathsf{miner}}((d_1^+,d_1^-),\ldots,(d_N^+,d_N^-))\} \underset{\equiv}{\mathsf{c}} \{\mathsf{view}_{\mathsf{miner}}((d_1^+,d^-_{\sigma(1)}),\ldots,(d_N^+,d^-_{\sigma(N)}))\},$$

where $\mathsf{view}_{\mathsf{miner}}((d_1^+,d_1^-),\ldots,(d_N^+,d_N^-))$ denotes the view of the miner when the original data is $(d_1^+,d_1^-),\ldots,(d_N^+,d_N^-)$; $\underset{\equiv}{\mathsf{c}}$ denotes computational indistinguishability of probability ensembles; the computational complexities are measured in terms of a security parameter κ, which is the length of cryptographic key. Note that we are sloppy in the above equation, because σ is a permutation on I and we apply it to indices out of I; for all $j \notin I$, we have $\sigma(j) = j$.

[1] In this paper, we only consider the *k*-anonymization methods of suppression. There are other ways to *k*-anonymize data but they are out of scope of this paper.

To design a practical protocol, we introduce an additional participant and make an assumption of *no collusion*: we assume that this new participant of data collection protocol does not collude with the miner. Note that this kind of assumptions (of no collusion between certain participants) have been widely used in cryptography (e.g., in [31,48]) and in privacy-preserving data mining (e.g., in [33,55]). The reason is that the involved participants belong to different organizations and have different interests; thus, it is natural to assume they are unwilling to collude with each other to violate anybody's privacy. (However, since the involved participants all agree to participate in the protocol, they still *cooperate* with each other in the protocol. In other words, no collusion between some participants does not mean no cooperation between them.)

Specifically, the additional participant we introduce in our problem is called the Data Collection Helper (DCH). We assume that DCH is an independent participant of the data collection protocol and that *it does not collude with the miner*. In reality, a DCH can be the representative of an external auditor or an external server who is paid to provide computing services. We do not trust the DCH, just like we do not trust the miner. All we need is that the DCH does not collude with the miner. Naturally, we also require that the DCH should not be able to find which user submitted which piece of data.

Definition 2.2. A kADC protocol is private against the DCH in the semi-honest model if the protocol is correct and for all $(d_1,\ldots,d_N)$, for all $i \in \{1,\ldots,N\}$ there exists $I \subseteq \{1,\ldots,N\}$ such that $i \in I|I| \geqslant k$, and that for all permutation σ on I,

$$\{\text{view}_{DCH}((d_1^+,d_1^-),\ldots,(d_N^+,d_N^-))\} \underset{\equiv}{c} \{\text{view}_{DCH}((d_1^+,d_{\sigma(1)}^-),\ldots,(d_N^+,d_{\sigma(N)}^-))\}.$$

Note that the above definitions only apply to the semi-honest model. That is, they are sufficient for the purpose of privacy protection if all parties follow the protocol. In contrast, in the fully malicious model, some protocol participant can deviate from the protocol, and thus we need to have additional protection. We discuss privacy in the fully malicious model in Section 4.

Also note that the above definitions do not consider the possibility that some users might be dishonest. Nevertheless, we can actually easily extend our definitions to cover this possibility; the only reason for not doing this is that such an extension needs complicated notations and we want to keep our presentation simple and clear, focusing on technical ideas rather than mathematical notations. Furthermore, our solutions can also be proved to satisfy the extended definitions.

3. Basic solution

In this section, we present a basic protocol for the kADC problem in the semi-honest model. Our solution in the malicious model (in Section 4), and our improved solution (in Section 5) that suppresses less information in quasi-identifiers, are both extended from this protocol.

3.1. Building blocks

Recall that the miner should obtain $((d_{\pi(1)}^{\star},d_{\pi(1)}^-),\ldots,(d_{\pi(N)}^{\star}d_{\pi(N)}^-))$ without knowing π. To design our protocol, we use *ElGamal encryption*, which allows *rerandomization* operations. Below we briefly describe the ElGamal encryption scheme and rerandomization operations.

Suppose that G is a cyclic group (in which the discrete logarithm is believed to be hard) and that $|G| = q$, where q is a large prime. Let g be a generator of G. The ElGamal encryption scheme is a public key cryptosystem. If x is a private key, then the corresponding public key is $y = g^x \text{mod} q$. If we want to encrypt a message d using the public key y, we can compute

$$\bar{d} = (dy^r, g^r),$$

where r is chosen uniformly at random from $[0, q-1]$. To make the presentation clear, hereafter we write the above encryption operation as $\bar{d} = E(d,r)$. If we want to decrypt the ciphertext $\bar{d}$ using the private key x, we should compute

$$d = \bar{d}[1]/\bar{d}[2]^x,$$

where $\bar{d}[1]$ and $\bar{d}[2]$ denote the first and the second components of $\bar{d}$, respectively. It has been shown in [53] that (under a standard cryptographic assumption) the ElGamal encryption scheme is *semantically secure* (see [23] for the definition of semantic security).

Just like other probabilistic cryptosystems, the ElGamal encryption scheme maps each cleartext to many ciphertexts, since the random number r can take many different values. ElGamal supports a rerandomization operation, which means computing a different encryption of d from a given encryption of d (without knowing the private key). An advantage of the rerandomization operation is that, if we rerandomize and permute a sequence of ciphertexts, then we get another sequence of ciphertexts corresponding to the same multiset of cleartexts but in a different order. Given these two sequences of ciphertexts, the adversary cannot gain any knowledge about which new ciphertext corresponds to which old ciphertext.

An interesting property of the ElGamal encryption scheme is that it is *homomorphic*. Recall that each ElGamal ciphertext has two components. If we multiply the corresponding components of two ElGmal ciphertexts, we get a new ciphertext whose cleartext is the product of the two old ciphertexts. For simplicity, hereafter we often say multiplying an ElGamal ciphertext by another ElGamal ciphertext; what we actually mean is to multiply the corresponding components. Similarly, if we say dividing a ciphertext by another ciphertext, we actually mean dividing each component of the first ciphertext by the corresponding component of the second ciphertext.

3.2. Protocol

Our protocol for k-Anonymous data collection is shown in Protocol 1. It consists of four phases: data submission, the miner's randomization, the DCH's randomization and decryption. Each phase is explained in more details below. For ease of understanding, we also include an example in our explanation.

Algorithm 1. k-Anonymous Data Collection Protocol – The Basic Solution

Let the miner's private key be x and his public key be $y = g^x$.

Let the DCH's private key be u and his public key be $v = g^u$.

1. Phase 1: Data submission.
2. **for** each respondent i **do**
3. i picks r_i^+, r_i^- uniformly and independently.
4. i encrypts her data using public key yv: $d_i^+ = E_{yv}(d_i^+, r_i^+); d_i^- = E_{yv}(d_i^-, r_i^-)$.
5. i submits $\overline{d_i^+}\,\overline{d_i^-}$ to the miner.
6. **end for**
7. Phase 2: Miner's Randomization Operations.
8. **for** each pair (i,j) **do**
9. The miner computes $\overline{q_{ij}} = (\overline{d_i^+}/\overline{d_j^+})^{r_{ij}}$, where each r_{ij} is chosen uniformly and independently.
10. **end for**
11. **for** each i **do**
12. The miner chooses a permutation θ_i on $\{1,\ldots,N\}$ uniformly at random.
13. **for** each $j \in \{1,2,\cdots,N\}$ **do**
14. The miner computes $\overline{q'_{ij}} = \overline{q_{i,\theta_i(j)}}$, $\overline{q''_{ij}}[1] = \overline{q'_{ij}}[1]/(\overline{q'_{ij}}[2])^x$ and sets $\overline{q''_{ij}}[2] = \overline{q'_{ij}}[2]$.
15. **end for**
16. **end for**
17. The miner sends the DCH: $\{\overline{d_i^+}, \overline{d_i^-}\}_{i=1,\ldots,N}$, $\{\overline{q''_{ij}}\}_{i=1,\ldots,N; j=1,\ldots,N}$.
18. Phase 3: DCH's Randomization Operations.
19. The DCH computes $q'_{ij} = \overline{q''_{ij}}[1]/(\overline{q''_{ij}}[2])^u$, for each pair (i,j).
20. **for** each i **do**
21. The DCH counts $c_i = |\{j : q'_{ij} = 1\}|$.
22. **if** $c_i < k-1$ **then**
23. The DCH sets $\overline{d'_i}$ to an encryption of $(\star,\star,\ldots,\star)$ under public key yv.
24. **else**
25. the DCH sets $\overline{d'_i} = \overline{d_i^+}$.
26. **end if**
27. **end for**
28. **if** $1 \leqslant |\{i : c_i < k-1\}| < k$ **then**
29. The DCH lets C be the smallest c_i that is greater than $k-1$.
30. For all i s.t. $c_i = C$, the DCH sets $\overline{d''_i}$ to an encryption of $(\star,\star,\ldots,\star)$ under public key yv; for all other i, the DCH sets $\overline{d''_i} = \overline{d'_i}$.
31. **end if**
32. **if** $|\{i : c_i < k-1\}| \geqslant k$ or $|\{i : c_i < k-1\}| = 0$ **then**
33. The DCH defines $\overline{d''_i} = \overline{d'_i}$.
34. **end if**
35. **for** each i **do**
36. The DCH computes $\overline{d'''_i}[1] = \overline{d''_i}[1]/(\overline{d''_i}[2])^u$ and $\overline{d_i^\sharp}[1] = \overline{d_i^-}[1]/(\overline{d_i^-}[2])^u$.
37. The DCH defines $\overline{d'''_i}[2] = \overline{d''_i}[2]$ and $\overline{d_i^\sharp}[2] = \overline{d_i^-}[2]$.
38. The DCH chooses a permuation π on $\{1,\ldots,N\}$ uniformly at random and computes $\overline{d''''_i} = \overline{d'_{\pi(i)}}$ and $\overline{d_i}^\diamondsuit = \overline{d_{\pi(i)}}^\sharp$.
39. **end for**
40. The DCH sends $\overline{d''''_i}$ and $\overline{d_i}^\diamondsuit$ to the miner for all i.
41. Phase 4: Decryption.
42. **for** each i **do**
43. The miner decrypts $\overline{d''''_i}$ and $\overline{d_i}^\diamondsuit$ using his own private key x, where the decryption of $\overline{d''''_i}$ is the part of data containing identifying information; the decryption of $\overline{d_i}^\diamondsuit$ is the part of data without identifying information.
44. **end for**
45. If the data needs to be published, the miner must publish it in a randomized order.

- Data submission

  In the data submission phase, each respondent computes the product of the miner's public key and the DCH's public key, and encrypts her own data using the product. (Note the data needs to be properly encoded before encryption, because

cryptosystems like ElGamal only deal with numbers.) Then she sends the encryptions to the miner.For example, suppose there are three respondents 1, 2, and 3, having data $(M, 23, stroke)(F, 24, flu)(M, 23, allergy)$, respectively, where the first two fields of each piece of data are the quasi-identifier. After proper encoding, respondent 1's data becomes (123, 55), respondent 2's data becomes (224, 38), and respondent 3's data becomes (123, 49), where the quasi-identifiers are 123, 224, and 123, respectively. Then the miner receives an encryption of 123 and an encryption of 55 from respondent 1; the public key used to encrypt them is the product of the miner's public key and the DCH's public key. Similarly, the miner receives an encryption of 224 and an encryption of 38 from respondent 2, and an encryption of 123 and an encryption of 49 from respondent 3.

- Miner's randomization operations
  In this phase, the miner first computes the quotient of each pair of quasi-identifier encryptions that he has received from the respondents; the result is an encryption of the quotient of the corresponding pair. Then, the miner raises each encrypted quotient to a random power – note that this generates an encryption of 1 if the quotient is 1, and a random encryption otherwise. Next, the miner permutes the encrypted powers of quotients related to each respondent, and decrypts all of them using his own private key. (The miner's decryption operations do not generate the cleartexts; they only generate partially decrypted ciphertexts. The DCH can decrypt these partially decrypted ciphertexts to get the cleartexts.) At the end of this phase, the miner sends the DCH the partially decrypted powers as well as all the encrypted data he has received from respondents.
  In the example we have considered, this phase should be as follows. The miner computes, for respondent 1, encryptions of 123/123, 123/224 and 123/123, respectively; he computes, for respondent 2, encryptions of 224/123, 224/224, and 224/123, respectively; he computes, for respondent 3, encryptions of 123/123, 123/224 and 123/123, respectively. Then the miner computes a random power of the each of the above encryptions. Hence, for respondent 1, the miner has got an encryption of 1, a random encryption, and encryption of 1; for respondent 2, the miner has got a random encryption, an encryption of 1, and a random encryption; for respondent 3, the miner has got an encryption of 1, a random encryption and an encryption of 1. He permutes them for each respondent and partially decrypts them and sends them to the DCH, together with the data he has received. So the DCH receives the follows from the miner: for respondent 1, two partially decrypted encryptions of 1, a partially decrypted random encryption, and encryptions of 123 and 55; for respondent 2, a partially decrypted random encryption of 1, two partially decrypted random encryptions, and encryptions of 224 and 38; for respondent 3, a partially decrypted random encryption and two partially decrypted encryptions of 1, and encryptions of 123 and 49.
- DCH's radomization operations
  First the DCH decrypts the powers using his own private key. If the result is 1, it means that the corresponding pair of quasi-identifiers are equal. So, for each respondent, the DCH counts the 1 s that he obtains. If it is less than $k-1$, i.e., there are less than $k-1$ other respondents having the same quasi-identifiers, then the DCH suppresses the quasi-identifier. After this round of suppression, if there are fewer than K encrypted $(\star, \star, \ldots, \star)$, then the DCH also suppresses the least frequently appeared quasi-identifiers that has not been suppressed, so that there are at least K encrypted $(\star, \star, \ldots, \star)$. After all rounds of suppression, the DCH decrypts the unsuppressed quasi-identifiers as well as other attributes, using his own private key. Note that this does not generate cleartexts; it only generates partially decrypted ciphertexts, which can be decrypted to cleartexts by the miner. The DCH permutes the decryptions and sends the results to the miner.
  In the example we have considered, this phase should be as follows. The DCH further decrypts the partially decrypted encryptions, and get the cleartexts of the powers. So, for respondent 1, the DCH has got two 1s and a random number; for respondent 2, the DCH has got a 1 and two random numbers; for respondent 3, the DCH has got a random number and two 1 s. Note that, with high probability, these random numbers are not equal to 1. So the DCH counts the number of 1 s for each respondent and get 2 for respondent 1, 1 for respondent 2, and 2 for respondent 3. He uses these numbers to do suppression, in order to k-anonymize the data. Suppose, for simplicity, that $k = 1$, so that no supression is really needed. Next, the DCH decrypts all the data using his own private key, so that these data become partially decrypted. The miner receives the follows from the DCH: partially decrypted encryptions of 123 and 55, partially decrypted encryptions of 224 and 38, partially decrypted encryptions of 123 and 49. Note that these are sent in a random order, so that the miner has no idea which respondent corresponds to which pieces of partially decrypted data.
- Decryption
  In this phase, the miner decrypts the data received from the DCH. The result is k-anonymous cleartext data.In the example we have considered, the miner simply decrypts all the partially decrypted encryptions he has received, and the result is a random permutation of ((123,55), (224, 38), (123, 49)).

In summary, kADC illustrated in Protocol 1 allows the miner to collect a k-anonymized version of the respondents' data in such a way that the miner cannot figure out which respondent submits which piece of sensitive data. In this procedure, we use the *suppression* method to achieve k-anonymity. We note that there are other ways to k-anonymize data, e.g., *generalization*. However, it will be much more challenging to build the kADC protocol using k-anonymization methods like generalization. The reason is that the operations involved in generalization is much more complicated than those involved in supression. When we use suppression to anonymize data, the main job is to count the number of occurrences of each distinct quasi-identifier; as we have seen, it is not very hard to make this operation privacy-preserving. In contrast, the generaliza-

tion method involves operations like choosing generalized identifiers and using them to replace the original ones; these operations are so complicated that it is really hard to design a privacy-preserving protocol that performs them very efficiently. Hence, we leave this widely open problem to future work.

3.3. Protocol analysis

In this subsection, we analyze our protocol in terms of correctness and privacy against the miner and the DCH. In order to formalize our analysis of k-anonymity, we first define the k-anonymous subset of each respondent as follows:

- If $|\{i : |\{j : d_j^+ = d_i^+, j \in \{1,\ldots,N\}\}| \geqslant k, i \in \{1,\ldots,N\}\}| = N$ or $|\{i : |\{j : d_j^+ = d_i^+, j \in \{1,\ldots,N\}\}| \geqslant k, i \in \{1,\ldots,N\}\}| \leqslant N - k$, then
  - –For all i such that $|\{j : d_j^+ = d_i^+, j \in \{1,\ldots,N\}\}| \geqslant kK(i) = \{j : d_j^+ = d_i^+, j \in \{1,\ldots,N\}\}$.
  - –For all i such that $|\{j : d_j^+ = d_i^+, j \in \{1,\ldots,N\}\}| < kK(i) = \{i' : |\{j : d_j^+ = d_{i'}^+, j \in \{1,\ldots,N\}\}| < k, i' \in \{1,\ldots,N\}\}$.
- If $N - k < |\{i : |\{j : d_j^+ = d_i^+, j \in \{1,\ldots,N\}\}| \geqslant k, i \in \{1,\ldots,N\}\}| < N$ then
  - –For all i such that $|\{j : d_j^+ = d_i^+, j \in \{1,\ldots,N\}\}| > \min_{|\{j:d_j^+ = d_{i'}^+, j\in\{1,\ldots,N\}\}|\geqslant k}|\{j : d_j^+ = d_{i'}^+, j \in \{1,\ldots,N\}\}|K(i) = \{j : d_j^+ = d_i^+, j \in \{1,\ldots,N\}\}$.
  - –For all the remaining $iK(i)$ is the set of such i.

  Denote by d_i'''' and $d_i^{\diamondsuit}$ the decryptions of $\overline{d_i''''}$ and $\overline{d_i^{\diamondsuit}}$, respectively.

Theorem 3.1 (Correctness). *If all parties follow the protocol, then with high probability the output satisfies that* $\forall i$,

$$d_i' = \begin{cases} d^+_{\pi(i)} & \text{if} \forall j \in K(\pi(i)),\ d_j^+ = d^+_{\pi(i)}; \\ (\bigstar, \bigstar, \ldots, \bigstar) & \text{otherwise}, \end{cases}$$

and that $\forall i d_i^{\diamondsuit} = d^-_{\pi(i)}$.

Proof. First, we observe that

$$\begin{aligned} q'_{i,j} &= \overline{q''_{i,j}}[1]/(\overline{q''_{i,j}}[2])^u \\ &= \frac{\overline{q'_{i,j}}[1]/(\overline{q'_{i,j}}[2])^x}{(\overline{q'_{i,j}}[2])^u} \\ &= \frac{\overline{q}_{i,\theta_i(j)}[1]}{(\overline{q}_{i,\theta_i(j)}[2])^{x+u}} \\ &= \frac{(\overline{d_i^+}[1]/\overline{d^+_{\theta_i(j)}}[1])^{r_{i,\theta_i(j)}}}{(\overline{d_i^+}[2]/\overline{d^+_{\theta_i(j)}}[2])^{(x+u)r_{i,\theta_i(j)}}} \\ &= (d_i^+/d^+_{\theta_i(j)})^{r_{i,\theta_i(j)}}. \end{aligned}$$

With high probability, we have

$$d_i = d_{\theta_i(j)} \Longleftrightarrow q'_{i,j} = 1.$$

Therefore, with high probability, $c_i = |j : d_j^+ = d_i^+, j \in \{1,\ldots,N\}|$. Consequently, it is easy to see that, if $\forall j \in K(i),\ d_j^+ = d_i^+$, then $\overline{d_i''} = \overline{d_i^+}$; otherwise, $\overline{d_i''}$ is an encryption of $(\bigstar, \bigstar, \ldots, \bigstar)$.

On the other hand, for any i, clearly we have

$$\begin{aligned} d_i'''' &= \overline{d_i''''}[1]/(\overline{d_i''''}[2])^x \\ &= \overline{d''''_{\pi(i)}}[1]/(\overline{d''''_{\pi(i)}}[2])^x \\ &= \frac{\overline{d''_{\pi(i)}}[1]/(\overline{d''_{\pi(i)}}[2])^u}{(\overline{d''_{\pi(i)}}[2])^x} \\ &= \frac{\overline{d''_{\pi(i)}}[1]}{(\overline{d''_{\pi(i)}}[2])^{x+u}}. \end{aligned}$$

If $\forall j \in K(\pi(i)),\ d_j^+ = d^+_{\pi(i)}$, then we have

$$\begin{aligned} d_i'''' &= \frac{\overline{d^+_{\pi(i)}}[1]}{(\overline{d^+_{\pi(i)}}[2])^{x+u}} \\ &= d^+_{\pi(i)}. \end{aligned}$$

Otherwise, we have $d_i'''' = (\bigstar, \bigstar, \ldots, \bigstar)$.

For any i, we always have

$$\begin{aligned} d_i^{\diamond} &= \overline{d_i^{\diamond}}[1]/(\overline{d_i^{\diamond}}[2])^x \\ &= \overline{d^{\sharp}_{\pi(i)}}[1]/(\overline{d^{\sharp}_{\pi(i)}}[2])^x \\ &= \frac{\overline{d^{-}_{\pi(i)}}[1]/(\overline{d^{-}_{\pi(i)}}[2])^u}{(\overline{d^{-}_{\pi(i)}}[2])^x} \\ &= \frac{\overline{d^{-}_{\pi(i)}}[1]}{(\overline{d^{-}_{\pi(i)}}[2])^{x+u}} \\ &= d^{-}_{\pi(i)}. \end{aligned}$$

This concludes the proof. □

Theorem 3.2 (Privacy against miner). *The basic solution is k-anonymous against the miner in the semi-honest model.*

Proof. We prove this by contradiction. Assume that this protocol is not k-anonymous against the miner. Based on this protocol, we give a probabilistic polynomial-time algorithm for the miner that distinguishes the ElGamal encryptions of two different cleartexts under public key yv, which contradicts the semantic security of ElGamal encryption [53].

The above assumption of the protocol being not k-anonymous against the miner means that there exist $(d_1,\ldots,d_N)$ and $i\in\{1,\ldots,N\}$ such that for all $I\subseteq\{1,\ldots,N\}(i\in I|I|\geqslant k)$, there exist a permutation σ on I, a probabilistic polynomial-time distinguisher X, and a polynomial $f()$ such that for infinitely many κ,

$$Pr[X(\mathsf{view}_{\mathsf{miner}}((d_1^+,d_1^-),\ldots,(d_N^+,d_N^-)))=1]-Pr[X(\mathsf{view}_{\mathsf{miner}}((d_1^+,d^-_{\sigma(1)}),\ldots,(d_N^+,d^-_{\sigma(N)})))=1]>1/f(\kappa).$$

Note that X implicitly takes all the public parameters and the miner's private key as input. Using the hybrid argument [21], we can easily show that the above implies that there exist $(d_1,\ldots,d_N)$ and $i\in\{1,\ldots,N\}$ such that for all $I\subseteq\{1,\ldots,N\}(i\in I|I|\geqslant k)$, there exist a permutation σ_0 on I that only switches two indices α and $\beta(\alpha,\beta\in I)$, a probabilistic polynomial-time distinguisher X, and a polynomial $f()$ such that for infinitely many κ,

$$Pr[X(\mathsf{view}_{\mathsf{miner}}((d_1^+,d_1^-),\ldots,(d_N^+,d_N^-)))=1]-Pr[X(\mathsf{view}_{\mathsf{miner}}((d_1^+,d^-_{\sigma_0(1)}),\ldots,(d_N^+,d^-_{\sigma_0(N)})))=1]>1/f(\kappa). \tag{3.1}$$

We fix I as the $K(i)$ of $(d_1,\ldots,d_N)$. Then, $(d_1^+d^-_{\sigma_0(1)}),\ldots,(d_N^+,\ d^-_{\sigma_0(N)})$ has the same $K(i)$. Furthermore, for all $j\in\{1,\ldots,N\}(d_1,\ldots,d_N)$ and $(d_1^+,d^-_{\sigma_0(1)}),\ldots,(d_N^+d^-_{\sigma_0(N)})$ have the same $K(j)$. So in the sequel, we will refer to $K(i)$ and $K(j)$ without mentioning the corresponding cleartext data. Note that $K(i)$ and $K(j)$ can be easily computed.

Below we give a probabilistic polynomial-time algorithm A that distinguishes an ElGamal encryption of d_α^- under public key yv from an ElGamal encryption of d_β^- under public key yv.

On input ciphertext e, A first computes, using the homomorphic property of ElGamal, another ciphertext e' such that the product of the cleartexts of e and e' is equal to $d_\alpha^-\cdot d_\beta^-$: A computes a random encryption of $d_\alpha^-\cdot d_\beta^-$ and then divides it by e. Then A rerandomizes e' to get e''. Next, A simulates two executions of our protocol; A extracts the view of the adversary generated in each simulated execution and applies X to it. The simulation is detailed as follows.

In Phase 1 of the protocol, for all j except α and βA simulates respondent j using a process with input (d_j^+,d_j^-); the process works exactly as described in the protocol. A simulates respondents α and β with two processes that have a mixture of cleartext and ciphertext inputs; these two processes do not encrypt their ciphertext inputs as described in the protocol but directly send out their ciphertext inputs to the simulated miner together with encryptions of their cleartext inputs. In both of the simulated executions, respondent α receives cleartext input d_α^+ and respondent β receives cleartext input d_β^+. During the first simulated execution, the simulated respondent α starts with ciphertext e and the simulated respondent β starts with e''; during the second execution, the simulated respondent α starts with ciphertext e'' and the simulated respondent β starts with e.

In Phase 2, A simulates the miner's operations as described in the protocol. Phase 3 is performed by the DCH. The only thing in the miner's view is the messages sent at the end of this phase. We include the simulation of these messages in the simulation of Phase 4.

In Phase 4, A chooses a random permutation ρ on $\{1,\ldots,N\}$. For each jA defines $d_j^{\diamond}$ as $d^-_{\rho(j)}$. If for all $j'\in K(j), d^+_{\rho(j')}=d^+_{\rho(j)}A$ defines $d_j^{'''}=d^+_{\rho(j)}$; otherwise, A defines $d_j^{'''}=(\star,\star,\ldots,\star)$. A defines the final output of the entire simulated protocol as $((d_1^{'''},d_1^{\diamond}),\ldots,(d_N^{'''},d_N^{\diamond}))$. (Note that we get exactly the same distribution of simulated output if A defines $d_j^{\diamond}$ as $d^-_{\rho(\sigma_0(j))}$, because $\alpha,\beta\in K(i)$ and thus $d_\alpha^{'''}=d_\beta^{'''}$.) Then, A can simulate the messages sent at the end of Phase 3 using random encryptions of these cleartexts under public key y. The coin flips in this phase can be easily computed from the simulated final output and the simulated messages at the end of Phase 3.

Applying X to the views of the adversary generated in the simulated executions, A can compute

$$o_1=X(\mathsf{view}_{\mathsf{miner}}((d_1^+,d_1^-),\ldots,(d^+_{\alpha-1},d^-_{\alpha-1}),(d^+_\alpha,D(e)),(d^+_{\alpha+1},d^-_{\alpha+1}),\ldots,(d^+_{\beta-1},d^-_{\beta-1}),(d^+_\beta,D(e'')),(d^+_{\beta+1},d^-_{\beta+1}),\ldots,(d_N^+,d_N^-))),$$

and

$$o_2 = X(\text{view}_{\text{miner}}((d_1^+, d_1^-), \ldots, (d_{\alpha-1}^+, d_{\alpha-1}^-), (d_\alpha^+, D(e'')), (d_{\alpha+1}^+, d_{\alpha+1}^-), \ldots, (d_{\beta-1}^+, d_{\beta-1}^-), (d_\beta^+, D(e)), (d_{\beta+1}^+, d_{\beta+1}^-), \ldots, (d_N^+, d_N^-))),$$

where $D(e)$ denotes the decryption of e. If $o_1 = 1$ and $o_2 = 0$$A$ outputs 1; if $o_1 = 0$ and $o_2 = 1$$A$ outputs 0; otherwise A outputs a uniformly random bit.

Now we analyze the probabilities of outputing 1 with input ciphertext of d_α^- or d_β^-. For convenience, let

$$p_1 = \Pr[X(\text{view}_{\text{miner}}((d_1^+, d_1^-), \ldots, (d_N^+, d_N^-))) = 1],$$

and

$$p_2 = \Pr[X(\text{view}_{\text{miner}}((d_1^+, d_{\sigma_0(1)}^-), \ldots, (d_N^+, d_{\sigma_0(N)}^-))) = 1].$$

When the input ciphertext is an encryption of d_α^-, the probability that we have output equals 1 is

$$\Pr[A(d_\alpha^-) = 1] = p_1(1-p_2) + p_1p_2/2 + (1-p_1)(1-p_2)/2.$$

When the input ciphertext is an encryption of d_β^-, the probability that we have output equals 1 is

$$\Pr[A(d_\beta^-) = 1] = p_2(1-p_1) + p_2p_1/2 + (1-p_2)(1-p_1)/2.$$

Combining the above two equations, we have

$$\begin{aligned} Pr[A(d_\alpha^-) = 1] - Pr[A(d_\beta^-) = 1] &= p_1(1-p_2) + p_1p_2/2 + (1-p_1)(1-p_2)/2 - (p_2(1-p_1) + p_2p_1/2 + (1-p_2)(1-p_1)/2) \\ &= p_1 - p_2 > \frac{1}{f(\kappa)}. \end{aligned}$$

The last inequality is due to Eq. (3.1). However, this contradicts the semantic security of ElGamal. □

Theorem 3.3 (Privacy against DCH). *The basic solution is k-anonymous against the DCH in the semi-honest model.*

Proof. Again, we show this by contradiction. Assume that this protocol is not k-anonymous against the DCH. Based on this protocol, we can construct a probabilistic polynomial-time algorithm for the DCH that distinguishes the ElGamal encryptions of two different cleartexts under public key yv.

The above assumption of the protocol being not k-anonymous against the DCH means that there exist $(d_1, \ldots, d_N)$ and $i \in \{1, \ldots, N\}$ such that for all $I \subseteq \{1, \ldots, N\}(i \in I|I| \geqslant k)$, there exist a permutation σ on I, a probabilistic polynomial-time distinguisher X, and a polynomial $f()$ such that for infinitely many κ,

$$Pr[X(\text{view}_{DCH}((d_1^+, d_1^-), \ldots, (d_N^+, d_N^-))) = 1] - Pr[X(\text{view}_{DCH}((d_1^+, d_{\sigma(1)}^-), \ldots, (d_N^+, d_{\sigma(N)}^-))) = 1] > 1/f(\kappa).$$

In the above, X implicitly takes all the public parameters and the DCH's private key as input. Using the hybrid argument, we can easily show that the above implies that there exist $(d_1, \ldots, d_N)$ and $i \in \{1, \ldots, N\}$ such that for all $I \subseteq \{1, \ldots, N\}(i \in I|I| \geqslant k)$, there exist a permutation σ_0 on I that only switches two indices α and $\beta(\alpha, \beta \in I)$, a probabilistic polynomial-time distinguisher X, and a polynomial $f()$ such that for infinitely many κ,

$$Pr[X(\text{view}_{DCH}((d_1^+, d_1^-), \ldots, (d_N^+, d_N^-))) = 1] - Pr[X(\text{view}_{DCH}((d_1^+, d_{\sigma_0(1)}^-), \ldots, (d_N^+, d_{\sigma_0(N)}^-))) = 1] > 1/f(\kappa). \quad (3.2)$$

Below we give a probabilistic polynomial-time algorithm A that distinguishes an ElGamal encryption of d_α^- under public key yv from an ElGamal encryption of d_β^- under public key yv.

On input ciphertext e, A first computes, using the homomorphic property of ElGamal, another ciphertext e' such that the product of the cleartexts of e and e' is equal to $d_\alpha^- \cdot d_\beta^-$ and then rerandomizes e' to get e''. Next, A simulates two executions of our protocol; A extracts the view of the adversary generated in each simulated execution and applies X to it. The simulation is detailed as follows.

In both simulated executions, the $\overline{d_1^+}, \ldots, \overline{d_N^+}$ the DCH receives is simulated by random encryptions of $d_1^+, \ldots, d_N^+$; for each $j \neq \alpha, \beta$, the $\overline{d_j^-}$ the DCH receives is simulated by a random encryption of d_j^-;

In the first simulated execution, the $\overline{d_\alpha^-}, \overline{d_\beta^-}$ the DCH receives is simulated by e, e''. In the second simulated execution, the $\overline{d_\alpha^-}, \overline{d_\beta^-}$ the DCH receives is simulated by e'', e.

In both simulated executions, for each j, the $\overline{q''_{j,1}}, \ldots, \overline{q''_{j,N}}$ the DCH receives is simulated by $|\{j' : d_{j'}^+ = d_j^+, j' \in \{1, \ldots, N\}\}|$ random encryptions of 1 under public key u and $N - |\{j' : d_{j'}^+ = d_j^+, j' \in \{1, \ldots, N\}\}|$ random ciphertexts, all in a random order.

Then A simulates the protocol execution as described in the Phase 2 of the protocol. This is enough to obtain the view of DCH.

Applying X to the views of the adversary generated in the simulated executions, A can compute

$$o_1 = X(\text{view}_{DCH}((d_1^+, d_1^-), \ldots, (d_{\alpha-1}^+, d_{\alpha-1}^-), (d_\alpha^+, D(e)), (d_{\alpha+1}^+, d_{\alpha+1}^-), \ldots, (d_{\beta-1}^+, d_{\beta-1}^-), (d_\beta^+, D(e'')), (d_{\beta+1}^+, d_{\beta+1}^-), \ldots, (d_N^+, d_N^-))),$$

and

$$o_2 = X(\mathrm{view}_{DCH}((d_1^+, d_1^-), \ldots, (d_{\alpha-1}^+, d_{\alpha-1}^-), (d_\alpha^+, D(e'')), (d_{\alpha+1}^+, d_{\alpha+1}^-), \ldots, (d_{\beta-1}^+, d_{\beta-1}^-)), (d_\beta^+, D(e)), (d_{\beta+1}^+, d_{\beta+1}^-), \ldots, (d_N^+, d_N^-))),$$

If $o_1 = 1$ and $o_2 = 0A$ outputs 1; if $o_1 = 0$ and $o_2 = 1$, A outputs 0; otherwise A outputs a uniformly random bit.

The remaining probability analysis is identical to that in the proof of Theorem 3.2. □

Remark 3.4. Theorems 3.2 and 3.3 guarantees k-anonymity of our basic solution. We note that the k-anonymity achieved here is *not* optimal. Compared with a (good) centralized algorithm for k-anonymization, our solution needs to suppress more information in the quasi-identifier attributes. In Section 5, we present an improved protocol that suppresses less information.

Remark 3.5. In terms of privacy, it is worth noting that even if our kADC protocol is used, the privacy of respondents may still be violated by some attacks on k-anonymity. For example, k-anonymity may not protect respondents' privacy when there is little diversity in the data, or when some background knowledge is available (see, e.g., [39]). However, we emphasize that these are the privacy weaknesses of k-anonymity in general, not the privacy weaknesses of our kADC protocol. What we can guarantee is that kADC provides the maximum amount of privacy protection that can be provided by any protocol based on k-anonymity. We believe this is a practical approach because k-anonymity is simple and widely used, although the weaknesses of k-anonymity have been found. Hence, we focus on k-anonymous data collection in this paper. In the future, we may be able to extend our data collection approach to stronger privacy concepts (such as l-diversity).

3.4. Practical efficiency analysis

Before we start our efficiency analysis, it is important that we revisit our definition of N. In the above, we have stated that N is the number of respondents in the protocol. In practice, it is nearly impossible to have all respondents online simultaneously; thus N is *not* the total number of respondents, but the number of respondents involved in one execution of the kADC protocol. Hereafter we call the users involved in one execution of the protocol a *user group*. Therefore, N is the size of a user group.

In our kADC protocol, each respondent has the advantage of "submit-and-go": once they submit their encrypted data, they can leave this protocol without waiting for the finish of the protocol. Each respondent's overhead is just two ElGamal encryptions which can be efficiently computed. The miner's overhead comes from the second and four phases. It is dominated by $3N^2 + 2N$ modular exponentiations. The DCH's overhead comes from the third phase. It is dominated by $N^2 + 2N \sim N^2 + 4N$ modular exponentiations. Here, we can see that the miner and the DCH carry out most computational tasks in this protocol. This is reasonable in the reality because the miner and the DCH are usually professionals who have more powerful computers than the users.

To measure the efficiency, we implement this basic protocol using the OpenSSL library. We test our implementation in a machine running NetBSD with 512 MB memory and 2 GHZ AMD CPU. In our implementation, we use 512-bit cryptographic key. For different combinations of k and N, we test the computational overheads of each respondent, of the miner, and of the DCH, respectively.

Fig. 1 shows each respondent's computational overhead. It is approximately 5.5×10^{-3} s regardless of N and k. Figs. 2 and 3 show the miner's and DCH's computational overheads, respectively. We can see they are quadratic in the number of

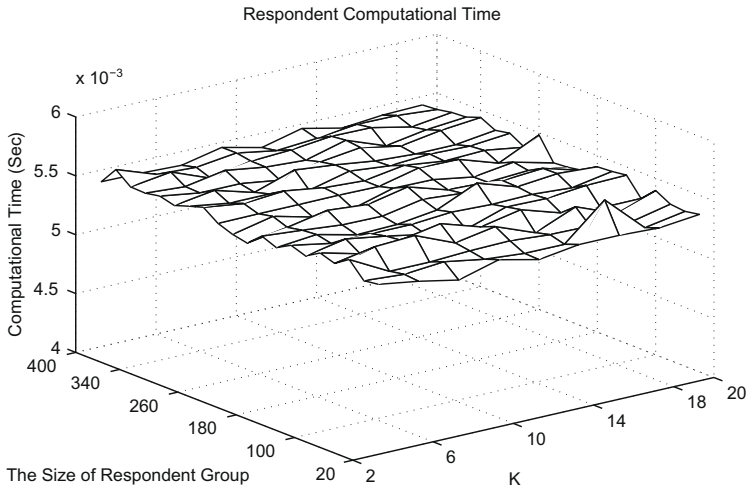

Fig. 1. Respondent's computation time.

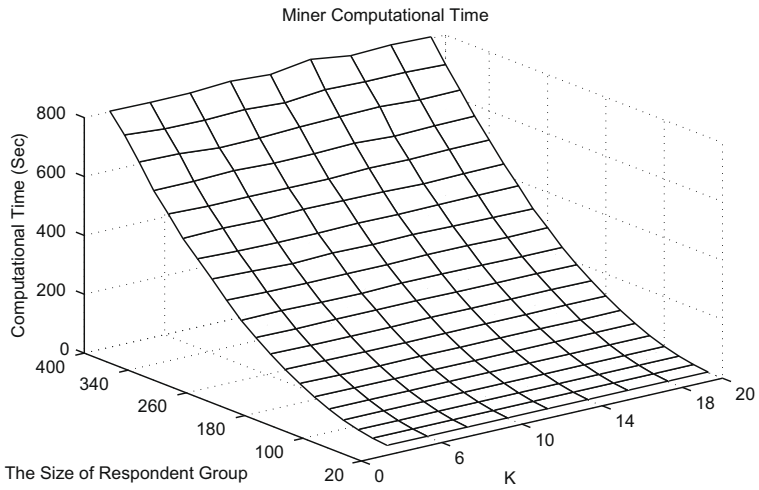

Fig. 2. Miner's computation time.

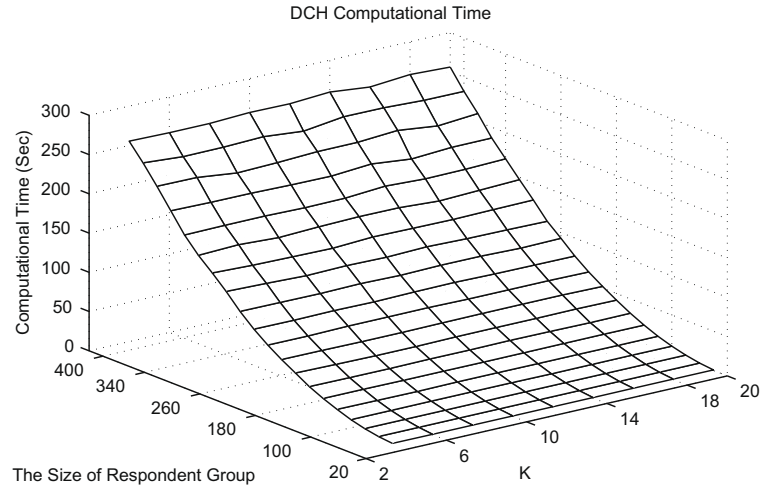

Fig. 3. DCH's computation time.

respondents; the effect of different k on the overheads is very small and thus can hardly be observed. For $N = 400$, the miner needs about 800 s while the DCH needs about 250 s. We stress that such overheads are acceptable because the respondents do not need to wait for the miner and the DCH to complete these computations. We can also see that, for all combinations of N and k, the miner's overhead is about 3.2 times as the DCH's.

To reduce the miner's overhead, we can optimize our protocol using precomputation of some intermediate results. Recall that, to rerandomize an ElGamal encryption, first the miner needs to compute some intermediate results by raising g and the public key to a random number. In practice, the miner can precompute these intermediate results because these computations do not depend on the user data. Fig. 4 shows the miner's computational overhead after using optimization. We can see that the miner's computational time is reduced to about 30% of the original.

4. Extension to the fully malicious model

As we have mentioned, our basic solution works only in the semi-honest model. However, we can use (digital signatures and) zero-knowledge proofs to extend the basic solution to the fully malicious model. In this section, we describe how to prevent the malicious behavior of the miner and the DCH, respectively. For easiness of presentation, we give all our zero-knowledge proofs in the *interactive* form. Note that we can make all these proofs *non-interactive* using the Fiat–Shamir heuristics [19].

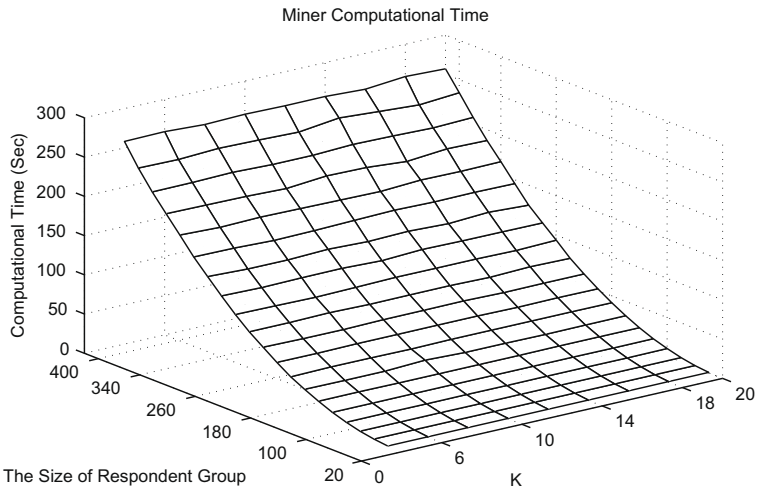

Fig. 4. Miner's computation time with optimization.

4.1. Building blocks

To prevent the miner and the DCH from deviating from the protocol, we need to use two building blocks for multiple times: zero-knowledge proof of permuted decryption and zero-knowledge proof of permuted rerandomization. Before we describe our extension of the basic solution, we first give a brief explanation of these two building blocks.

4.1.1. Proof of permuted decryption

Suppose that there are a prover and a verifier. Let χ and ψ be the prover's private key and public key, respectively. The prover needs to show that $(\omega_1,\ldots,\omega_n)$ is a *permuted decryption* of $(\overline{\omega_1},\ldots,\overline{\omega_n})$ under her own private key. That is, the prover needs to show that $(\omega_1,\ldots,\omega_n)$ is a permutation of the cleartexts of $(\overline{\omega_1},\ldots,\overline{\omega_n})$.

To achieve this goal, the prover chooses a permutation ξ on $\{1,\ldots,n\}$; for $i=1,\ldots,n$, she sets $\overline{\omega_i'}$ to a rerandomization of $\overline{\omega_{\xi(i)}}$. The prover sends $(\overline{\omega_1'},\ldots,\overline{\omega_n'})$ to the verifier. The verifier sends back a uniformly random bit to the prover as the first-round challenge.

If the first-round challenge is 0, the prover sends ξ to the verifier, together with the random numbers she used to rerandomize all $\overline{\omega_i}$. The verifier checks that each $\overline{\omega_i'}$ is indeed a rerandomization of $\overline{\omega_{\xi(i)}}$.

If the first-round challenge is 1, the prover does the follows. Since $(\omega_1,\ldots,\omega_n)$ is a permuted decryption of $(\overline{\omega_1},\ldots,\overline{\omega_n})$ under private key χ, there exists a permutation ξ' on $\{1,\ldots,n\}$ such that each ω_i is the decryption of $\overline{\omega_{\xi'(i)}}$ under private key χ. So the prover sends $(\xi')^{-1}\xi$ to the verifier. In addition, for each i, the prover chooses a random exponent γ_i and computes

$$\lambda_{i,0} = g^{\gamma_i},$$
$$\lambda_{i,1} = \overline{\omega_i'}[2]^{\gamma_i}.$$

The prover sends $\lambda_{i,0},\lambda_{i,1}$ to the verifier.

For each i, the verifier chooses a random number $\lambda_{i,2}$. The verifier sends these random numbers back as the second-round challenge. The prover computes

$$\lambda_{i,3} = \gamma_i + \lambda_{i,2}\chi.$$

Then, the prover sends $\lambda_{1,3},\ldots,\lambda_{n,3}$ to the verifier. The verifier checks that, for each i,

$$g^{\lambda_{i,3}} = \lambda_{i,0}\psi^{\lambda_{i,2}},$$
$$\overline{\omega_i'}[2]^{\lambda_{i,3}} = \lambda_{i,1}\left(\frac{\overline{\omega_i'}[1]}{\omega_{(\xi')^{-1}\xi(i)}}\right)^{\lambda_{i,2}}.$$

To minimize the probability of cheating, the above procedure should be repeated for a number of times. The probability of successful cheating decreases exponentially in the number of times the above procedure is repeated.

4.1.2. Proof of permuted rerandomization

Again, suppose that there are a prover and a verifier. The prover needs to show that $(\overline{\omega_1'},\ldots,\overline{\omega_n'})$ is a *permuted rerandomization* of $(\overline{\omega_1},\ldots,\overline{\omega_n})$ under the public key ψ. That is, the prover needs to show that $(\overline{\omega_1'},\ldots,\overline{\omega_n'})$ is a permutation of the rerandomizations of $(\overline{\omega_1},\ldots,\overline{\omega_n})$.

To achieve this goal, the prover chooses a permutation ξ on $\{1,\ldots,n\}$; for $i=1,\ldots,n$, she sets $\overline{\omega_i''}$ to a rerandomization of $\overline{\omega_{\xi(i)}}$ under the public key ψ. The prover sends $(\overline{\omega_1''},\ldots,\overline{\omega_n''})$ to the verifier. The verifier sends back a uniformly random bit to the prover as the first-round challenge.

If the first-round challenge is 0, the prover sends ξ to the verifier, together with the random numbers she used to rerandomize each $\overline{\omega_{\xi(i)}}$ to get $\overline{\omega_i''}$. The verifier checks that each $\overline{\omega_i''}$ is indeed a rerandomization of $\overline{\omega_{\xi(i)}}$.

If the first-round challenge is 1, the prover does the follows. Since $(\overline{\omega_1'},\ldots,\overline{\omega_n'})$ is a permuted rerandomization of $(\overline{\omega_1},\ldots,\overline{\omega_n})$, there exists a permutation ξ' on $\{1,\ldots,n\}$ such that each $\overline{\omega_i'}$ is a rerandomization of $\overline{\omega_{\xi'(i)}}$ under public key ψ. So the prover sends $(\xi')^{-1}\xi$ to the verifier. In addition, for each i, the prover chooses a random exponent γ_i and computes

$$\lambda_{i,0}=g^{\gamma_i},$$
$$\lambda_{i,1}=\psi^{\gamma_i}.$$

The prover sends $\lambda_{i,0},\lambda_{i,1}$ to the verifier.

For each i, the verifier chooses a random number $\lambda_{i,2}$. The verifier sends these random numbers back as the second-round challenge. Suppose φ_i is the discrete logarithm of $\frac{\overline{\omega'_{(\xi')^{-1}\xi(i)}}[2]}{\overline{\omega_i''}[2]}$ with respect to base g. Note that φ_i can be easily computed from the random numbers the prover uses in rerandomizations. The prover computes

$$\lambda_{i,3}=\gamma_i+\lambda_{i,2}\varphi_i.$$

Then, the prover sends $\lambda_{1,3},\ldots,\lambda_{n,3}$ to the verifier. The verifier checks that, for each i,

$$g^{\lambda_{i,3}}=\lambda_{i,0}\left(\frac{\overline{\omega'_{(\xi')^{-1}\xi(i)}}[2]}{\overline{\omega_i''}[2]}\right)^{\lambda_{i,2}},$$
$$\psi^{\lambda_{i,3}}=\lambda_{i,1}\left(\frac{\overline{\omega'_{(\xi')^{-1}\xi(i)}}[1]}{\overline{\omega_i''}[1]}\right)^{\lambda_{i,2}}.$$

To minimize the probability of cheating, the above procedure should also be repeated for a number of times.

4.2. Preventing the miner's malicious behavior

Given the building blocks, now we can prevent the miner's malicious behavior.

In Phase 2, we need to make sure: (1) the miner forwards each $\overline{d_i^+},\overline{d_i^-}$ from respondent i without tampering; (2) for each i, the miner computes $\overline{q_{i,1}''},\ldots,\overline{q_{i,N}''}$ properly.

To ensure that the miner forwards each $\overline{d_i^+},\overline{d_i^-}$ without tampering, we only needs to add the following operations to our protocol. At the end of Phase 1, each respondent should sign her message. At the end of Phase 2, the miner should forward the respondents' signatures together with $\overline{d_i^+},\overline{d_i^-}$, so that the DCH can verify these signatures.

To ensure that the miner computes $\overline{q_{i,1}''},\ldots,\overline{q_{i,N}''}$ properly, we first make a minor change to Phase 2: Instead of having $\overline{q_{i,j}'}=\overline{q_{i,\theta_i(j)}}$, the miner should set $\overline{q_{i,j}'}$ to a rerandomization of $\overline{q_{i,\theta_i(j)}}$. Then the miner should prove that he properly computes each $\overline{q_{i,j}}$ from $\overline{d_i^+}$ and $\overline{d_j^+}$, that he properly computes $(\overline{q_{i,1}'},\ldots,\overline{q_{i,N}'})$ from $(\overline{q_{i,1}},\ldots,\overline{q_{i,N}})$, and that he properly computes each $\overline{q_{i,j}''}$ from $\overline{q_{i,j}'}$. The first proof can be done using a standard proof of knowledge of discrete logarithm [46]. The second proof can be done using the technique given in 4.1.2. The third proof can be done using a standard proof of ElGamal decryption [29].

In Phase 4, the miner only needs to prove that the output consists of a permuted decryption of $(\overline{d_1''''},\ldots,\overline{d_N''''})$ and a permuted decryption of $(\overline{d_1^\diamondsuit},\ldots,\overline{d_N^\diamondsuit})$. This can be done using the technique given in Section 4.1.1.

4.3. Preventing the DCH's malicious behavior

Before we show how to prevent the DCH's malicious behavior, we make minor changes to Phase 3 for computing $\overline{d_i''}$, $\overline{d_i'''d_i^\sharp}$, $\overline{d_i''''}$, and $\overline{d_i^\diamondsuit}$.

If $1\leqslant|\{i:c_i<k-1\}|<k$, then the DCH does the follows: Recall C be the smallest c_i that is greater than $k-1$. For all i such that $c_i=C$, the DCH still sets $\overline{d_i'''}$ to an encryption of $(\star,\star,\ldots,\star)$ under public key yv; but for all other i, the DCH sets $\overline{d_i'''}$ to a rerandomization of $\overline{d_i'}$. If $|\{i:c_i<k-1\}|\geqslant k$ or $|\{i:c_i<k-1\}|=0$, then the DCH similarly defines $\overline{d_i'''}$ as a rerandomization of $\overline{d_i'}$ for all i.

For each i, the DCH chooses a permutation π on $\{1,\ldots,N\}$ uniformly at random. The DCH sets each $\overline{d_i''''}$ to a rerandomization of $\overline{d'''_{\pi(i)}}$ and each $\overline{d_i^\sharp}$ to a rerandomization of $\overline{d_{\pi(i)}^-}$. Then, the DCH computes $\overline{d_i''''}[1]=\overline{d_i'''}[1]/(\overline{d_i'''}[2])^u$ and $\overline{d_i^\diamondsuit}[1]=\overline{d_i^\sharp}[1]/(\overline{d_i^\sharp}[2])^u$. The DCH defines, for all i, $\overline{d_i''''}[2]=\overline{d_i'''}[2]$ and $\overline{d_i^\diamondsuit}[2]=\overline{d_i^\sharp}[2]$.

Given the above modification to Phase 3, the DCH gives his zero knowledge proofs as follows. First, the DCH proves, using standard techniques [6], that each $\overline{d_i''}$ is either a rerandomization of $\overline{d_i^+}$ or an encryption of $(\star,\star,\ldots,\star)$, and that the cleartext

of each $\overline{d_i''}$ appears at least k times.[2] Then, the DCH proves that he computes $\overline{d_1'''}, \ldots, \overline{d_N'''}$ properly from $\overline{d_1''}, \ldots, \overline{d_N''}$ using the technique given in Section 4.1.2. Similarly, the DCH proves that he computes $\overline{d_1^\sharp}, \ldots, \overline{d_N^\sharp}$ properly from $\overline{d_1^-}, \ldots, \overline{d_N^-}$ using the technique given in Section 4.1.2. Next, the DCH proves that he computes $\overline{d_1''''}, \ldots, \overline{d_N''''}$ properly from $\overline{d_1'''}, \ldots, \overline{d_N'''}$ using the technique given in Section 4.1.1. Similarly, the DCH proves that he computes $\overline{d_1^\diamondsuit}, \ldots, \overline{d_N^\diamondsuit}$ properly from $\overline{d_1^\sharp}, \ldots, \overline{d_N^\sharp}$ using the technique given in Section 4.1.1.

4.4. Computational overhead analysis

Now we give a brief analysis of our extended protocol for the malicious model. Since our theoretical analysis of the basic protocol's computational overheads is consistent with the experimental results, for the extended protocol we only provide a theoretical analysis of overheads. Interested readers can compare it with our results for the basic protocol to estimate the amounts of time needed in practice.

In the extended protocol, for each respondent, the additional computational overhead (compared with the basic protocol) is just the signing of one single message. For the miner, the additional computational overhead consists of N^2 rerandomization operations and 4 zero-knowledge proofs. Each rerandomization is dominated by 2 modular exponentiations. The computation cost of the zero-knowledge proofs in 4.11 and 4.12 are both dominated by by $O(N)$ modular exponentiations; the computation cost of the proofs in [29] and [46] are both dominated by $O(N^2)$ modular exponentiations. Therefore, the total additional overhead of the miner is about $O(N^2)$ modular exponentiations. For the DCH, the overhead consists of $2N \sim 4N$ rerandomizations and 6 zero-knowledge proofs in which the proof described in [6] is also dominated by $O(N)$ modular exponentiations. So the total additional overhead of the DCH is about $O(N)$ modular exponentiations.

5. Improved solution

In Section 3, we have presented a basic protocol for kADC. However, the basic protocol suppresses a considerable amount of information in the quasi-identifiers, because its k-anonymization is very coarse-grained: it either does not change a quasi-identifier, or completely suppresses it. In practice, suppressing too much information is often undesirable. Consequently, we now present an improved protocol that suppresses significantly less information.

5.1. Improved protocol

Just as in the basic protocol, each respondent i encrypts her data using public key yv:

$\overline{d_i^+} = E_{yv}(d_i^+, r_i^+)$;

$\overline{d_i^-} = E_{yv}(d_i^-, r_i^-)$;

Each respondent i submits $\overline{d_i^+}\,\overline{d_i^-}$ to the miner.

First, the miner and the DCH execute a subprotocol AnonySet() for m times to anonymize each individual attribute in the quasi-identifier: AnonySet({1}), AnonySet({2}), ..., AnonySet({m}). (See Section 5.2 for the details of this subprotocol.)

Then, for $L = 1, 2, \ldots, \lfloor \log m \rfloor$, the miner and the DCH do the follows: the miner chooses $\lfloor \frac{m}{2^L} \rfloor$ random attribute subsets of the quasi-identifier such that each subset has a size of 2^L. For each random subset M, the miner and the DCH execute the subprotocol AnonySet(M) to anonymize M.

Finally, the miner and the DCH execute AnonySet ($\{1, \ldots, m\}$) and do the follows to decrypt the data: the miner first sends all the data to the DCH. Then, for each i, the DCH decrypts $\overline{d_i^+}$ and $\overline{d_i^-}$ using his own private key u; let the results be $\widetilde{d_i^+}$ and $\widetilde{d_i^-}$. The DCH sends them back to the miner. The miner uses his own private key x to decrypt $(\widetilde{d_i^+}, \overline{d_i^+}[2])$ and $(\widetilde{d_i^-}, \overline{d_i^-}[2])$. The new results are the k-anonymized version of the collected data. Again, if the data needs to be published, the miner must publish it in a randomized order.

5.2. Subprotocol for anonymizing attribute subset M:anonyset(M)

In the previous section, we have presented our improved protocol using a subprotocol AnonySet(). Now we explain how this subprotocol works.

The input $M (M \subseteq \{1, \ldots, m\})$ of this subprotocol is a subset of quasi-identifier attributes. Denote by d_i^M the attribute values of d_i^+ in the set M. The subprotocol goes as follows.

- Miner's randomization operations.
  - –For each pair (i, j), the miner computes $\overline{q_{ij}} = (\overline{d_i^M}/\overline{d_j^M})^{r_{ij}}$, where each r_{ij} is chosen uniformly and independently.

[2] In this step, in stead of proving that he is following the protocol precisely, the DCH actually proves that he is performing k-anonymization. Therefore, there is a possibility that the DCH deviates from the protocol without being detected. However, in that case, the output of the protocol is still a k-anonymization version of the submitted data and nobody's privacy is violated. Since the target of this work is to protect privacy (rather than "security" in the broad sense), we allow this type of "benign" cheating. In reality, the DCH typically does not want to cheat in this way since it does not give the DCH any advantage.

–For each i, the miner chooses a permutation θ_i on $\{1,\ldots,N\}$ uniformly at random and computes, for each j, $\overline{q'_{ij}} = \overline{q_{i,\theta_i(j)}}$.
–The miner computes $\overline{q''_{ij}}[1] = \overline{q'_{ij}}[1]/(\overline{q'_{ij}}[2])^x$ and sets $\overline{q''_{ij}}[2] = \overline{q'_{ij}}[2]$.

–The miner sends the DCH: $\{\overline{d_i^M}\}_{i=1,\ldots,N}\{\overline{q''_{ij}}\}_{i=1,\ldots,N;j=1,\ldots,N}$.

- DCH's randomization operations.
  –For each pair (i,j), the DCH computes $q'_{ij} = \overline{q''_{ij}}[1]/(\overline{q''_{ij}}[2])^u$.
  –For each i, the DCH counts the number of j such that $q'_{ij} = 1$. Let this number be c_i. If $c_i < k-1$, then the DCH sets $\overline{d'_i}$ to an encryption of $(\bigstar,\bigstar,\ldots,\bigstar)$ under public key yv; otherwise, the DCH sets $\overline{d'_i} = d_i^M$.
  –If $1 \leqslant |\{i : c_i < k-1\}| < k$, then the DCH does the follows: Let C be the minimum value of c_i such that it is greater than $k-1$. For all i such that $c_i = C$, the DCH sets $\overline{d''_i}$ to an encryption of $(\bigstar,\bigstar,\ldots,\bigstar)$ under public key yv; for all other i, the DCH sets $\overline{d''_i} = \overline{d'_i}$. If $|\{i : c_i < k-1\}| \geqslant k$ or $|\{i : c_i < k-1\}| = 0$, then the DCH defines $\overline{d''_i} = \overline{d'_i}$ for all i.
  –For each i, the DCH sends $\overline{d''_i}$ to the miner.
- The miner replaces d_i^M with $\overline{d''_i}$.

5.3. Computational overhead analysis

In the improved protocol, each respondent has the same computational overhead as in the basic protocol. The additional computational overheads for both the miner and the DCH in the improved protocol include the execution of AnonySet and the decryption at the end. The decryption at the end is dominated by $2N$ modular exponentiations. The computational overhead for the miner for each execution of AnonySet is dominated by $2N^2$ modular exponentiations while the computational overhead for the DCH for each execution of AnonySet is dominated by N^2 modular exponentiations. Throughout the protocol, AnonySet is run for $m + \sum_{L=1}^{\lfloor \log m \rfloor} \lfloor \frac{m}{2^L} \rfloor (= 2m-1)$ times, for both the miner and the DCH. Therefore, the total additional overheads for the miner and the DCH are dominated by $2(2m-1)N^2 + 2N$ and $(2m-1)N^2 + 2N$ modular exponentiations, respectively.

6. Conclusion

In this paper, we study the kADC, a cryptographic technique for online data collection by which respondents can submit data anonymously, even if the data contains identifying information and no unidentified communication channel is available. We give a basic protocol working in the semi-honest model, an extension of the basic protocol working in the fully malicious model, and then an improved protocol that suppresses less information in quasi-identifiers. Theoretical analysis and experimental study are presented for the correctness, privacy, and efficiency of the protocol.

We notice that the k-anonymization procedure our improved protocol performs on the data is still not optimal in the amount of suppressed information. Therefore, a good future research topic is to further improve our protocol so that it suppresses even less information. Before the further improved protocol is available, if our kADC protocol suppresses too much information in an application, then we recommend falling back to APDC for data collection in that application.

References

[1] Gagan Aggarwal, Tomas Feder, Krishnaram Kenthapadi, Rajeev Motwani, Rina Panigrahy, Dilys Thomas, An Zhu, Approximation algorithms for k-anonymity, Journal of Privacy Technology, Paper Number: 20051120001, 2005.
[2] G. Aggarwal, N. Mishra, B. Pinkas, Secure computation of the kth-ranked element, in: EUROCRYPT, 2004, pp. 40–55.
[3] D. Agrawal, C. Aggarwal, On the design and quantification of privacy preserving data mining algorithms, in: Proceedings of the 20th ACM PODS, 2001, pp. 247–255.
[4] R. Agrawal, R. Srikant, Privacy-preserving data mining, in: Proceedings of the ACM SIGMOD, 2000, pp. 439–450.
[5] R. Bayardo, R. Agrawal, Data privacy through optimal k-anonymization, in: Proceedings of the 21st ICDE, 2005.
[6] Emmanuel Bresson, Jacques Stern, Proofs of knowledge for non-monotone discrete-log formulae and applications, in: Proceedings of the ISC, 2002, pp. 272–288.
[7] Justin Brickell, Vitaly Shmatikov, Efficient anonymity-preserving data collection, in: Proceedings of the ACM SIGKDD, 2006, pp. 76–85.
[8] D. Chaum, Untraceable electronic mail, return address and digital pseudonyms, Communications of the ACM 24 (2) (1981) 84–88.
[9] D. Chaum, The dining cryptographers problem: unconditional sender and recipient untraceability, Journal of Cryptology 1 (1) (1988) 65–75.
[10] C. Clifton, D. Marks, Security and privacy implications of data mining, in: Proceedings of the ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery, 1996, pp 15–19.
[11] G. Canfora, C.A. Visaggio, Tuning anonymity level for assuring high data quality: an empirical study, in: First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007), 2007, pp. 91–98.
[12] T. Dalenius, Finding a needle in a haystack or identifying anonymous census records, Journal of Official Statistics 2 (3) (1986) 329–336.
[13] I. Dinur, K. Nissim, Revealing information while preserving privacy, in: Proceedings of the 22nd ACM PODS, 2003, pp. 202–210.
[14] W. Du, Z. Zhan, Using randomized response techniques for privacy-preserving data mining, in: Proceedings of the Ninth ACM SIGKDD, 2003, pp. 505–510.
[15] C. Dwork, K. Nissim, Privacy-preserving datamining on vertically partitioned databases, in: CRYPTO 2003, 2004.
[16] A. Evfimievski, J. Gehrke, R. Srikant, Limiting privacy breaches in privacy preserving data mining, in: Proceedings 22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, 2003, pp. 211–222.
[17] A. Evfimievski, R. Srikant, R. Agrawal, J. Gehrke, Privacy preserving mining of association rules, in: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2002, pp. 217–228.
[18] B. Fung, K. Wang, P.S. Yu, Top-down specialization for information and privacy preservation, in: Proceedings of the 21st International Conference on Data Engineering, Tokyo, Japan, April 2005.

[19] A. Fiat, A. Shamir, How to prove yourself: practical solutions to identification and signature problems, in: Advances in Cryptology – Crypto'86, 1986, pp. 186–194.
[20] B. Gilburd, A. Schuster, R. Wolff, k-TTP: a new privacy model for large-scale distributed environments, in: Proceedings of the 2004 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM Press, 2004, pp. 563–568.
[21] O. Goldreich, Foundations of Cryptography, vol. 1, Cambridge University Press, 2001.
[22] O. Goldreich, Foundations of Cryptography, vol. 2, Cambridge University Press, 2003.
[23] S. Goldwasser, S. Micali, Probabilistic encryption, Journal of Computer and System Sciences 28 (1984) 270–299.
[24] P. Golle, A. Juels, Dining cryptographers revisited, in: Advances in Cryptology – EUROCRYPT, 2004, pp. 456–473.
[25] P. Golle, S. Zhong, D. Boneh, M. Jakobsson, A. Juels, Optimistic mixing for exit-polls, in: Advances in Cryptology – ASIACRYPT 2002, Springer-Verlag, 2002, pp. 451–465.
[26] HIPAA, The health insurance portability and accountability act of 1996, October 1998. <www.cms.hhs.gov/hipaa>.
[27] C. Hsu, Y. Chuang, A novel user identification scheme with key distribution preserving user anonymity for distributed computer networks, Information Sciences 179 (4) (2009) 422–429.
[28] Z. Huang, W. Du, B. Chen, Deriving private information from randomized data, in: Proceedings of the ACM SIGMOD Conference, 2005.
[29] M. Jakobsson, A practical mix, in: Proceedings of the Eurocrypt 98, 1998, pp. 448–461.
[30] M. Jakobsson, Flash mixing, in: Proceedings of the Eighteenth PODC, 1999, pp. 83–89.
[31] Qinglin Jiang, Douglas S. Reeves, Peng Ning, Improving robustness of pgp keyrings by conflict detection, in: Topics in Cryptology CT-RSA, 2004, pp. 194–207.
[32] M. Kantarcioglu, C. Clifton, Privacy-preserving distributed mining of association rules on horizontally partitioned data, in: DMKD'02, 2002, pp. 24–31.
[33] M. Kantarcioglu, J. Vaidya, An architecture for privacy-preserving mining of client information, in: Proceedings of the IEEE ICDM Workshop on Privacy, Security and Data Mining. Maebashi City, Japan, December 2002, pp. 37–42.
[34] H. Kargupta, S. Datta, Q. Wang, K. Sivakumar, On the privacy preserving properties of random data perturbation techniques. in: The Third ICDM, 2003.
[35] S. Kim, S. Park, J. Won, S. Kim, Privacy preserving data mining of sequential patterns for network traffic data, Information Sciences 178 (3) (2008) 694–713.
[36] Kristen LeFevre, David J. DeWitt, Raghu Ramakrishnan, Workload-aware anonymization, in: Proceedings of the ACM SIGKDD, 2006, pp. 277–286.
[37] B.N. Levine, C. Shields, Hordes – a multicast based protocol for anonymity, Journal of Computer Security 10 (3) (2002) 213–240.
[38] Y. Lindell, B. Pinkas, Privacy preserving data mining, Journal of Cryptology 15 (3) (2002) 177–206.
[39] A. Machanavajjhala, J. Gehrke, D. Kifer, M. Venkitasubramaniam, l-Diversity: privacy beyond *k*-anonymity, in: Proceedings of the 22nd ICDE, 2006.
[40] A. Meyerson, R. Williams, On the complexity of optimal *k*-anonymity, in: Proceedings of the 22nd ACM PODS, June 2004.
[41] C. Park, K. Itoh, K. Kurosawa, Efficient anonymous channel and all/nothing election scheme, in: EUROCRYPT 93, 1993, pp. 248–259.
[42] European Parliament, Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data, Official Journal of the European Communities, 1995, p. 31.
[43] European Parliament, Directive 97/66/EC of the European Parliament and of the Council of 15 December 1997 concering the processing of personal data and the protection of privacy in the telecommunications sector, Official Journal of the European Communities, 1998, pp. 1–8.
[44] M.K. Reiter, A.D. Rubin, Crowds: anonymity for web transactions, ACM Transactions on Information and System Security 1 (1) (1998) 66–92.
[45] S. Rizvi, J. Haritsa, Maintaining data privacy in association rule mining, in: Proceedings of the 28th VLDB Conference, 2002.
[46] K. Sako, J. Kilian, Receipt-free Mix-type voting schemes – a practical solution to the implementation of a voting booth, in: Proceedings of the EUROCRYPT 95, 1995, pp. 393–403.
[47] P. Samarati, L. Sweeney, Generalizing data to provide anonymity when disclosing information (abstract), in: Proceedings of the 17th PODS, 1998, p. 188.
[48] C. Su, K. Sakurai, Secure computation over distributed databases, IPSJ Journal (2005).
[49] D. Shah, S. Zhong, Two methods for privacy preserving data mining with malicious participants, Information Sciences 177 (23) (2007) 5468–5483.
[50] L. Sweeney, Guaranteeing anonymity when sharing medical data, the datafly system, in: Proceedings of Journal of the American Medical Informatics Association, 1997.
[51] L. Sweeney, Achieving *k*-anonymity privacy protection using generalization and suppression, International Journal of Uncertainity Fuzziness Knowledge-Based Systems 10 (5) (2002) 571–588.
[52] L. Sweeney, *k*-anonymity: a model for protecting privacy, International Journal of Uncertainity Fuzziness Knowledge-Based Systems 10 (5) (2002) 557–570.
[53] Y. Tsiounis, M. Yung, On the security of ElGamal-based encryption, in: Public Key Cryptography'98, 1998, pp. 117–134.
[54] J. Vaidya, C. Clifton, Privacy-preserving *k*-means clustering over vertically partitioned data, in: Proceedings of the Ninth ACM SIGKDD, 2003, pp. 206–215.
[55] J. Vaidya, C. Clifton, Privacy-preserving outlier detection, in: Proceedings of the IEEE ICDM, 2004.
[56] L. von Ahn, A. Bortz, N.J. Hopper, *k*-anonymous message transmission, in: Proceedings of the ACM CCS, 2003, pp. 122–130.
[57] M. Waidner, Unconditional sender and recipient untraceability in spite of active attacks, in: EUROCRYPT'89, 1989, pp. 302–319.
[58] K. Wang, P.S. Yu, S. Chakraborty, Bottom-up generalization: a data mining solution to privacy protection, in: The Fourth ICDM, 2004, pp. 249–256.
[59] A. Williams, K. Barker, Controlling inference: avoiding p-level reduction during analysis, in: Proceedings of the Fifth Australian Symposium on ACSW Frontiers, 2007.
[60] Raymond Chi-Wing Wong, Jiuyong Li, Ada Wai-Chee Fu, Ke Wang, (alpha, *k*)-anonymity: an enhanced *k*-anonymity model for privacy preserving data publishing, in: Proceedings of the ACM SIGKDD, 2006, pp. 754–759.
[61] R.N. Wright, Z. Yang, Privacy-preserving Bayesian network structure computation on distributed heterogeneous data, in: Proceedings of the 10th ACM SIGKDD, 2004, pp. 713–718.
[62] Jian Xu, Wei Wang, Jian Pei, Xiaoyuan Wang, Baile Shi, Ada Wai-Chee Fu, Utility-based anonymization using local recoding, in: Proceedings of the ACM SIGKDD, 2006, pp. 785–790.
[63] Z. Yang, R.N. Wright, Improved privacy-preserving Bayesian network parameter learning on vertically partitioned data, in: Proceedings of the International Workshop on Privacy Data Management, 2005.
[64] Z. Yang, S. Zhong, R.N. Wright, Privacy-preserving classification of customer data without loss of accuracy, in: Proceedings of SIAM International Conference on Data Mining, 2005.
[65] Z. Yang, S. Zhong, R.N. Wright, Anonymity-preserving data collection, in: Proceedings of the 11th ACM SIGKDD, 2005.
[66] Michael M. Yin, Jason T.L. Wang, GeneScout: a data mining system for predicting vertebrate genes in genomic DNA sequences, Information Sciences 163 (1–3) (2004) 201–218.
[67] J.X. Yu, Z. Chong, H. Lu, Z. Zhang, A. Zhou, A false negative approach to mining frequent itemsets from high speed transactional data streams, Information Sciences 176 (16) (2006) 1986–2015.
[68] L. Zhang, B. Zhang, Fuzzy reasoning model under quotient space structure, Information Sciences 176 (4) (2005) 353–364.
[69] S. Zhong, Z. Yang, R.N. Wright, Privacy-enhancing *k*-anonymization of customer data, in: Proceedings of the 24th ACM PODS, 2005.
[70] S. Zhong, Privacy-preserving algorithms for distributed mining of frequent itemsets, Information Sciences 177 (2) (2007) 490–503.
[71] S. Zhong, Z. Yang, Guided perturbation: towards private and accurate mining, VLDB Journal 17 (5) (2008) 1165–1177.
[72] L. Zhang, W. Zhang, Generalization-based privacy-preserving data collection, in: Proceedings of the 10th International Conference on Data Warehousing and Knowledge Discovery, DaWak, 2008.