COMPSCI 725 - Term Paper
An Algorithmic Comparison of Location Privacy k-Anonymity Systems

David MacDonald
dmac127 - 4051614

ABSTRACT

Services which are customised based on a user's location are starting to become common-place at the cost of privacy: it is possible for an attacker to use a series of requests containing location data in order to infer the identity of a user. One solution to this is a k-Anonymity system which takes location data destined for a service (both in terms of time and space) and expands it such that the location now contains k users who all look identical. This protects a user's identity and hence privacy as an attacker is unable to distinguish between any of the k users. We review two location privacy k-Anonymity algorithms and find very different complexities; one claims to be linear in the degree of anonymity, k, while the other is NP-hard. By examining the underlying assumptions and functionality of each system and their algorithm we analyse how such a difference arises and whether this affects the ability of either system to protect a user's location privacy.

1 INTRODUCTION

Issues with location privacy arise when users consent to send their current location to some kind of service in order to obtain information: they may want the service but not to be identified at that location. One example is the ability for a service station company to return the service station closest to a user. The user's mobile phone sends a *Service Request* containing the user's current location (using a device like GPS) to a *Service Provider* which, in this example, is the service station company, who carry out some calculations and returns to the user details about the closest station (terms from [1]). While the identity of a user may not be sent directly to the service provider, by using external sources of information an attacker may be able to infer one. This is a privacy breach not just because the user may not want to be identified at the request location but also because the service provider (or an attacker) may track the user to further locations which are private indeed.

We compare and contrast two different k-Anonymity approaches to protecting location privacy from attackers interested in gaining the personal identity (and locations) of service requesters. The technique used by both systems to obscure requests is k-Anonymity; a *Trusted Server* attempts to make a group of k users look the same such that an attacker is unable to identify any particular user within this group. One approach by Gedik & Liu (GL) [4] uses

a centralised trusted server which 'expands' the service request's location such that a service provider is unable to identify any of the k users within a group of service requests. The other approach by Bettini, Wang & Jajodia (BWJ) [1] also uses a trusted server to expand a request's location but has the advantage of only needing to include k users that could *potentially* make a service request – the reasoning for this difference will become clear soon. We should point out that the word *expand* has important consequences: there is a trade-off between privacy and accuracy which we will discuss in detail.

By examining the two systems we find a very different algorithmic complexity: the k-Anonymity algorithm proposed by GL needs to find a clique of size k in a graph; this is NP-hard [5]. BWJ however propose an algorithm which is linear in k. We investigate the assumptions that lead to these complexities and what the consequences are for both privacy and system scalability.

This paper is arranged as follows: we first discuss how each of the two papers understands privacy and k-Anonymity, this allows us to identify some of the axioms that help the authors to create their systems. Section 3 analyses the two systems and correspondingly the advantages and disadvantages of each. Given these systems we take the assumptions and consequences of the systems' implementation and investigate in section 4 why there is such a dramatic difference in the complexity of the two k-Anonymity algorithms. In section 5 we discuss the overall issues with both of these systems; especially with regards to the trade off between privacy and quality-of-service. Section 6 concludes.

2 PRIVACY AND k-ANONYMITY

Privacy can be defined as the 'ability to control private information, including identity and identifiers, sensitive information such as medical or financial records and information about certain kinds of personal, corporate, or government activity' [2]. While this definition certainly cannot capture all of the complexity involved with privacy, it is broad enough to use as a foundation for our analysis of the privacy understandings of the two systems. Both systems are concerned with protecting *location privacy*, which in the context of this definition means that they provide the *ability to control a user's identity* from being revealed given a set of location-based service requests. The systems want to protect a user's privacy with respect to their personal location-based activities. This is the idea of 'freedom of movement'; free to travel without worrying about location tracking and its consequences [2].

Information which indirectly reveals the identity of a user is called a *Quasi-Identifer* by both papers – details which, in combination with external data-sources can reveal a user's identity. Identifying a user given only their location at night is likely to reveal their place of residence: an attacker may be able to infer their identity by using external information sources to find the residents at that location. This is treated as a privacy breach in terms of

the earlier definition due to a user's identity and corresponding location being compromised – this in itself may be considered *private information* but additionally, given further locations, a user may be identified at locations they wish to remain private. By using these services the user no longer has the *ability* to control the release of their identity.

In terms of location privacy, both systems want to remove the possibility of finding quasi-identifiers given a user's location data – this involves modifying the location and time of a request such that an attacker cannot easily combine any other information in order to determine the identity of a user. BWJ extend this by defining a *Location Based Quasi-Identifier* (LBQID) which is effectively a sequence of times and locations that the user believes could identify them. We can think of a LBQID as the 'movement pattern' which uniquely identifies this user. As an example, a user may be identified by observing that they travel from their home to the University in the morning and back each evening for every week day; their LBQID would be constructed by including their movements (and times) to and from the University over the period of a week. BWJ claim that, in not being able to determine the LBQID of a user, the user's location privacy will be maintained.

For our purposes when we specify *Location Data* we include both the spatial and temporal location of a user: it is useful to conceptualise this as a 3D space where the x, y and time, t, vectors are orthogonal – we denote this as the *Location Space*. A request with location data received from a user contains an exact x, y and time coordinate in this space: (x, y, t), while an *Expanded Area* that a service provider receives can be thought of as a 3D box: $((x_1, x_2), (y_1, y_2), (t_1, t_2))$ where the two points form a boundary in each dimension.

Given this problem of protecting location-based privacy, both papers propose solutions based on k-Anonymity. This means that an attacker should not be able to distinguish between at least k users for a given request: the request that a service provider receives will match at least $k - 1$ users other than the requester. In general, this amounts to making the request location cover an area which includes k or more users i.e. creating an expanded area in the location space containing k users. Referring again to the privacy definition we see that these systems attempt to give users the *ability* to protect their privacy to a *certain extent* – the system can not ensure true privacy as an attacker still has a $1/k$ chance of correctly guessing a user's identity. This is an important point which illustrates the trade-off users must be willing to make in order to use location-based services: neither paper proposes a system which *truly* protects a user's privacy – only in the trivial case of not sending any location data at all is this possible. This kind of trade-off is preferred in the real world as the ability to disclose 'selected information to selected parties' provides users with greater flexibility in the services they can use while still maintaining some degree of privacy [2]. By using k-Anonymity, both systems do however offer the possibility of protecting a user's privacy to an extent which users are comfortable with: a larger k provides a smaller chance that a user can be identified. The

cost however of forming a larger group of identical-looking users is accuracy – intuitively the higher the k, the larger the expanded area is likely to be in order to include this many users.

There is an interesting conflict between these two papers on the ease with which a quasi-identifier can be found: BWJ argue that in an earlier paper, the assumption of GL that every request containing a location is a quasi-identifier and hence a possible privacy breach [3] is too strong. The reasoning behind their argument is that every request location could only act as a quasi-identifier if it is also assumed that an external source such as a camera is always available to identify the k users within an expanded area. The current version published by GL still makes this assumption. This leads to quite different requirements for the two systems in order to enforce k-Anonymity. In the GL system, a service provider that receives a set of requests containing location data should find that for any request, there are k requests from *distinct users* which will have the same expanded area. On the other hand, BWJ only require that a request's location data is subject to k-Anonymity if it matches some part of the user's LBQID (using the example above; a request from home in the mornings is one element of the LBQID and hence will need to be anonymised) – a request which does not contain any elements of the LBQID can be sent without modification as an attacker will not have any quasi-identifier in the location data which they can use to determine a user's identity.

In order to enforce k-Anonymity, GL requires that there are k requests from different users which all get anonymised together. BWJ are more flexible in their use of k-Anonymity as it includes users who were in the approximate location when the user made the request and hence *could* have made a request. Intuitively we can see that it is easier to find a group of users in an area that could send a request containing location data rather than those who *actually* send a request as in the GL system: this is likely to result in more fine-grained results.

Using these ideas we can get an idea how k-Anonymity works according to BWJ; if the first element of a LBQID for the user is encountered in a request then the request location is expanded such that there is $k-1$ users in the location who could have made such a request. For any following request which matches other elements of this user's LBQID, the expanded area is made to include both the user and the same $k-1$ users. This means that even if a service provider receives requests which correspond to all elements of a LBQID for a user and hence would be able to identify the user, they would find that there are $k-1$ other users who followed the same 'movement pattern' and hence *could* have made the same series of requests.

3 SYSTEM DESIGN

The systems proposed by both BWJ and GL rely on the same infrastructure, although they name the components differently. We use the BWJ terminology and as already described, a mobile user wants to make a *Service Request*; this contains their exact location data. The user

sends this service request to a *Trusted Server* (named *Trusted Anonymity Server* by GL) which carries out the k-Anonymity algorithm (discussed later) in order to protect the user's privacy. The trusted server then passes this modified request to the *Service Provider* (named *Location-Based Services* by GL) which carries out the relevant computation; returning the result to the trusted server who then passes it back to the user. We see that the trusted server effectively acts as a proxy between the user and service provider. It will become clear that even if the overall infrastructure for both systems is identical, the underlying implementation details are very different.

The word 'trusted' in 'trusted server' is an important part of the system. Trust, briefly speaking, means that a user relies on a system and this trust is binary, that is, we trust something or we do not [7]. The reliance here is key as users rely on the trusted server to use their personal details in order to protect their anonymity: they believe that it will act in their best interests and also protect this information. We note that there are different types of trust in these systems and this trust is there or not (binary): a user may trust a trusted server with their request location but not with their location at any given time. The types of trust that users are willing to place in a 'trusted server' will have important consequences for each system.

Both systems are flexible in that they allow users to customise the k value and hence the level of anonymity they think will be suitable given the service provider they are attempting to use. This is especially useful as it means the system can handle users that have different privacy evaluations: users are not dependent on a fixed k value for a given service provider. If a user believes there is little risk with using a particular service provider they can set a smaller k value and are therefore likely to obtain a more accurate service. The contrapositive applies to service providers whom users do not trust with an increased possibility of identifying them. A user can avoid any privacy adjustments by setting $k = 1$ and therefore have the service request sent to the service provider as-is since the expanded area does not need to be enlarged in order to include any other users.

As already described, both system's trusted server use an algorithm which acts as a location 'expander': creating an expanded area from users' request location data such that a service provider is unable to distinguish between any k users. This creates a trade off with quality-of-service; users might want to impose some constraints on the inaccuracy of the request that the service provider receives: locating the nearest fast food restaurant is likely to require far finer resolution than a request for local weather. The advantage with the GL system is that the user can specify in a request the spatial and temporal tolerances (along with a k value) meaning they can limit the inaccuracy that the trusted server can impose both in terms of time and location. GL denote this as the *Tolerance Constraints* for a request and we can imagine this as a 3D box centred about the request location data which specifies the largest expanded

area in the location space that the user is willing to accept. The BWJ paper mentions that they have an additional parameter 'tolerance constraints' which is determined based on the particular service that is provided: the previous example shows how this is required in order to make certain services useful. It appears that this is set by service providers but BWJ do not make this clear.

Tolerance constraints mean that there is a possibility of failure and each system handles this differently. In the case of GL, a request sits in a queue on the trusted server until k-Anonymity can be achieved within the tolerance constraints specified by the user. If the user's temporal tolerances are exceeded (the request 'times-out') then it will be dropped from the queue and classified as a failure. We presume that the user is informed of this failure although this is not made explicit by the authors. BWJ try to be more supportive in their handling of failure – if the tolerance constraints fail because the expanded area is too large then the trusted server will progressively reduce its size until the tolerance constraints are met. Consequentially this means that k-Anonymity is not assured; the trusted server provides users with the choice of continuing at their own risk using the adjusted location, or trying again later when there is a possibility of more data becoming available which ensures k-Anonymity.

The system proposed by BWJ requires that 'a location update may be received by the TS [Trusted Server] even if the user did not make a request when being at that location', this is because their system relies on having at least $k-1$ users that could *potentially* send requests rather than finding $k-1$ other users requests to manipulate as is done by the GL trusted server. BWJ see this as an advantage because the set of users that actually send a request is at least as large as the set of users that could potentially send a request: it is likely that the size of the expanded area is smaller (and more accurate). BWJ calls this a *Personal History of Locations* which is effectively a list stored on the trusted server indicating the places a user has visited and at what time. The consequences of this are not discussed by the authors – users have to hold different types of trust for the server: they must trust the server with their current location as well as a unique location tracking identifier: their LBQID. BWJ are assuming that users will automatically submit to this kind of trust although it is unknown whether users are willing to have their location constantly monitored by a 'trusted' server in order to maintain privacy with several untrusted service providers. In the GL system since only unprocessed requests need to be stored, users need only show trust to the extent of their current request and user details. While both systems have a 'trusted server' the types of trust that users have to hold for each type of server are very different.

A major distinction between the two systems is the assumption of the capabilities of the service providers. GL make the explicit assumption that the service providers are at least 'semi-trusted', that is, while they may try to determine to identity a user based on service requests they will not actively attack the system. Having an attacker attempt to inject

'dummy' requests in order to identify a user is not considered. BWJ indicate that location data could be released to an 'untrusted' service provider, however since they make no further mention of defences against active attacks we assume that this assumption is implicit. This is because a malicious service provider could introduce users into the system; knowing the location of these fake users makes identifying actual users easier: the system goes from providing k-Anonymity to some value less than k depending on the number of fake users that were added. In addition, since BWJ appear to assume that each service provider has its own tolerance constraints, unless these are checked by the trusted server we will have to assume that the service provider will choose reasonable constraints based on the service it is providing. A malicious service provider could, after all, choose very small values in an attempt to force k-Anonymity failure and hence frustrate the user into disabling privacy settings on their device in order to obtain a reasonable service.

4 ALGORITHM ANALYSIS

4.1 The Gedik & Liu (GL) Algorithm

The GL trusted server uses a k-Anonymity algorithm that finds an expanded area for a set of requests based on an undirected graph – the nodes in this graph are represented by the set of unprocessed requests that have been sent to the trusted server. Two nodes (requests) are connected if the requests belong to different users and the request's location data exists within the other's tolerance constraints. When a request is received it is added to the graph and the algorithm tries to find a clique of size k (specified in the request) which includes this 'base' request. Since each request has its own personalised k value, the algorithm can only include other requests in this clique if their k is no larger than the k of the 'base' request. If a clique is found then the expanded area for these k requests is set to an area which contains the location data of each request: all k requests will look identical in terms of location data. In addition, they meet the anonymity requirements and tolerance constraints for each user as requests are only connected in the graph if they are close enough in the location space as to not exceed each other's tolerance constraints. The k requests are then removed from the graph, randomised in order (ensuring the service provider is unable to gain any information given the order in which the requests were received) before being sent to the service provider.

However finding a clique in a graph of size k is NP-hard [5]. Since there was no specific mention of the exact complexity of this algorithm in the paper other than 'we would like to note that the general problem of the optimal k-anonymization is shown to be NP-Hard' we attempt to form our own complexity estimate. Since the clique search algorithm is called when a new request arrives r_s (i.e. the algorithm searches for a clique which contains this request) we can use this as our 'base', searching iteratively from this point. If we imagine the brute force approach of binary patterns with $k = 3$: $(0,0,0), (0,0,1), \ldots, (1,1,1)$, we need only be

concerned with the combinations where the node that we start with is 1 (included); we have half the number of combinations: $O(2^{k-1}) = O(2^k)$ which, as expected, is exponential.

The authors recognised that this complexity will affect the scalability of the system and devised several heuristics in order to improve performance. One method involves observing that, in starting with the most recently received request, the clique would only be at most the size of its k value. A modified algorithm instead searches all neighbours of the received request and attempts to find the largest possible clique that includes this request. This allows the trusted server to forward a larger batch of requests to a service provider at once; users may not have to wait as long for a response. We do not discuss the other heuristics due to space constraints.

4.2 The Bettini, Wang & Jajodia (BWJ) Algorithm

The BWJ k-Anonymity algorithm only gets executed by the trusted server if the request location data contains an element of the user's LBQID: other requests can be sent directly to the service provider. If the trusted server encounters a request which matches the first element of a user's LBQID then it needs to find the smallest expanded area which contains the request location data and the $k-1$ closest users who can *potentially* make a request (based on their personal history of locations). These $k-1$ user identifiers are stored for future use. In the case where the trusted server encounters a request which contains any successive element of the user's LBQID then the algorithm takes the $k-1$ users that were found for the first element of the LBQID and finds the smallest expanded area containing these users. If a service provider receives a series of requests containing expanded areas enclosing all elements of the user's LBQID then there will be $k-1$ other users within the expanded area for each element of the LBQID that *could* have sent the same pattern of requests. The authors believe these steps can be carried out in $O(k * n)$ time where n is the number of location points currently stored in the trusted server: the algorithm finds the closest point in the personal history of locations for k users which may involve iterating over the entire location space of size n a total of k times. We note that this is effectively linear complexity in terms of k.

Using the same $k-1$ users for each element of the user's LBQID raises an interesting efficiency condition for this algorithm if, for example, when identifying the first $k-1$ users the algorithm finds them all clustered closely together giving a relatively small expanded area. Since for the following matches of the user LBQID the algorithm uses the same $k-1$ users for finding an expanded area there may be a worst case scenario of all users diverging in opposite directions (like a sphere in the location space) which would consequentially lead to a large and possible sub-optimal expanded area. This problem will get worse as k or the length of the LBQID increases. One alternative is that instead of a greedy approach of taking the closest $k-1$ users at the start, the algorithm could take users which are further away but converge

as elements in the LBQID are discovered for a user's requests: this may provide a series of more accurate requests in the long-run. The authors do not mention such issues and we can hypothesise that adding such features would have a significant effect on the overall complexity of their algorithm.

A potential error we found with the algorithm as presented by BWJ is the input parameters: the paper states as an input: 'k user-ids (if r matches the initial element of an LBQID) or a parameter k'. This statement, however, should state: 'k user-ids should be provided (if r does **not** match the initial element of an LBQID) or a parameter k'. The reasoning for this is that they state in their algorithm description that k users are discovered for the first element in the LBQID and these k users are used for all consequential elements. Hence we see that the first time an element of a LBQID is located, it creates a list of k users which are then used as an input of the algorithm the next time it is called for further elements in the LBQID.

4.3 Complexity Issues

Now we begin to get an idea as to why, despite having two systems which provide location k-anonymity, there is such a dramatic difference in complexity. The algorithm proposed by BWJ is concerned only with finding k users that are in approximately the same area as the user request data. By keeping a personal history of locations it is possible to process a request straight away without concern as to the requests made by other users. Conversely, by working on a per-request basis, the GL algorithm must find k requests at a given instance which can be made to look the same with the additional requirement that all requests meet their tolerance and anonymity constraints. Since no request history is kept, anonymity must be re-calculated every time a request is received.

A paper by Meyseron & Williams (MW) proves that the problem of k-Anonymity is NP-hard in general [6]: this therefore matches what was found with the interpretation of k-Anonymity that GL use. The MW proof however is focussed on anonymising databases: they define a *suppressor* function, t which takes a set of records and omits certain fields. k-Anonymity is therefore defined by stating that for any record v_1 in a database which is anonymised, there exist $k-1$ other records which, when anonymised using the function t, would be equivalent i.e. $t(v_1) = t(v_2) = \ldots = t(v_k)$. The suppressor function is equivalent to the GL location k-Anonymity algorithm where instead of 'omitted' values, the algorithm expands the area of k requests to be the same. The GL algorithm which finds a clique can be treated as t and for requests $r_1, \ldots, r_k$ we find that in computing $t(r_1)$, the expanded area for r_1 is the same as $r_2, \ldots, r_k$ and hence: $t(r_1) = t(r_2) = \ldots = t(r_k)$. Further research could be conducted into extending the MW proof to show that the same NP-hard result applies to location k-Anonymity; we conjecture that this is true and therefore the GL system is ultimately bound by this complexity: there is no general polynomial time algorithm (unless $P = NP$).

In a basic simulation the GL system indicated that $k < 12$ was computable in a reasonable time, however anything larger was intractable – the system is ultimately limited by this. A practicality investigation would need to be conducted in order to see whether this is actually a problem or people are willing to use a k value of less than 12.

Since the *suppressor* function for BWJ does meet the same definition as MW we cannot claim that it has the same complexity. The BWJ algorithm expands the location from a request to look like it was sent by $k-1$ other users, rather than taking k requests (or database entries) and making them all look the same as proposed by GL. It is not possible to say that the output of the BWJ suppressor function t for a particular request r_1 is the same as another request, r_2, from a different user even if they are in the same location k-Anonymity 'group' used to protect identity – it may not be the case that $t(r_1) = t(r_2)$. By interpreting k-Anonymity differently BWJ have effectively created an algorithm with reasonable complexity: the same complexity limitation for k does not apply.

5 SYSTEM CONSEQUENCES

Given the two systems it appears that BWJ have created a system which provides a better trade-off between privacy and quality-of-service than GL. This is because only requests which compromise the identity of a user are anonymised (based on the LBQID): GL assume that every request can potentially compromise identity and hence all requests are anonymised, sacrificing quality-of-service. In addition, since the BWJ trusted server periodically receives location data from users, the algorithm has more information with which to find k users that can look 'the same', providing a more defined location request which a service provider can use to return a more accurate answer. The GL algorithm however only has unprocessed requests to work with, finding k users will be harder and hence is likely to lead to more undefined responses. This quality-of-service metric includes not just spatial dimensions, but also a temporal dimension, we see that the GL system may have to wait until a group of requests from k users can be anonymised and this takes time: testing the patience of some users. In saying this, BWJ are quiet on is how a group of k 'diverging' users discussed earlier may affect quality-of-service.

But these advantages of the BWJ system come at a cost. The privacy of a user in the BWJ system rests on the correct derivation of the LBQID – missing an element means that a user could be identified with ease because the service provider will receive their exact position. It is unfortunate that BWJ are quiet on the subject of LBQID derivation as such identifiers are critical to the success of their system; without reference to other research that has done analysis into this derivation we have our doubts as to the practicality of this system. There are many possible 'external sources' which could be used to identify a user: is it possible create an anonymised statistical method which accounts for all possible sources of external information for a range of users? While the GL approach is more conservative in considering

each location a possible privacy breach, it at least provides a guarantee of location privacy given the capabilities of k-Anonymity.

Interestingly BWJ explain that 'sensitive data' in the context of their system includes details such as medical and financial information which may be part of a service request and hence privacy should be preserved: an attacker should still not be able to infer the identity of a user and connect these sensitive details. Difficulty arises when BWJ are only concerned with protecting *location* privacy; while trying to protect the association of 'sensitive information' including personal data with a user, they are not concerned with whether this additional data is a quasi-identifier itself. We believe financial and medical information could easily be used to identify a user. Since GL are concerned with sending only location information they need not be concerned with any further potential quasi-identifiers.

6 CONCLUSION

As the use of location services becomes more common, it is likely that users are going to become concerned about protecting their location privacy. The two systems we presented, one by Bettini, Wang & Jojodia and the other by Gedik & Liu, attempt to find ways to ensure users can protect their privacy to a *certain extent* while still enjoying requests being personalised to their locale. By using k-Anonymity both systems provide the *ability* for users to protect their identity: users can trade off service accuracy in return for greater privacy.

However both of these systems pursue this objective very differently; we have a competition between two systems that can protect users' location privacy. The algorithmic complexities indicates just how different these systems are: having the GL algorithm being NP-hard and the BWJ algorithm linear in complexity for k shows the effect that each systems assumptions have made. It is unclear whether users are willing to be bound by the practical limit of k in the GL system or prefer the wider range of k provided by BWJ. While the GL system may be capable of enforcing k-Anonymity in all situations, its corresponding inaccuracy and the times required for a response may give BWJ an advantage in this respect. However the difficulties with BWJ may well be equally as concerning for users – it is unclear whether users will be willing to have their location monitored in addition to having their LBQID calculated when this information may be considered private beyond the extent that the user is willing to trust it with the trusted server. What types of trust are users willing to show for a 'trusted' server? While both systems are a sophisticated contribution to the field of location k-Anonymity, further work is needed to determine not just any issues with algorithmic scalability, but also the practicalities of deploying this to a large population of mobile users.

References

[1] Claudio Bettini, X. Sean Wang, and Sushil Jajodia. Protecting Privacy Against Location-Based Personal Identification. In *Proc. of the 2nd VLDB Workshop on Secure Data Management*, pages 185–199. Springer-Verlag, 2005.

[2] Mike Burmester, Yvo Desmedt, Rebecca N. Wright, and Alec Yasinsac. Accountable Privacy. In *Security Protocols*, volume 3957 of *Lecture Notes in Computer Science*, pages 83–95. Springer Berlin / Heidelberg, 2006.

[3] Bugra Gedik and Ling Liu. A Customizable k-Anonymity Model for Protecting Location Privacy. In *In ICDCS*, pages 620–629, 2004.

[4] Bugra Gedik and Ling Liu. Protecting Location Privacy with Personalized k-Anonymity: Architecture and Algorithms. *IEEE Transactions on Mobile Computing*, 2007, 2007.

[5] R. M. Karp. Reducibility Among Combinatorial Problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[6] Adam Meyerson and Ryan Williams. On the Complexity of Optimal k-Anonymity. In *PODS '04: Proceedings of the Twenty-Third ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 223–228, New York, NY, USA, 2004. ACM.

[7] Mark Stamp. *Information Security: Principles and Practice*. Wiley Publishing Inc., 1st edition, 2005. pp. 328.