VENETA: Serverless Friend-of-Friend Detection in Mobile Social Networking

Marco von Arb *, Matthias Bader †, Michael Kuhn ‡ and Roger Wattenhofer §
* Computer Engineering and Networks Laboratory, ETH Zurich, Switzerland, varb@ee.ethz.ch
† Computer Engineering and Networks Laboratory, ETH Zurich, Switzerland, baderma@ee.ethz.ch
‡ Computer Engineering and Networks Laboratory, ETH Zurich, Switzerland, kuhnmi@tik.ee.ethz.ch
§ Computer Engineering and Networks Laboratory, ETH Zurich, Switzerland, wattenhofer@tik.ee.ethz.ch

***Abstract*—Recently, mobile social software has become an active area of research and development. A multitude of systems have been proposed over the past years that try to follow the success of their Internet bound equivalents. Many mobile solutions try to augment the functionality of existing platforms with location awareness. The price for mobility, however, is typically either the lack of the popular friendship exploration features or the costs involved to access a central server required for this functionality. In this paper, we try to address this issue by introducing a decentralized method that is able to explore the social neighborhood of a user by detecting friends of friends. Rather than only exploiting information about the users of the system, the method relies on real friends, and adequately addresses the arising privacy issues. Moreover, we present VENETA, a mobile social networking platform which, among other features, implements our novel friend of friend detection algorithm.**

***Index Terms*—P3, Social Networking, Friend-of-Friend, Friend Detection, Mobile, Decentralized, Privacy, Security**

I. Introduction

Internet based social networking services have experienced an enormous growth over the past years. Popular social networking services, such as *MySpace* or *Facebook*, have gained tens of millions of users in less than 10 years. Sales figures for these companies today by far exceed the one billion dollar mark. Simultaneously to the social networking services' triumphant advance, mobile devices in general and smart phones in particular have rapidly penetrated the consumer market. Recent phones do not only exhibit considerable computing resources but also feature means for Internet access as well as wireless short distance communication such as Bluetooth and Wi-Fi. They seem to be the perfect platform to combine the market potential of traditional social networking services and the success story of mobile devices. Even though several attempts that have tried to merge the two worlds could not reach the masses, experts expect that future mobile social networking systems possibly even exceed the success of their Internet bound counterparts.

There exist several differences between the traditional and the mobile environment. Complex (and expensive) billing models in the mobile context ask for short connection times and low data-volumes—requirements that do not exist in the flat rate dominated world of landline Internet connection. These monetary obstacles, together with the restricted input and output capabilities prevent the implementation of many of the currently successful features such as multimedia sharing or blogging on mobile devices. However, these devices exhibit other characteristics that are of advantage in mobile social networking. We believe that two key features are the user's permanent reachability and location awareness. The location aspect is well reflected by the buzzword *P3 (people-to-people-to-geographic-place)* that recently emerged in the field.

While popular social networking systems differ in many aspects, they still all build—as their name suggests—around one common component: The possibility to explore a user's social network. Obviously, people like to see who their friends' friends are, and they also like to get in contact with them, as the high clustering coefficients[1] of social networks indicate. So far, mobile solutions either heavily rely on costly Internet access to implement such features, or, to fully preserve the advantages of the mobile environment, entirely omit them.

The work presented in this paper tries to bridge this gap. In particular, we show how the exploration of a user's social neighborhood can smoothly be integrated into a mobile system without the need of costly Internet

[1]The clustering coefficient measures the probability that two friends of a person are friends themselves.

access. To reduce the bootstrapping issues commonly known to social networking systems, our method bases on existing "real world contacts". It is able to detect friends of friends that are in the user's current physical proximity and thus well fits the P3 paradigm. Moreover, our system respects the user's privacy. As indicated before, it works in a purely decentralized way and thus overcomes the previously mentioned monetary issues. It is well suited for use with mobile phones featuring any short distance wireless communication technologies. We have implemented a mobile social networking platform, *VENETA*, that combines our neighborhood exploration method with other features. The platform is based on Java Microedition (J2ME) and is designed for Bluetooth enabled mobile phones. It builds around server-independent core features, namely decentralized friend finding and ad-hoc multi-hop messaging. The latter feature might provide a cheap alternative to SMS in "quasi-static" scenarios, such as in auditoria, sport-stadiums or at a conference. To further reduce bootstrapping problems, a server component has been added that can map location information from JSR-179 compliant devices and that supports some additional chatting functionality.

In the remainder of this paper, we will first explain the decentralized friend-of-friend finding idea in more detail (Section II). We will then discuss the privacy concerns related to this idea and present a cryptography based solution that overcomes these concerns (Section III). Finally, we show how we have integrated the decentralized friend finding component into the mobile social networking system *VENETA* (Section IV). The paper is rounded off by a brief discussion of relevant related work and some concluding remarks.

II. Friend-of-Friend Detection

Friends of friends play an important role in society. They are not only major actors in gossip and stories, but also proof extremely important in many other contexts. People are hired because their friends tell them about a friend who has a particular job opening. Headhunters of large companies nowadays try to take advantage of this fact. The friend of friend job recommendation idea is also the major feature of the social networking platform *LinkedIn*. Friends of friends are, however, not only important in the job market, but also in other businesses, such as the real estate market, or products recommendations. The high clustering coefficients of social networks further indicate that friends of friends are an important ingredient of our social and emotional surrounding and that they are likely to become our (direct) friends. We believe that a common friend at least as well indicates a potential "match" between two persons as typical profile information of matchmaking applications, such as common interests or character traits, does.

Interestingly enough, we all implicitly carry this extremely valuable information around, but do not systematically take advantage of it: Whenever two strangers meet in the street, all they have to do to figure out whether they are friends of a (common) friend, is to compare their mobile phones' contact books. This is exactly how our decentralized friend of friend detection works. Whenever two mobile phones come into Bluetooth[2] connection range, they compare their contact book entries. If none of the users appears in the other's contact book (i.e. the users are friends already) and they share at least one common contact, then the two users are identified as friends of a friend. Observe that such a comparison can easily be done, as the phone numbers (and similarly e-mail addresses, etc.) act as globally unique identifiers. This basic idea is illustrated in Figure 1. Surely, some contacts, such as the provider's help desk number, the doctor, or 911, do not reflect real friends and have to be deselected prior to the comparison. Moreover, due to different prefix formats (e.g. `+1` vs. `0011-1` vs. `001`), only the last 7 digits of the phone numbers are compared.[3]

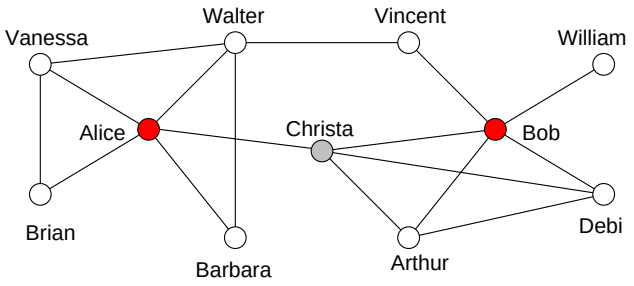
Unfortunately, this idea has a relevant snag: Hardly anybody wants to broadcast his/her contact book to everybody. One might think that this problem can easily be solved by simple cryptographic hash-functions, i.e., by sending and comparing hash values instead of the actual phone numbers. However, an adversary could simply construct a lookup table containing the hashes to all the possible 10^7 phone numbers. Therefore, a more sophisticated solution is required. The next section discusses how the decentralized friend of friend finding idea can be realized in a privacy preserving way.

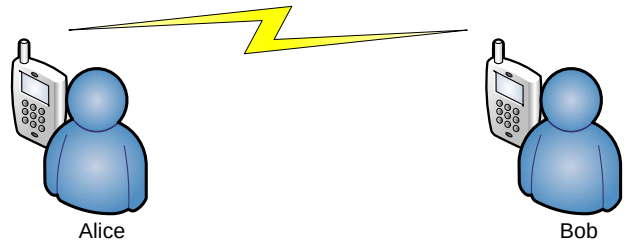
III. Preserving Privacy

How can two persons compare their address books without revealing the contacts (except for those that match)? More formally, two parties each own a set and they want to find the intersection of these sets without revealing the own set to the other party. This problem is known as *secure two party set intersection*,

[2]Any other short distance wireless connectivity technology would work as well.

[3]Note that the probability that two persons meet and own a contact with (semantically) different prefixes, but coinciding "base number" (i.e. the last 7 digits) is negligible.





Barbara Smith	1246098
Brian Lewis	6583404
Christa Doe	4320438
...	...
Vanessa Platt	7280422
Walter Boyd	8750557

Arthur V.	7092139
Christa D.	4320438
Debi B.	4457622
...	...
Vincent M.	7764982
William M.	5833210

Figure 1. Detecting friends of friends: Whenever the mobile phones of two strangers come into connection range, they can compare their users' contact books. If a matching entry is found, the two strangers are informed to be friends of friends. A possible social network that reflects the information from the contact books is illustrated in the top.

and belongs into the cryptographic area of secure multi-party computation. The goal of secure multi-party computation is to evaluate a function (or algorithm) that takes an input value of each participating party. At the end of the protocol, each participant should know the result. However, none of the participants should know more about the other participants' input values than what can be derived from the result and the own input value. Throughout this section, we will make use of the following notations:

- *Encryption*: We will denote the encryption of a message m with key κ as $E_\kappa(m)$.
- *Commutative Encryption*: A commutative encryption scheme is invariant to the order in which encryption and decryption functions are applied. In particular, the following holds: $E_\alpha(E_\beta(m)) = E_\beta(E_\alpha(m))$.
- *Homomorphic Encryption*: A homomorphic encryption scheme allows to calculate $E_\kappa(m_1+m_2)$ given $E_\kappa(m_1)$ and $E_\kappa(m_2)$.
- *Passive Adversary*: Passive adversaries try to find out something about the others' input values, but strictly follow the protocol. They are also called *semi-honest*, or *honest but curious adversaries*.
- *Active Adversary*: As opposed to a passive adversaries, active (or *malicious*) adversaries do *not* necessarily follow the protocol. They can do whatever they want to compromise the other party's privacy.

It is known that any $0-1$ valued function can be evaluated in this model under the assumption of *passive adversaries* [9]. Unfortunately, the generic method this result is based on is computationally too expensive for most real-world problems. Moreover, real adversaries might deviate from the protocol, such that specialized solutions are required.

There are two major approaches that reduce the computation and communication complexity of two party set intersection protocols compared to generic solutions: Either an algorithm bases on a commutative encryption scheme or it exploits the power of homomorphic encryption in combination with polynomials. Both alternatives exhibit linear communication complexity. The basic construction of the second approach, as first proposed by Freedman et al. [3], is vulnerable to a simple attack in the presence of active (or malicious) adversaries. The problem has been recognized and addressed by the original paper as well as succeeding variants [3], [4], [7]. All of these fixes, however, make the protocol rather complicated. Therefore, we decided to rely on a construction based on commutative encryption. Huberman et al. [5] have proposed the following basic variant. Alice (A) owns a set $X = x_1, ..., x_N \subset V$, and Bob ($B$) owns a set $Y = y_1, ..., y_M \subset V$. In our case, X and Y correspond to the phone numbers in Alice's and Bob's contact books, respectively, and V is the set of all 10^7 possible phone numbers. The following protocol allows to find $X \cap Y$:

1) $A \rightarrow B$: $E_\alpha(x_1), ..., E_\alpha(x_N)$ (α randomly chosen)
2) $B \rightarrow A$: $E_\beta(y_1), ..., E_\beta(y_M)$ (β randomly chosen)
3) $A \rightarrow B$: $E_\alpha(E_\beta(y_1)), ..., E_\alpha(E_\beta(y_M))$
4) $B \rightarrow A$: $E_\beta(E_\alpha(x_1)), ..., E_\beta(E_\alpha(x_N))$
5) Both, A and B, can compare the lists from step 3 and 4. Due to the commutativity assumption, if $x_i = y_j$ then $E_\beta(E_\alpha(x_i)) = E_\alpha(E_\beta(y_j))$. Moreover, both parties know the original elements

(and their order) in one of the lists, such that they can derive the matching elements (phone numbers, in our case).

Observe that this protocol does not only reveal the set intersection, but also the size of the input sets (which is not critical in our context). Agrawal et al. [1] provide a detailed analysis of the protocol. They show that, given the decisional Diffie-Hellman hypothesis (DDH) holds, the following encryption function satisfies the requirements of the protocol:

- $E_\kappa(m) = h(m)^\kappa \mod p$.
- p is strong prime, i.e. $p = 2q + 1$, with p, q prime. Clearly, p has to be large enough, such that the discrete logarithm problem is hard.
- Dom $\mathcal{E}$ consists of all quadratic residues mod p.
- $\kappa \in 1, 2, ..., q - 1$.

Agrawal et al. further assume that there exists an ideal hash function $h : V \rightarrow$ Dom $\mathcal{E}$ that maps each element $v \in V$ to a perfectly random element $d \in$ Dom $\mathcal{E}$. In our implementation, $h(\cdot)$ is a "normal" cryptographic hash function. Figure 2 illustrates the complete protocol for our example.

The protocol has further been analyzed with respect to malicious adversaries. Zhang and Zhao [10] as well as Li et al. [8] state two major problems of the protocol under this adversary model:

- Simply changing the input set, such as extending it by a few elements, can compromise the other party's privacy. We believe that this is not a relevant issue in our case. If somebody can extend the input set, he/she could clearly have had the corresponding contact in the contact book. Since the input set sizes are revealed, simply sending all 10^7 possible phone numbers is also not possible. In fact, for performance as well as security reasons, we restrict the input sets to contain a maximum of 300 entries.
- The protocol is not symmetric. B could decide to skip step 4 of the protocol, such that A does not know the matches, whereas B does. Zhang and Zhao refer to a cryptographic primitive that allows to "simultaneously" exchange values. Due to the additional complexity and the related performance issues, we have not implemented this idea. We assume that the attack cannot cause any serious damage. A victim only reveals a contact he was willing to share, without getting this information in return. Moreover, observe that such an attack is very unlikely to happen: If Alice wants to compromise Bob's privacy, she has to guess a contact she *might*[4] have in common with Bob. To be of any relevance, this contact would further require to be of delicate nature. Most likely, however, Bob would not have made such a delicate contact available to *VENETA*, such that Alice's chance of success is virtually zero.

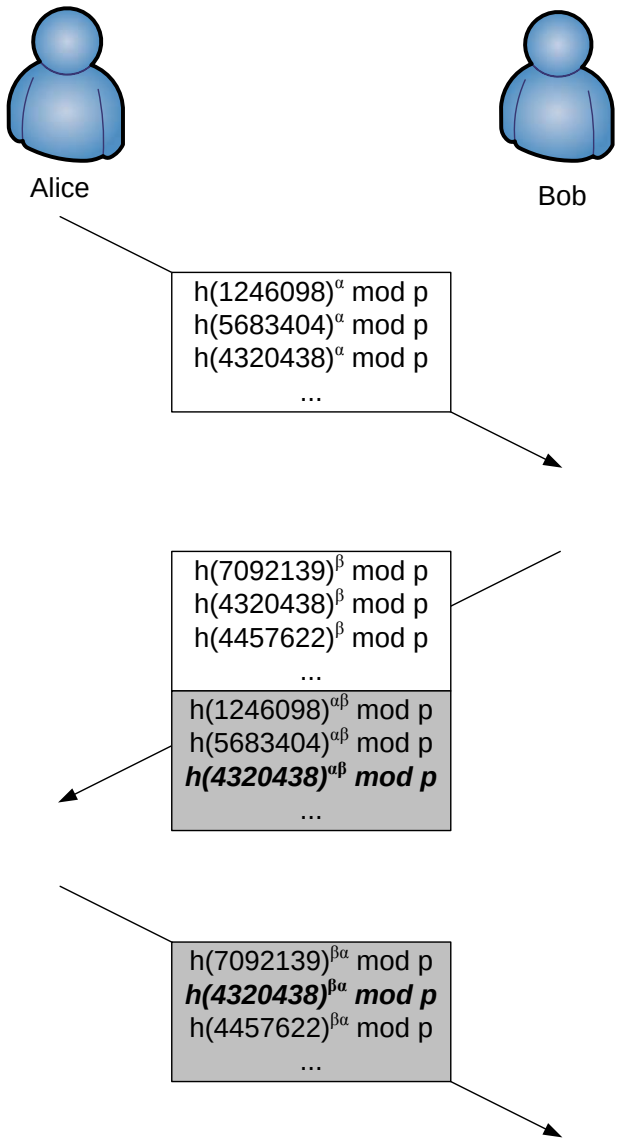


Figure 2. The complete protocol for our example phone numbers (recall Figure 1). After the last step of the protocol, both, Alice and Bob, know the shaded lists. After comparing the lists, they find that there is a matching entry, namely Christa's phone number (4320438).

Zhang and Zhao finally introduce an intent based adversary model and a utility function that measures the trade-off between privacy disclosure and correctness of the result. They show that the basic protocol is secure

[4]Only if she is not sure, she can gain information.

if there is a mutual interest for the information sharing to succeed, and the participants are rational (in a game theoretic sense), even if adversaries do not follow the protocol.

IV. VENETA

A system that solely implements the contact matching idea exhibits a major bootstrap problem: If only few people use the system, the meeting probability of two people possessing an identical contact is very low. Consequently, the contact matching functionality has to be embedded into a more comprehensive system. We have therefore developed *VENETA*[5] (available at `www.veneta-project.net`), a mobile social networking platform that addresses the outlined bootstrapping problem from two sides.

On one hand, additional infrastructureless functionality has been integrated, which decouples the system from any network provider and is thus free of charge. On the other hand, we did not want to sacrifice the possibilities of server bound features. Despite of the incurred costs, we have thus implemented a central server which is able to provide benefits even for users that are not in each others immediate proximity. As this functionality is dependent on the aforementioned provider dependent billing models, all server bound features are optional, and care has been taken to keep data volumes low.

The resulting platform exhibits the following features:

- *Contact matching:* (see Section II).
- *Profile matching:* Besides contact matching, a traditional user profile based matching has been implemented. A search mode only considering age and gender (including a wildcard) ensures a chance of success even for low user density. Such a trivial matching scheme is particularly important in the bootstrapping phase. We are convinced that the success of the Japanese system *Lovegety* [6] (see Section V) was only due to this simplicity.
- *Decentralized messaging:* Often, SMS-messages are sent over short distances (e.g. at a sport event, to locate the office mate on the same floor, in an auditorium, at school, etc.). We have implemented a simple Bluetooth based messaging service that delivers messages up to 3 hops using epidemic routing. We believe that this feature is particularly appealing to young users with limited budget.
- *Server bound messaging:* Decentralized messaging is smoothly extended by centralized messaging. To keep in touch with friends even if they are not currently close-by, messages can also be delivered by the server. For privacy reasons, all data traffic is encrypted. The server acts as an intermediary to bind public keys to phone numbers (which are verified using SMS) and to distribute these keys when required.
- *User locating tracking:* For devices that implement the Java Location API (JSR-179), the user location can be tracked and submitted to the server. The server can (optionally) notify users, if potential matches (according to the profile) are close by. For privacy reasons, we do *not* allow location queries on a user basis (i.e. “where is user xy”).

[5]The application name is derived from the latin word *veneta* (blue, sea blue), to emphasize the decentralized character (*Blue*tooth).

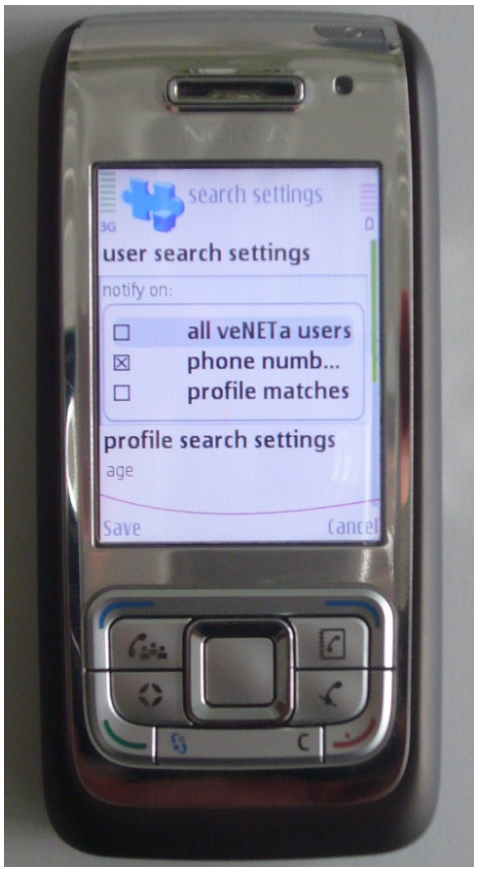


Figure 3. VENETA in action: The application can be configured to notify its owner whenever other VENETA users are nearby, if contact matches are found, and/or if profile matches are found.

We again want to stress that the system can be used independently of the server (at the expense of the server based features, of course). Figure 3 shows *VENETA* running on a Nokia E65 phone.

Finally, we would like to remark that VENETA could be extended by taking information from existing social

networking platforms into account. Existing profile information could be imported for profile matching, and the friend of friend detection algorithm could make use of other kinds of identifiers, such as *ICQ* numbers, e-mail addresses, and so on. Such extensions might further reduce bootstrapping issues.

V. Related Work

Mobile social networking has been an active field of research over the past years. As a consequence, a wide variety of systems have been proposed. Many of these systems primarily rely on central infrastructure, and thus do not fall into the main focus of this paper. Examples are *Plazes*, *Dodgeball*, *Jambo*, *Jaiku* or *Bluepulse*. All of them build around the location awareness concept, which is typically combined with traditional social networking functionality, such as the (centralized) exploration of friendship links. Some of them, such as *Jaiku* also offer some Bluetooth based features. The power of such features has been recognized by the developers of mainly decentralized applications. Commercial examples are *Nokia Sensor*, or *MobiLuck*, which both offer profile based matchmaking via Bluetooth. More remarkable is probably the *Lovegety* device from Japan[6]. The dedicated device exists in a "male" and "female" version, and in addition features a "mini-profile" (3 possible choices, one of which is a wildcard). The device alerts the user once a potential match is nearby. Since its introduction in 1998, more than 600K of these devices have been sold. It is probably the most successful mobile social system so far—presumably due to its simplicity.

Another approach was followed by *Sixsense*, which relies on laptops rather than mobile phones as primary platform. It is mainly thought for "quasi-static" settings, such as in class or in a library. In addition to Bluetooth, *Sixsense* makes use of Wi-Fi to explore the neighborhood.

Finally, a remarkable contribution from the academic world is the *Social Serendipity* project [2]. As part of the project, a mobile platform has been implemented. The system makes use of Bluetooth traces to track user behavior, which enhances the—again—server based profile matching techniques present in the system. The work, however, goes beyond matchmaking. The authors emphasize the big potential of mobile social software in various settings and provide user studies that indicate the high acceptance rate of such applications.

Interestingly, for most of the server focused approaches, browsing through friendship links to find friends of friends is a central feature. Decentralized systems, however, typically lack this features and build around other, mostly location related, concepts. We believe that this is mainly due to the problems of a server independent implementation of friendship browsing scheme. In this paper, we have thus addressed these issues. By relying on *real* friends (rather than only people participating in the system), and having the system searching (rather than the user browsing) for friends of friends, we mitigate a major problem: The lack of user density in mobile applications.

VI. Conclusion

We believe that a major target group of social mobile applications are young people that dispose of a low budget. We have addressed this group by implementing an application that provides a wide range of functionality free of charge. In particular, we have presented a technique that seamlessly incorporates the friendship exploration functionality of traditional social networking websites into purely decentralized environments. The arising privacy issues have thereby adequately been addressed. Considering the high popularity of such features in server based systems, we believe that our friend of friend detection mechanism might become an important ingredient in upcoming mobile social applications.

References

[1] R. Agrawal, A. V. Evfimievski, and R. Srikant. Information Sharing Across Private Databases. In *SIGMOD Conference*, pages 86–97, 2003.

[2] N. Eagle and A. Pentland. Social Serendipity: Mobilizing Social Software. *IEEE Pervasive Computing*, 4(2):28–34, 2005.

[3] M. J. Freedman, K. Nissim, and B. Pinkas. Efficient Private Matching and Set Intersection. In *EUROCRYPT*, pages 1–19, 2004.

[4] S. Hohenberger and S. A. Weis. Honest-Verifier Private Disjointness Testing Without Random Oracles. In *Privacy Enhancing Technologies*, pages 277–294, 2006.

[5] B. A. Huberman, M. K. Franklin, and T. Hogg. Enhancing privacy and trust in electronic communities. In *ACM Conference on Electronic Commerce*, pages 78–86, 1999.

[6] Y. Iwatani. Love: Japanese Style. *Wired News*, 11, 1998.

[7] A. Kiayias and A. Mitrofanova. Testing Disjointness of Private Datasets. In *Financial Cryptography*, pages 109–124, 2005.

[8] Y. Li, J. Tygar, and J. Hellerstein. Private matching. *Computer Security in the 21st Century*, pages 25–50, 2005.

[9] A. Yao. Protocols for secure computations. *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, pages 160–164, 1982.

[10] N. Zhang and W. Zhao. Distributed Privacy Preserving Information Sharing. In *VLDB*, pages 889–900, 2005.