Enhancing Privacy and Trust in Electronic Communities

Bernardo A. Huberman Matt Franklin Tad Hogg

Xerox Palo Alto Research Center
Palo Alto, CA 94304

Abstract

A major impediment to using recommendation systems and collective knowledge for electronic commerce is the reluctance of individuals to reveal preferences in order to find groups of people that share them. An equally important barrier to fluid electronic commerce is the lack of agreed upon trusted third parties. We propose new non-third party mechanisms to overcome these barriers. Our solutions facilitate finding shared preferences, discovering communities with shared values, removing disincentives posed by liabilities, and negotiating on behalf of a group. We adapt known techniques from the cryptographic literature to enable these new capabilities.

1 Introduction

With the advent of the World Wide Web and the ease of entry enabled by the Internet, electronic commerce is becoming an increasing reality, with a consequent growth in the number and variety of information providers and e-commerce sites. While this growth generates a diverse set of offerings from which consumers can only benefit, it also makes it hard for people to choose, in part because it is difficult to judge a priori the value of the offerings. In addition, since providers of electronic commerce sometimes lack recognizable reputations and can offer similar services, it is seldom possible to make optimal decisions as to which sites to access and which ones to avoid. As with many other situations where choice is costly, people resort to a cooperative mechanism which relies on the collective search performed by a whole community to find desirable and useful sites. Large groups of people surfing and buying on their own can sample a much larger information space than single individuals, and any exchange of relevant findings can increase the awareness of possibly interesting sites. Even though recommendations, both personal and institutional, can be unreliable and highly idiosyncratic, they decrease the cost of searching for optimal sources of information, while leading to the discovery of new sites and improved ways of surfing the Web.

Given these considerations, one would expect to find within the Web sites and communities that issue useful recommendations on a number of topics[1][2][3]. This information can then be used to create recommendations for other users and to identify similar individuals, thereby helping to make informal communities apparent. But while a great deal of economically useful information is distributed widely within groups of people such as large organizations, communities of practice[4], scientific communities and the economy at large, privacy issues make it hard to successfully exploit that knowledge. The limitations range from having to assess the quality of a recommendation from a group whose preferences might differ from the inquirer, to the natural reticence people have to reveal their preferences to an anonymous group with possibly different values. One issue that makes recommender systems perform below their potential is the difficulty of convincing potential advice-takers of the credibility and reliability of the recommendations. This depends in part on the willingness of potential recommenders to make available the right information at the right times. An important factor that dissuades potential recommenders from participating effectively is the risk that failed advice could lead to bruised reputations and liabilities[5].

As in the physical world, there exist a number of useful mechanisms to circumvent problems of privacy, trust and liability. For example, a useful strategy for maintaining privacy consists in the anonymous posting of information. In recommender systems this can be useful when the recommendations are based on coarse characteristics such as the number of people voting for a particular choice. But anonymity has the drawback of preventing users from learning the usefulness of recommendations from particular people, track trends over time, and to use reputations which are built up over repeated interactions. The consistent use of pseudonyms can address some of these issues, but not all. One drawback of pseudonyms is that the very link which establishes reputation over time becomes a vulnerability if authorship can be established by other means for *any* pseudonymous message. Issues of privacy can also be tackled by the use of trusted third parties to mediate the exchange of information. However, it can be difficult to get everyone in a community to agree on a suitable third party, particularly when new users continually enter the system. Furthermore, the collection of all information by a single third party can lead to a system-wide failure if such a party is compromised. What is truly desirable is the enhancement of privacy and trust in electronic communities without having to resort to anonymity, pseudonymity, or trusted third parties.

In what follows we address the above issues by the novel application of existing cryptographic techniques. In particular, we propose solutions to the problems of finding shared preferences, discovering communities with shared values and removing the disincentives posed by liabilities. In addition we propose a mechanism that allows an individual to negotiate on behalf of a group by proving membership in that group without revealing one's identity.

The remainder of this paper is organized as follows. We first review the basic cryptographic capabilities required in our discussion. In section 3 we introduce a protocol for finding shared preferences and finding communities of similar interests that preserves privacy and also allows for the use of selectivity based on reputations. Moreover, we show how to allow private communication among the members of such informal communities by sharing a public key that is issued only to them. Section 4 uses deniable signatures to remove liabilities in recommendations, while allowing users to discriminate based on their view of the recommenders' reputations. Section 5 deals with two issues: proving membership in a community without revealing one's identity, and establishing the size of that community without having to list its membership. Section 6 summarizes the findings and discusses the implications of this technology for electronic communities and their self-governance. Cryptographic details for all of our protocols can be found in the Appendix.

2 Cryptographic Primitives for Communities

The mechanisms we propose rely on a variety of cryptographic techniques, which in turn exploit the use of two fundamental cryptographic primitives: hash functions and public key systems. For the benefit of the reader unfamiliar with this field we now describe the general properties of these two primitives.

In general, cryptographic functions operate on inputs such as "messages" and "keys", and produce outputs such as "ciphertexts" and "signatures". It is common to treat all of these inputs and outputs as large integers according to some standardized encoding. Throughout this paper, you should assume that any value involved in a cryptographic function is a large integer, no matter what it may be called.

A cryptographic hash function, H, is a mathematical transformation that takes a message m of any length, and computes from it a short fixed-length message, which we'll call $H(m)$. This fixed length output has the important property that there is no way to find what message produced it short of trying all possible messages by trial and error. Equally important, even though there may exist many messages that hash to the same value, it is computationally infeasible to find even two values that "collide". This practically guarantees that the hash of a message can "represent" the message in a way which is very difficult to cheat. An even stronger property that we will require is that the output of a cryptographic hash function cannot be easily influenced or predicted ahead of time. Thus someone who wanted to find a hash with a particular pattern (beginning with a particular prefix, say) could do no better than trial and error. In practice, hash functions such as MD-5 and SHA are often assumed to have these properties.

Public key encryption (or signature) rely on a pair of related keys, one secret and one public, associated with each individual participating in a communication. The secret key is needed to decrypt (or sign), while only the public key is needed to encrypt a message (or verify a signature). A public key is generated by those wishing to receive encrypted messages, and broadcasted so that it can be used by the sender of the message to encode it. The recipient of this message then uses his own private key in combination with his public key to decrypt the message. While slower than secret key cryptography, public key systems are preferable when dealing with networks of people that need to be reconfigured fairly often. Popular public key systems are based on the properties of modular arithmetic.

3 Community Discovery

It is often the case that a group of individuals shares a number of preferences while being unaware of the existence of each other. While the Internet provides mechanisms for speeding up the process of discovering people with similar interests, it does not remove the disincentive inherent in having to disclose private information to unknown people while searching for a community that shares a given set of preferences. Consider, for example, the problem of finding someone from whom to request a recommendation about a particular topic. This can be difficult if one is reluctant to reveal one's preferences to people who might or might not share them. It would be useful to design a mechanism that circumvents this problem. In what follows we present a procedure that allows for a group of individuals to privately search for others with similar preferences while keeping their preferences private. Furthermore, this discovery process is made operational by producing keys that are available only to members of the group and allow them to communicate with each other.

Another application of this mechanism for community discovery obtains recommendations from users with similar interest profiles without contacting them directly. This application can be used in e-commerce situations to recommend products likely to be of interest based on the preferences of similar users. By maintaining both privacy and selectivity based on reputations, this mechanism allows for precise recommendations.

Community discovery can also be useful in creating additional services. For instance, as discussed in section 4, it can be useful in distributing pieces of some information among a group of people in such a way that no individual can determine the information but the group acting together can do so. Reputations for trustworthiness are an important aspect of discovering such groups, and preserving privacy can help encourage people to participate in this community service. An example is providing backup storage of private information.

These techniques could be useful not only for users providing information but also for those requesting recommendations. Examples include determining the majority opinion in a group without revealing individual preferences, or identifying significant but unpopular viewpoints. Furthermore, these protocols allow queries over a set of topics without revealing the particular question of interest. This could be useful when a sudden shift of interest in particular questions or products might change the group behavior in undesirable ways, e.g., causing changes in price levels before an e-commerce transaction is completed.

Community Discovery: Our approach to community discovery uses an idea that goes back to work by Bellare and Micali on non-interactive oblivious transfer[6]. Anyone can ask a question Q by posting it on a bulletin board. For simplicity, assume that Q is a yes/no question, although our techniques generalize to arbitrary multiple-choice questions. We can associate an unpredictable random "challenge" with each question Q in a standard way, by taking the challenge to be the hash of Q together with some system-wide public keys.

To answer a question Q, create two public keys y_0, y_1 that when multiplied together equals the challenge associated with Q. It turns out that it is easy to create these public keys in such a way that you know the corresponding private key for *one* of them. However, it is widely believed to be hard to create the public keys in such a way that you know the corresponding private key for *both* of them. Post on the bulletin board these two public keys y_0, y_1. Your "answer" will be rejected if these keys don't multiply together to the challenge. Otherwise, your answer is accepted, although no one can tell how you've really answered, because that depends on whether you know the private key for y_0 (in which case your answer was no) or the private key for y_1 (in which case your answer was yes). There is no need to post your answer anonymously. In fact, it may be desirable to require answers to be digitally signed, to prevent someone from joining both sides of the debate by answering the same question twice.[1]

Now anyone can encrypt a message that you can read only if you answered a question in a certain way. Suppose that I want to send you a message M, but I only want you to be able to read it if you answered no to question Q. Then I encrypt the message using y_0 as the public key. I can send this message to you directly, or post it to a bulletin board, possibly anonymously. If your answer was no, then you know the private key for y_0, and thus you can decrypt my message. Otherwise, you know the private key for y_1 but not y_0, and you cannot decrypt the message.

Community-Wide Conference Key: The mechanism described above is already enough for the community to find itself and begin a discussion. It might be desirable to generate a single key that was known to all members of the community to facilitate a community-wide discussion. One way to achieve this is to have any member of the community choose a secret key and encrypt it so that every other community member can decrypt it. For ex-

[1] Alternatively, cryptographic pre-registration techniques (e.g., using off-line electronic coins [22] as "one-show" credentials) could be used to prevent double answering.

ample, if I answered no to question Q, then I can choose a random "community-wide conference key" and encrypt it using the "no" public key for every answerer. All of these encryptions can be posted anonymously if desired, and signed with a proof of anonymous group membership as described in Section 5. Then everyone in the community (i.e., everyone who answered no to question Q) can decrypt to recover the community-wide conference key. It is easy for a new member to join the community in an ongoing discussion. This is achieved by posting an encryption of the conference key that the newcomer can decrypt only if he has joined the community, together with a signature of anonymous group membership so the newcomer knows that the key came from a fellow community member.

Private Preference Matching: The community discovery techniques described above could be repeated for a number of different questions. Then I could send you a message which you could read only if you answered each question a certain way, by encrypting the message so that all of the corresponding keys were necessary to decrypt. Another approach to multiple shared preferences is to perform a "secure distributed computation" to find people who answered questions in a compatible way. This can be done quite efficiently in the case where compatibility is measured by the number of yes/no questions that were answered in common.

A basic preference-matching function takes as input two lists of yes/no answers and a threshold. It outputs "true" if the number of answers where the two lists match is at or above the threshold. The one-against-many variant takes as input a "query" answer list, a "database" of answer lists, and a threshold. It outputs pointers to all answer lists in the database that have a sufficiently large match with the query list. The many-against-many variant is similar, except there are two database lists, or a single list compared against itself.

There are a number of techniques in the cryptographic literature for two or more parties to compute these kind of preference matching functions, under a wide variety of assumptions about the fault model, the amount of information leakage, the communication model, and so forth (see, e.g., [13, 17]). In the Appendix, we present one technique that is somewhat easier to describe, and quite efficient. Note that this scheme leaks a small amount of additional information, i.e., the number of matches in two preference lists rather than the one-bit decision about whether the number of matches exceeds some threshold. This is not an inherent limitation. It is a design trade-off to achieve greater efficiency.

4 Removing Liability

Finding someone satisfying a number of shared preferences is not enough in order to obtain a valuable recommendation. A potential recommender might be concerned about the liability that would result if the recommendation turned out to be of negative value to the requester. This is a concern for a number of communities, including malpractice-sensitive doctors, financial advisors, or even members of a recommendation system such as knowledge pump[3]. While anonymity might address this problem, it then generates another one, which has to do with the lack of a reputation that could be built over time. Pseudonyms allow reputations to be built over time, but they are "brittle" in the sense that uncovering the author of any message would establish the authorship of all messages with that pseudonym. Another possible approach is a contract in which the parties explicitly agree to waive liability, but this may be cumbersome and costly to devise, especially when multiple jurisdictions are involved.

We propose the use of "deniable signatures" to allow reputations and limited associations without fear of liability. With a deniable signature, the recipient of a recommendation knows it came from a person with an appropriate reputation, but cannot prove that connection to anyone else.

A further enhancement can give deniable signatures that are "convertible". The signer keeps an additional secret for each signed message which, when revealed, converts a deniable signature into an "ordinary" signature that is clearly and convincingly connected to the signer. This could give the recommender the flexibility to avoid liability as a default, while maintaining the ability to prove authorship if necessary. A further enhancement can distribute the conversion mechanism among many parties, to prevent the signer from being coerced to convert a deniable signature against his will.

If the verifier first tells the signer what signing key to use, then the resulting signature could be deniable. This interactive approach could be implemented using symmetric-key encryption and message authentication, which could make it especially efficient computationally. When it is infeasible to send a set-up message from verifier to signer, then other methods are needed. One particularly efficient approach to (non-interactive) deniable signatures that relies on ideas from Cramer, Damgaard and Schoenmakers [9] and Jakobsson, Sako and Impagliazzo[11]. It relies on a technique to prove knowledge of one out of two secret keys without revealing which is known.

Deniable Signatures: To begin, we describe a generic kind of three-round proof of knowledge. The Schnorr protocol [18] is an example that fits the model we describe. The prover knows a secret key that corresponds to a given public key. To authenticate himself to a verifier, the prover wishes to demonstrate knowledge of the secret key. They proceed as follows:

1. Prover → Verifier: cryptographic commitment based on prover's secret key
2. Verifier → Prover: random challenge based on verifier's random coin flips
3. Prover → Verifier: consistent response
4. Verifier accepts if response is consistent with commitment, challenge, and prover's public key. This is convincing to the verifier because it would have been very difficult for the prover to compute a consistent response without knowing the secret key.

To create a one-out-of-two proof of knowledge, another trick is needed. Many of these three-round proofs of knowledge (including the Schnorr protocol) have the remarkable property that it is very easy to forge transcripts – without knowing the secret key. The forger works backwards, starting with a (third-round) random response, and then choosing a (second-round) random challenge. Given these, it easy to compute a (first-round) commitment that will complete a valid transcript. (Of course, this doesn't contradict the security of the proof of knowledge, since there is a big difference between being able to forge a transcript by working backwards, and being able to fool a verifier in real-time going forwards.)

For the one-out-of-two proof of knowledge, the Prover *forges* a transcript ahead of time for the secret key that he does not know. Then the protocol is as follows:

1. Prover → Verifier: commit_1, commit_2
2. Verifier → Prover: challenge (only one!)
3. Prover → Verifier: challenge_1, response_1, challenge_2, response_2 such that $\text{challenge}_1 + \text{challenge}_2 = \text{challenge}$
4. Verifier tests the following:
   (a) consistency of commit_1, challenge_1, response_1, public key_1
   (b) consistency of commit_2, challenge_2, response_2, public key_2
   (c) $\text{challenge} = \text{challenge}_1 + \text{challenge}_2$.

The intuition behind this protocol is that the Prover is free to split the Verifier's challenge so that one of the pieces matches the pre-forged transcript. This leaves the Prover to respond to the other piece of the challenge, which he can do because he knows that secret key. If the Prover knows neither secret key, then it is very unlikely that the Verifier's challenge can be split to match *two* pre-forged transcripts.

These interactive protocols can be converted into non-interactive signature schemes by using the Fiat-Shamir heuristic [14]. The idea is that the signer plays the role of the Prover, but computes the Verifier's challenge himself using a cryptographically strong hash function such as MD5 or SHA. The signer applies the hash function to the message to be signed, the public key(s), and the commitment(s) "sent" in the first round.

Now we can describe the deniable signature scheme at an intuitive level (see Appendix for details). Sign a message using the non-interactive version of a one-out-of-two proof of knowledge, where the two public keys belong to the signer and the receiver. That is, the signer is proving knowledge of either his own private key or the receiver's private key. This signature could only have been produced by the signer or the receiver, and thus it is completely convincing to the receiver (by process of elimination!). However, no third party can tell whether the signer or the receiver has created this signature, and so the signer has deniability.

For convertibility, the signer takes advantage of the freedom that he has in generating the forged transcript. Instead of beginning his forgery with a random third-round response, the forger computes this value as the output of a cryptographic hash function. Revealing the input to the hash function would be convincing evidence as to which part of the transcript was forged, and thus which secret key must have been known to the signer. It is easy to distribute the conversion function among many parties, by using a sum of hash outputs instead of a single hash output to compute the forged response, where each party knows only one of the hash inputs.

5 Proving Membership in a Group

It is often the case that membership in a particular group or community can be valuable for establishing one's credentials, reputation or even for negotiating with another group or firm on behalf on one's group. And yet, there are many situations when one might desire to remain anonymous in case the group that one belongs to has an image or value that could be negative to the firm. Finally, group membership could be established without revealing the particular individual identity, which could be used as authorization or capability for some transaction or to negotiate with a firm or individual on behalf of the whole group.

There are a number of schemes in the cryptographic literature that can be used to solve these problems. We show one such scheme by adapting the approach from the previous section. One drawback of what we present here is that the effort involved (and the size of the messages) is proportional to the size of the group. It may be desirable to hide one's identity within a smaller group, giving up a degree of anonymity for greater efficiency.

The deniable signature scheme from the previous section relied on a kind of "one-out-of-two" proof of knowledge. The message signer proved that he knew either his own private key or the recipient's private key. This technique generalizes easily to a 1-out-of-n proof of knowledge, which is useful for anonymously proving membership in a group. It also generalizes to a t-out-of-n proof of knowledge, which is useful for anonymously demonstrating negotiating power. These generalizations can be realized as either an interactive identification protocol or a non-interactive signature scheme. We describe both of these generalizations following [9].

Group Membership: For a 1-out-of-n proof of knowledge, the prover begins by forging the Schnorr transcript for the $n-1$ private keys that he does not know. Then the protocol is as follows:

1. Prover $\rightarrow$ Verifier: commit_1, .., commit_n
2. Verifier $\rightarrow$ Prover: challenge (only one!)
3. Prover $\rightarrow$ Verifier: chall_1, resp_1, .., chall_n, resp_n such that challenge $= \text{chall}_1 + .. + \text{chall}_n$
4. Verifier tests the following:
   (a) consistency of commit_i, chall_i, resp_i, public key_i for every i, $1 \leq i \leq n$.
   (b) challenge $= \text{chall}_1 + \ldots + \text{chall}_n$

The intuition is as in the one-out-of-two case. The prover has the freedom to choose all but one of the challenges, and can use his knowledge of the secret key to respond to the one challenge he cannot control. The verifier cannot tell which transcripts were forged.

For proving knowledge of t-out-of-n private keys, everything is the same as before except for the relationship of challenge, challenge_1, $\ldots$, challenge_n. There must be a degree $n-t$ polynomial $f(x)$ such that $f(i) = \text{challenge}_i$ for every i, $1 \leq i \leq n$, and such that $f(0) =$ challenge. The intuition is that the Prover can forge transcripts ahead of time for the $n-t$ private keys that he does not know, and then interpolate to find $f(x)$ (uniquely determined) from those challenges together with the challenge from the verifier.

Signature versions of both of these protocols can be derived by using the Fiat-Shamir heuristic as before. More details of the cryptographic schemes described in this section can be found in the Appendix.

The signature version of the 1-out-of-n proof of knowledge is useful for distributing community-wide conference keys as discussed in Section 3. Say that I have created a conference key for everyone who answered yes to question Q. I randomly choose an additional $n-1$ parties who answered question Q, without knowing how they answered. Then I can prove knowledge of one of the private keys corresponding to the set of n "yes" public keys. By choosing a suitably large n, my identity is hidden well. By signing the encrypted conference keys in this way, anyone who successfully decrypts the conference key will have the added assurance that it was created by a fellow member of the community.

There are variations on these ideas in the cryptographic literature that allow for "identity escrow". This means that the true identity of the prover or signer might be recoverable under exceptional circumstances (e.g., with a search warrant). Relatively efficient implementations can be found in [8, 16]. These schemes also scale well as the size of the group increases, which is not the case for the simpler schemes described in this section.

6 Discussion

In this paper we introduced a number of new techniques for finding members of groups sharing similar preferences and obtaining their recommendations in ways that protect privacy while also allowing reputations to be built and updated. Moreover, these mechanisms do not require the creation of trusted third parties and their attendant problems. Reputations are extremely valuable in the context of electronic commerce, for when authenticated they provide a mechanism for trust to established, thus circumventing a number of costly transactions[19]. Trust is an important component of an efficient market, since fake postings of particular messages can lead to inefficient allocation of resources. Witness the recent posting of a bogus Bloomberg financial news story, which sent shares of a small technology company soaring 31 percent in one day, only to fall to previous values when the story proved false[20].

Another application we designed consists in the removal of the disincentive associated with the liability implied in issuing recommendations, thus making recommender systems more effective. Finally, we showed how individuals can prove membership in groups without revealing their identity, thus paving the way for negotiations between groups that seek to remain anonymous and firms that could profit from dealing with them.

These mechanisms involve trade-offs among computational efficiency, the leaking of information and ease of use. These trade-offs can be resolved differently depending on the specific application. For example, one may want to make it easier for new people to join a community by lowering the number of passwords and preferences that need to be listed, at the expense of reduced privacy. Another instance would be one in which everybody in a group shares the same key, which is a simple and secure procedure as long as no one leaves the group.

Additional trade-offs appear when on considers spoofing, whereby people can present false preferences in order to gain access to privileged information or to deter others from gaining an advantage from a weak adversary. One response might be anonymity, but at the cost of loosing the benefit of reputation building. Another one could be analogous to biological situations, where false signalling is used by many organisms to deter attack or to gain access to valuable resources. A strategy that has evolved to address the problem of spoofing in that context is for signals themselves to be costly to produce, and thus to imitate[21]. Similar strategies could be applied to electronic communities by increasing the number of challenges needed to access a given group, or by imposing a waiting period. On the other hand, this could deter legitimate new people from joining the group. Moreover, even if the trade-offs could be negotiated successfully, there remains the problem of misusing these techniques, as in the case of fraudulent financial transactions, insider trading or the unauthorized collection of personal data.

In spite of the great potential for electronic commerce that the Web is enabling through its global reach, there are vast areas of knowledge and expertise that remain untapped for lack of mechanisms that ensure privacy and trust. The techniques that we proposed make it easier to access vast repositories of information that are not readily known to producers and consumers, thus leading to improvements in economic efficiency through the more focused use of resources.

References

[1] W. C. Hill, L. Stead, M. Rosenstein and G. Furnas, "Recommending and evaluating choices in a virtual community of use", proc.CHI'95, (1995), 194-201.

[2] C. Avery and R. Zeckhauser, "Recommender systems for evaluating computer messages". *Communications of the ACM* 40, (1997), 88-89.

[3] N. Glance, D. Arregui, and M. Dardenne, "Knowledge pump: supporting the flow and use of knowledge", in *Information Technology for Knowledge Management.* Eds. U. Borghoff and R. Pareschi, Springer (1998).

[4] B. A. Huberman and T. Hogg, "Communities of practice: performance and evolution", *Computational and Mathematical Organization Theory* 1, (1995), 73-92.

[5] P. Samuelson, "Liability for Defective Electronic Information", *Communications of the ACM* 36, (1993), 21-26.

[6] M. Bellare and S. Micali, "Non-interactive oblivious transfer and applications", proc. Crypto '89, 547–557.

[7] S. Bellovin and M. Merritt, "Encrypted key exchange: password-based protocols secure against dictionary attacks", proc. IEEE Symposium on Security and Privacy (Oakland 1992), 72–84.

[8] J. Camenisch and M. Stadler, "Efficient group signatures schemes for large groups", proc. Crypto '97, 410–424.

[9] R. Cramer, I. Damgaard and B. Schoenmakers, "Proofs of partial knowledge and simplified design of witness hiding protocols", proc. Crypto '94, 174–187.

[10] W. Diffie and M. Hellman, "New directions in cryptography", *IEEE Transaction on Information Theory* 22 (1976), 644–654.

[11] M. Jakobsson, K. Sako and R. Impagliazzo, "Designated verifier proofs and their applications", proc. Eurocrypt '96, 143–154.

[12] T. ElGamal, "A public key cryptosystsem and a signature scheme based on discrete logarithms", *IEEE Transactions on Information Theory* 31 (1985), 469–472.

[13] R. Fagin, M. Naor, and P. Winkler, "Comparing information without leaking it", *Communications of the ACM* 39 (1996), 77–85.

[14] U. Feige, A. Fiat and A. Shamir, "Zero-knowledge proofs of identity", *Journal of Cryptology* 1 (1988), 77–94.

[15] S. Goldwasser and S. Micali, "Probabilistic public key encryption" *Journal of Computer and System Sciences* 28 (1984), 270–299.

[16] J. Kilian and E. Petrank, "Identity Escrow", proc Crypto '98, 169–185.

[17] M. Naor and B. Pinkas, "Oblivious transfer and polynomial evaluation", proc. ACM STOC, 1999, 245–254.

[18] C. Schnorr, "Efficient signature generation by smart cards", *Journal of Cryptology* 4 (1991), 161-174..

[19] F. Fukuyama, "Trust: the social virtues and the creation of prosperity", Free Press (1996).

[20] The New York Times, April 8, 1999.

[21] A. Zahavi and A. Zahavi, "The Handicap Principle: A Missing Piece of Darwin's Puzzle", Oxford Univ. Press (1997).

[22] D. Chaum, A. Fiat, and M. Naor, "Untraceable electronic cash", proc. Crypto '88, 319–327.

A Cryptographic Details

Discrete Log Problem: Let p be a large prime, and let q be a large prime factor of $p-1$. Let g be an element of order q modulo p. That is, $g, g^2, \ldots, g^q$ are all distinct elements modulo p, and $g^q = 1 \bmod p$. It is widely believed that the "discrete log" problem is hard in this setting: Given g, p, y, find x such that $g^x = y \bmod p$.

Discrete Log Based Community Discovery: The values p and g are system-wide parameters. The challenge associated with question Q is the cryptographic hash of Q, p, and g. To answer the question Q with $b = 0$ or $b = 1$, choose a random x between 1 and q, and compute $y_b = g^x \bmod p$. Then compute $y_{1-b} = c/y_b \bmod p$. Then post on the bulletin board your "answer" (y_0, y_1). If $y_0 y_1 \bmod p$ is not equal to c, then your answer is considered to be invalid, and it is ignored. Otherwise, your answer is accepted. If your answer is accepted, then it is extremely unlikely that you could know the discrete log of both y_0 and y_1. If someone encrypts a message using one of y_0 or y_1 as the public key, you will be able to decrypt it only if you know the corresponding private key.

One example of a discrete log based public key encryption scheme that can be used is due to Elgamal [12]. In the ElGamal public key encryption scheme, the public key for a user is some g, p, y, and the corresponding private key is the discrete log x such that $g^x = y \bmod p$. The encryption of message m is $(g^r \bmod p, my^r \bmod p)$, where r is chosen randomly by the encryptor. Given a ciphertext (u, v), the decryptor computes $v/u^x \bmod p = m$. Without knowing x, it is believed to be hard to decrypt ciphertexts (by an assumption related to the hardness of the discrete log problem)).

Private Preference Matching: Here is a protocol for Alice and Bob to evaluate the basic preference-matching function. It is related to the classic key exchange protocol of Diffie and Hellman [10], and to the encrypted key exchange protocol of Bellovin and Merritt [7]. Alice has the list $x_1, \ldots, x_n$ and Bob has the list $y_1, \ldots, y_m$.

1. $A \rightarrow B$: $H(x_1)^a, \ldots, H(x_n)^a \bmod p$, where a is randomly chosen, and where the list is randomly permuted.
2. $B \rightarrow A$: $H(y_1)^b, \ldots, H(y_m)^b \bmod p$, where b is randomly chosen, and where the list is randomly permuted.
3. $A \rightarrow B$: $H(y_1)^{ab}, \ldots, H(y_m)^{ab} \bmod p$, where the list is randomly permuted.
4. $B \rightarrow A$: $H(x_1)^{ab}, \ldots, H(x_n)^{ab} \bmod p$, where the list is randomly permuted.
5. Each party can now count the matches.

There are some interesting variations to consider. Suppose Alice and Bob don't randomly permute their lists in steps 3 and 4. Then they get to learn exactly which elements they have in common, and not just how many. Suppose Alice and Bob don't randomly permute their lists in steps 1 and 2, but do randomly permute in steps 3 and 4. Then they don't know exactly which elements they have in common, but they do learn the exact positions of those matches in the other party's original list. If those lists are originally sorted by preference, then this might be useful. Suppose Alice and Bob use a different random exponent for each element in each list. Further suppose that they don't randomly permute in steps 1 and 2. Now they learn only about those common elements that are in the same position in both original lists. Lastly, it is possible to incorporate a relatively efficient zero-knowledge proof that each party is following the protocol honestly, so that active cheating by either party (other than strategic choice of inputs) will be detected.

Schnorr Public Key Signature Scheme: In the Schnorr public key signature scheme, the public key for a user is some g, p, y, and the corresponding private key is the discrete log x such that $g^x = y \bmod p$. The signature of message m is $(g^r \bmod p, c, r + cx \bmod q)$, where r is chosen randomly by the signer, and where c is random but *not* chosen by the signer. One way to get c is to compute $c = H(g, p, y, m, g^r \bmod p)$ for some cryptographically strong hash function. To verify that (z, c, u) is a signature of m with respect to public key g, p, y, the verifier confirms that $zy^c \equiv g^u \bmod p$ and that $c = H(g, p, y, m, z)$.

Deniable Schnorr Signature: Suppose that party i wants to deniably sign a message m to party j. The deniable signature will be $z_i, c_i, u_i, z_j, c_j, u_j$, where $[z_i, c_i, u_i]$ is a valid Schnorr identification transcript for prover i, and where $[z_j, c_j, u_j]$ is a valid Schnorr identification transcript for prover j, and where $c_i + c_j = H(m, z_i, z_j, y_i, y_j, p, g)$. Party i proceeds as follows:

1. Forge a transcript $[z_j, c_j, u_j]$ of a Schnorr identification protocol for prover j, by choosing random $u_j, c_j \in [1 \ldots q]$ and computing $z_j = g^{u_j} / {y_j}^{c_j} \bmod p$.
2. Choose a random $r_i \in [1 \ldots q]$ and computes $z_i = g^{r_i} \bmod p$. (to "begin" the id protocol for prover i.)
3. Compute $c = H(m, z_i, z_j, y_i, y_j, p, g)$, and let $c_i = c - c_j \bmod q$.
4. Compute $u_i = r_i + c_i x_i \bmod q$. (To "complete" the id protocol for prover i.)

A deniable signature is valid if the two identification transcripts are valid, and if the challenges add up to the hash output as indicated. This signature could only be efficiently computed by someone who knows the private key of party i or party j. That is why it is convincing to party j when he receives it, and why it is deniable by party i afterwards.

Anonymous Group Membership For notational convenience, assume the group is parties 1 through n, and the prover is party 1. The prover begins by forging Schnorr transcripts $[z_2, c_2, u_2], \ldots, [z_n, c_n, u_n]$ for the private keys he does not know. The protocol proceeds as follows:

1. Prover (party 1) $\rightarrow$ Verifier: $z_1, \ldots z_n$, where $z_1 = g^{r_1} \bmod p$ for a random $r_1 \in [1 \ldots q]$
2. Verifier $\rightarrow$ Prover: c random in $[1 \ldots q]$
3. Prover $\rightarrow$ Verifier: $(c_1, u_1), \ldots, (c_n, u_n)$ such that $c = c_1 + \ldots c_n \bmod q$ and such that $u_1 = r_1 + c_1 x_1 \bmod q$.
4. Verifier accepts if $z_i y_i^{c_i} = g^{u_i} \bmod p$ for all i, and $c = c_1 + \ldots + c_n \bmod q$.

For the signature version, the signer computes his own challenge $c = H(m, z_1, \ldots, z_n, y_1, \ldots, y_n, p, g)$ where m is the message to be signed.

Anonymous Group Power For notational convenience, assume the group is parties 1 through n, and the prover knows the private keys of parties $1, \ldots, t$. The prover begins by forging Schnorr transcripts $[z_{t+1}, c_{t+1}, u_{t+1}], \ldots, [z_n, c_n, u_n]$ for the private keys he does not know. The protocol proceeds as follows:

1. Prover $\rightarrow$ Verifier: $z_1, \ldots z_n$, where $z_i = g^{r_i} \bmod p$ for a random $r_i \in [1 \ldots q]$ for every i, $1 \leq i \leq t$.
2. Verifier $\rightarrow$ Prover: $c \in [1 \ldots q]$
3. Prover $\rightarrow$ Verifier: $(c_1, u_1), \ldots, (c_n, u_n)$ such that $u_i = r_i + c_i x_i \bmod q$ for every i, $1 \leq i \leq t$, and such that $c_i = f(i) \bmod q$ for the unique polynomial f of degree at most $n - t$ satisfying $f(0) = c \bmod q$ and $f(j) = c_j \bmod q$ for all j, $t + 1 \leq j \leq n$.
4. Verifier accepts if $z_i y_i^{c_i} = g^{u_i} \bmod p$ for all i, $1 \leq i \leq n$, and if there is a polynomial f of degree at most $n - t$ such that $f(0) = c \bmod q$ and $f(i) = c_i \bmod q$ for all i, $1 \leq i \leq n$.

For the signature version of anonymous group power, the signer computes his own challenge $c = H(m, z_1, \ldots, z_n, y_1, \ldots, y_n, p, g)$, where m is the message to be signed.