Comparing Information Without Leaking It

Ronald Fagin
IBM Research Division
Almaden Research Center
650 Harry Road
San Jose, CA 95120 USA
fagin@almaden.ibm.com

Moni Naor*
Dept. of Applied Math
and Computer Science
Weizmann Institute
Rehovot 76100 Israel
naor@wisdom.weizmann.ac.il

Peter Winkler†
AT&T Bell Laboratories
600 Mountain Avenue, 2D-147
Murray Hill, NJ 07974-0636 USA
pw@research.att.com

Abstract

We consider simple means by which two people may determine whether they possess the same information, without revealing anything else to each other in case that they do not.

---

*Incumbent of the Morris and Rose Goldman Career Development Chair. Research supported by an Alon Fellowship and a grant from the Israel Science Foundation administered by the Israeli Academy of Sciences. Most of this work was done while the author was at the IBM Almaden Research Center.

†Most of this work was done while the author was with Bellcore.

1 Introduction

Consider the following problem, which actually arose in real life (we have masked the problem somewhat to protect confidentiality). Bob comes to Ron, a manager at his company, with a complaint about a sensitive matter; he asks Ron to keep his identity confidential. A few months later, Moshe (another manager) tells Ron that someone has complained to him, also with a confidentiality request, about the same matter.

Ron and Moshe would like to determine whether the same person has complained to each of them, but, if there are two complainers, Ron and Moshe want to give no information to each other about their identities.

The protocol typically used in a situation like this one is akin to the game "twenty questions," but goes by the name of "delicate conversational probing." Ron might ask Moshe if Moshe's complainer is male, and if the answer is "yes" Moshe might then ask Ron if Ron's complainer's surname begins with a letter preceding "M" in the alphabet. This goes on until Ron and Moshe have ascertained whether they have the same person in mind. When they do not, however (particularly when the first "no" occurs late in the game) a great deal of information may have been exchanged.

Before suggesting some superior protocols, let us point out that there are myriad other reasons why two or more parties might wish to "compare information without leaking it."

Obviously, whenever two persons suspect that they have the same person in mind to nominate for some post, or invite for dinner, or blame for a disaster, etc., but neither is willing to risk embarrassment by revealing the name, we have a situation much like the one with Ron and Moshe.

Similarly, two persons may each wish to determine whether the other is a co-member of a certain secret society. The test of membership, however, is knowledge of a password. The danger is that if A states the password, in order to convince B of his authenticity, and B is an impostor, then B may later be able to make illicit use of the password.

The reader will no doubt be able to imagine some circumstances where comparing information without leaking it will enable two persons to engage in an illegal activity without fear of entrapment. However, the techniques we suggest later might also be used by the forces of law. Suppose that the police suspect person A of masterminding a certain crime and have captured underling B. They have offered B immunity for incriminating A, but B is afraid to name A unless he can be assured that the police already suspect A. The police are hamstrung because if they suggest A's name to B, then B's statement won't hold up in court.

Still other cases arise when two people wish to gossip without spreading rumors (each would like to determine if the other is in possession of the same information) or to overcome shyness (perhaps there is a certain desirable course of action before them, which neither wishes to be the first to suggest).

In some situations the objective is to compare numerical values (that is, determine which is larger) rather than to test for equality. A typical case is Yao's "two millionaires" problem [20], in which the parties wish to determine who is richer without revealing their numerical worths. Similarly, a group of friends may wish to determine who among them is the oldest, or has had the most lovers, or has the "extreme value" of any potentially embarrassing parameter. We envision an adult game called "Are You Thinking What I'm Thinking?" in which both matching and comparison without leaking are employed to encourage people to reveal more about themselves.

A more serious application occurs in business and in labor relations, where much time is wasted when parties jockey to try to determine whether they have any common ground. Suppose party A is interested in acquiring a property belonging to party B, but would not consider paying more than $\$x$ for it; B is willing to sell but is not interested in entertaining any offer below $\$y$. A and B would like very much to know which is the greater value, because if $x \geq y$, then they can begin bargaining in earnest, and if $x < y$, then they can go home. But neither can reveal his number without undermining his bargaining position.

When more than two parties are involved, there are possible applications in a board room or even a jury room setting. For example, leaders often need to know what their advisors really think, in the absence of "fear of dissension." When a group is polled concerning the advisability of some course of action, it might be desirable to set things up so that if a majority agree, the outcome will be the same as if there were unanimity. Then no one need be afraid of distinguishing himself or herself as a renegade, and if a majority of those present have doubts, discussion will ensue. (Perhaps the Iran-Contra scandal could have been avoided!) Similarly, secret ballot elections are supposed to preserve the confidentiality of the voters, but in a small group the tally often tells all. Some of our techniques can be employed to hold elections in which only the name of the winner (or perhaps the names of candidates involved in a runoff) is revealed.

Although we have now gone far afield from Ron and Moshe, there is an obvious solution to all of the above problems.

Solution 1: JOSEPHINE

A trusted third party is found; all information is presented to him or her with the agreement that only the required results are revealed. In the case of Ron and Moshe, each could individually go to Josephine (the trusted third party), and tell her the name of the complainer. Josephine could then announce to Ron and Moshe whether the complainers are the same.

How satisfactory this solution is depends on how easy it is to find a person to play Josephine's role, and on how much information Josephine must be trusted with. Best would be if Josephine had no idea why Ron and Moshe are whispering names to her, although even then, Ron and Moshe might feel that they are violating the complainers' trust.

An exciting trend in cryptography in recent years is the study of protocols for secret function evaluation [13, 14, 17, 20, 21]. These protocols avoid the use of trusted third parties, and can be used to solve any of the above problems. Why don't Ron and Moshe just follow such a protocol?

A technical problem with such protocols is that they rely on unproven assumptions about computational complexity, such as the existence of functions that are easy to compute but difficult to invert by any feasible machine. In practice, a more serious problem is simply that they are complex schemes, which only certain experts can be expected to understand. Thus, blind trust is given to the system designer. An additional point is that these solutions are not yet genuinely practical even if implemented with the best possible care.

Our goal in this paper is to provide some schemes whose implementation is so transparent that no expertise is needed to assure correctness. We hope that some of these schemes will provide not only practical solutions to our problem, but also insight into the subtleties of communication and information.

As it happens, Ron and Moshe do happen to trust cryptographic protocols; their main problem is that they are lazy ("overworked" might be a kinder term). They don't want to program up such a protocol. Indeed, they would like to avoid any programming whatsoever, if possible. Their goal is a "quick and dirty" solution, which can be expected to work reasonably well.

Perhaps the time has come to list the properties that we might wish an information-comparing protocol to have, emphasizing those on which we will concentrate in the present work. We will remain loyal to the Ron-and-Moshe story, for simplicity; but the reader is invited to try to extrapolate our schemes to cover multiple participants, comparing numbers, etc. In what follows Ron's "value" is the name of the person who complained to him (or whatever other piece of information is the subject of the protocol) and similarly for Moshe.

A. "Resolution": Ron and Moshe should find out whether or not their values match.

B. "Leakage": Assuming they follow the protocol faithfully, Ron and Moshe gain no knowledge about each other's value, except whether or not they are equal.

C. "Privacy": No one else should gain any information about Ron and Moshe's values.

D. "Security": Neither Ron nor Moshe should be able to profit by cheating. That is, by failing to the follow the protocol, neither should be able to determine the other's value without matching it, or to determine whether the values match while denying that information to the other.

E. "Simplicity": The protocol should be easy both to implement and understand; that is, Ron and Moshe should be required to expend only a small amount of time, energy and money to learn, use and be confident of the protocol.

F. "Remoteness": Ron and Moshe need not be physically present at the same place in order to execute the solution.

Property B says intuitively that if the complainers are different, then neither Ron nor Moshe should have gained any information at the conclusion of the protocol as to who the other complainer is, other than the fact that it is a different complainer.

A comment is in order about property D. One virtually unavoidable problem is *lying by simulation.* Thus, we can do little about the possibility that, say, Ron tries deliberately to mislead Moshe by acting as if the person who complained to him was Alice, when it was in fact Bob. Of course, then Ron will determine whether the person who complained to Moshe is Alice, rather than determining whether the person who complained to Moshe is Bob. So in property D, we should really require that the cheater learns for exactly one value whether it is equal to the other party's input.

Cryptographic protocols are very good at satisfying properties A through D and F, with the help of complexity-theoretic assumptions. However, our interest is in achieving property E as well. To do so, we are willing to sacrifice some of the others when necessary; for example, Ron and Moshe trust each other, so property D is not so important. Condition C may be weakened to allow some information to pass to a third party; property B may be weakened by allowing small amounts of information to pass in the non-match case, or by allowing critical information to pass only in rare cases, or both. Depending on the circumstances property F may or may not be significant.

After presenting our proposed solutions, we will return to properties A through F and examine briefly how they fared.

2 Solutions

We now present some solutions suggested by ourselves or our colleagues. Note that some of the proposals assume that there is a (small) known list of candidates for "complainer," e.g. the department members; or that the name of the complainer is uniquely representable as an alphabetic string. Some of the proposals assume the existence of a trusted third party ("Josephine") and others do not.

Solution 2: COMPUTER PROGRAM

To avoid giving information to Josephine as in solution 1, we replace Josephine by a computer. Ron and Moshe could obtain a computer program that works as follows:

1. The program asks for input from Ron, who types, while Moshe is not looking, the name (Bob) of the person who complained to him.
2. The program then clears the screen, and asks for input from Moshe, who types, while Ron is not looking, the name of the person who complained to him.
3. The program clears the screen again, and announces whether the names that Ron and Moshe typed in were the same.
4. The computer finally erases all input information from its memory.

Solution 3: SPECIAL-PURPOSE DEVICE

Readers will note that, alas, third-party trust is still a factor in solution 2—namely, the computer programmer. In particular, how can Ron and Moshe be assured that the program will perform the last step, and not instead squirrel away the input for future use? Conceivably Ron and Moshe can take advantage of their professional expertise to actually *write* the program and input it side-by-side, but this is too much trouble for them. They might be especially concerned with debugging, since they would like to feel confident that there is no subtle error in programming that could cause the program to abort, spewing the data onto the screen, because, say, a name that was typed had more characters than anticipated or contained an unanticipated space.

To get around this difficulty the third author has designed (and obtained a patent for) a special-purpose box, called the Electronic Trusted Party, which implements solution 2. It is a portable electronic device that accepts information from two or more persons, compares or combines it in a preselected fashion, destroys the input information, and then displays the results. The critical feature of the Electronic Trusted Party is that it is *incapable* of storing information between uses or of revealing information other than as intended. It thus functions as a limited-access computing device, whose special features enable users to trust it with private information.

For serious applications, the device can be certified as genuine, by its manufacturer, in one of various ways; and it can be made tamper-proof, so that for example, opening the battery case blanks the memory and opening anything else breaks a seal.

The Electronic Trusted Party can be used for any of the applications described in the previous section, by selecting an appropriate mode (e.g. "match," "rank," "vote"), and specifying the number of participants and type of information desired. Like a computer program, it must to some extent be trusted; but the fact that it is a manufactured special-purpose device, without any means of accessing memory or any apparent way to misuse information, makes it much easier to trust in practice. The reader may imagine a plastic or metal box about the size of a hand calculator, but with alphanumeric keys and a special hood/cover to facilitate private entry of data.

Solution 4: RANDOM PERMUTATION

Another modification of solution 1 (JOSEPHINE) can be obtained by disguising the identities of the possible complainers. Let us assume, for example, that there are twenty possible candidates. Ron and Moshe agree on a random labeling of these candidates by numbers from 0 to 19; then each privately tells Josephine the number of the person who complained to him, and Josephine announces whether the two numbers received are the same.

Noga Alon of Tel Aviv University suggested that if the set of candidates is not clear or if the number of candidates is so large that it is not practical for Ron and Moshe to generate a random labeling of the name space, the same effect can be obtained using *universal classes of hash functions*, for example those of Carter and Wegman [5, 18]. A concrete example is as follows.

Fix a method of assigning unique numbers to candidates, e.g. the candidate's name interpreted as a number base 27 (alphabetic characters plus space), and let p be a prime number larger than the number assigned to any candidate. Ron and Moshe agree on two random numbers a and b mod p, where $a \not\equiv 0 \bmod p$, without telling Josephine. If x_R is the number corresponding to the person who complained to Ron, then Ron tells Josephine the number $ax_R + b \bmod p$. Similarly, Moshe tells Josephine the number $ax_M + b \bmod p$, where x_M is the number corresponding to the person who complained to Moshe. The point is that if $ax_R + b \bmod p$ and $ax_M + b \bmod p$ are different, then they are, as far as Josephine is concerned, simply a random pair uniformly chosen from $\{(c_1, c_2) \,|\, 0 \leq c_1 \leq p-1, 0 \leq c_2 \leq p-1, \text{ and } c_1 \neq c_2\}$.

Solution 5: RANDOM ROTATION

An even simpler implementation of the idea of RANDOM PERMUTATION was suggested by Silvio Micali of MIT. Let us assume that there are twenty candidates, to whom Ron and Moshe have openly assigned numbers from 0 to 19 (perhaps via alphabetical order). Josephine generates a random number k between 0 and 19, and tells it to Ron; Ron adds the number corresponding to his complainer to k, and passes the resulting number mod 20 on to Moshe. For example, if Bob is assigned the number 10 in the ordering, then Ron tells Moshe the number $k + 10$ mod 20.

Moshe takes this number, and subtracts from it the number corresponding to the person who complained to him, and passes the result mod 20 to Josephine. Josephine then announces whether the number she received from Moshe is the same as the number she passed to Ron. If it is, then Ron and Moshe know that it was the same person who complained to both of them, otherwise not.

Note here that Josephine does obtain some information in this solution, namely the difference between the numbers assigned to the two complainers.

Solution 6: PERMUTATION COMPOSITION

One of the shortcomings of RANDOM PERMUTATION and RANDOM ROTATION is that Josephine does find out whether Ron and Moshe have obtained a match. Can even this information be denied the third party?

We consider keeping the structure of the protocol. Thus, Ron and Moshe agree on some random string, send Josephine some function of their inputs, and Josephine announces a result from which Ron and Moshe can deduce the answer. A simple but imperfect approach (suggested by Bill Aiello of Bellcore) is for Ron and Moshe to repeat the procedure of solution 4 or 5 (say) 15 times, 14 of those being phony; for each phony procedure Ron and Moshe decide using a secret coin flip whether to provide Josephine with a match or non-match. Not knowing which procedure is the real one, Josephine is (probably) kept more or less in the dark as to the genuine outcome.

To keep Josephine completely in the dark while maintaining the structure of the protocol seems to require substantial effort, but it can be done. The following protocol is based on work by Feige et al. [11] on secret computation and builds upon [13, 14]. Using Barrington's Theorem [1], Ron and Moshe, working together, can associate with each candidate a pair of lists $\rho_1, \ldots, \rho_m$ and $\mu_1, \ldots, \mu_m$ such that

1. Each ρ_i and μ_i is a permutation of the set $\{1,2,3,4,5\}$;

2. If $\rho_1, \ldots, \rho_m$ corresponds to Ron's complainer and $\mu_1, \ldots, \mu_m$ to Moshe's, then the composition $\rho_1\mu_1 \cdots \rho_m\mu_m$ yields the rotation $\tau = (12345)$ when the complainers are the same, and the identity permutation otherwise.

Now Ron and Moshe generate and agree upon an additional list $\sigma_0, \ldots, \sigma_{2m}$ of *random* permutations of the set $\{1,2,3,4,5\}$, and they supply permutations alternately (and privately) to Josephine as follows. Ron gives her $\sigma_0\rho_1\sigma_1^{-1}$; then Moshe gives her $\sigma_1\mu_1\sigma_2^{-1}$; then Ron gives her $\sigma_2\rho_2\sigma_3^{-1}$, etc. Josephine composes these permutations and announces the result, which by construction is either $\sigma_0\tau\sigma_{2m}^{-1}$ or $\sigma_0\sigma_{2m}^{-1}$.

Ron and Moshe can of course tell from the result whether their complainers were the same, but Josephine has seen nothing but uniformly chosen random permutations. It is interesting to note that this method is applicable to any function, not just equality testing. The catch is that for an arbitrary function, the number m may be very large.

Solution 7: MESSAGE FOR MOSHE

Henceforth we leave Josephine behind and try to make do without the voluntary participation of a third party. Our next solution, which is actually a take-off on an old joke, was suggested by Russell Impagliazzo of the University of California, San Diego. Ron and Moshe assign a random telephone number to each candidate. Ron dials the phone number corresponding to the person (Bob) who complained to him, and asks to leave a message for Moshe. Of course, the person who answers the phone has no idea who Moshe is. A few minutes later, Moshe dials the phone number corresponding to the person who complained to him and asks if anyone has tried to leave a message for him.

Solution 8: AIRLINE RESERVATION

A rather neat practical version of MESSAGE FOR MOSHE takes advantage of the fact that airlines will not provide names of persons who have reserved flights. While Moshe is out of the room, Ron calls Delta Airlines and makes a reservation in the name of his complainer for the Tuesday afternoon flight from Atlanta to Salt Lake City. Moshe then takes over and tries to cancel the reservation in *his* complainer's name. Finally Ron cancels, or tries to cancel, the reservation he made.

Solution 9: PASSWORD

Dick Lipton of Princeton University and Nick Pippenger of the University of British Columbia independently suggested a protocol that relies on the presence of a computer but does not require any new programming. Ron changes his password to be the name of the person who complained to him (thus, Ron changes his password to BOB). Moshe then tries to log on as Ron, where he uses as a password the name of the person who complained to him. Moshe succeeds precisely if the same person complained to Ron and Moshe. Note that the password-checking program is (typically) designed fortuitously for the task at hand, and is heavily debugged for high security (so that, for example, when the password is typed in, it does not appear on the screen).

It is interesting that Pippenger thought of this solution by considering hash functions. This is because many computer systems store not the password, but a hashed version of the password, so that a system hacker cannot look at a table of stored passwords (instead, the hacker can see at best a hashed version of the password, from which it should be difficult to infer the password). In such a system, when someone types in his password, the computer compares the hashed value of the password typed in with the stored hashed value of the actual password. Thus, in such a system, the Lipton-Pippenger solution is a pre-programmed version of both COMPUTER PROGRAM and RANDOM PERMUTATION.

The following clever variation of PASSWORD, suggested by Alain Plagne of École Normale Supérieure in Paris, removes any possible temptation on Moshe's part to surreptitiously try more than one password against Ron's. Ron issues the "passwd" command, which asks for the old password and then the new one, for which Ron enters his complainer's name as before. Then Moshe takes over when the "passwd" program *demands confirmation of the new password.* Afterwards Ron checks whether or not his old password is still valid.

Solution 10: CUPS

Miki Ajtai of the IBM Almaden Research Center suggested a physical solution to our problem. We need to assume that there is a fairly small pool of candidates, say twenty. Ron and Moshe obtain twenty identical containers (perhaps by purchasing disposable cups), arrange them in a line, and write labels in front of each cup, one for each candidate.

Ron then puts a folded slip of paper saying "Yes" in the cup of the person who complained to him, and a slip saying "No" in the other nineteen cups. Moshe does the same. Ron and Moshe then remove the labels, and shuffle the cups at random. They then look inside the cups to see whether one of them contains two slips saying "Yes."

Solution 11: DECK OF CARDS

The following scheme gives a practical method for handling large or unspecified candidate sets, although it has compensatory flaws. It makes use of the coincidence that the number of playing cards in an ordinary deck is twice the number of letters in the Latin alphabet.

The red and black cards of a deck of 52 playing cards are separated to form two decks of 26 cards; each deck is then shuffled and placed face-down on the table. A 26-step procedure is now begun.

At step i, Ron removes the top card from each deck and puts the two cards together face-to-face, not looking at their values, with the card from the red deck on top. He takes the pair behind his back, and if the ith letter of the alphabet is in his secret name, he inverts the pair so that the black card is on top; otherwise he does nothing.

Then Ron passes the pair of cards to Moshe; Moshe takes the cards behind his own back, again inverting them just if the ith letter of the alphabet is in Moshe's secret name. The pair of cards is now placed on an accumulating "result stack" on the table.

After 26 steps all the cards are in the result stack, which is then riffle-shuffled carefully so that card colors do not show. Finally, cards are then dealt off the top of the result stack; a red card face up signals a mismatch of secret names.

It is interesting to note that Crépeau and Kilian [9] have shown how Ron and Moshe can evaluate any function secretly using a deck with 4 kinds of cards. All they have to be able to perform is a random cyclic shift of the cards. The number of cards required is proportional to the size of the circuit to evaluate the function.

Solution 12: ENVELOPES

We now give a physical solution which avoids leaking information (even if the name space is large) and requires no trust. This scheme is somewhat technical and may be skipped by the casual reader.

Suppose for ease in description that the name space is $\{0,1\}^n$, i.e. assume that the name of the person who complained to Ron is (encoded as) the sequence $x = x_1 x_2 \dots x_n$ of n bits, and the name of the person who complained to Moshe is the sequence $y = y_1 y_2 \dots y_n$

of n bits. As in many cryptographic protocols, our solution allows a small probability of error (here, the error represents the probability that Ron and Moshe believe, at the conclusion of the protocol, that their candidates are the same, when they are actually different). Let k be a positive integer large enough so that probability of error of 2^{-k} is sufficiently small to be acceptable.

Ron and Moshe each select $2n$ random numbers, chosen independently between 0 and $2^k - 1$. They each prepare a set of $2n$ identical envelopes, put one of their $2n$ random numbers in each envelope, and seal the envelopes. Ron and Moshe each arrange their $2n$ envelopes in pairs so that each pair corresponds to one of the n bit positions. For each of his pairs, Ron assigns one envelope to correspond to the bit 0 and the other to the bit 1, and similarly for Moshe. We could imagine that Ron's envelopes (and similarly, Moshe's) are arranged on the table in a $2 \times n$ rectangle, where the ith column corresponds to bit position i, with the top envelope in the ith column representing the bit 0.

Denote Ron's pairs by $(R_1^0, R_1^1), (R_2^0, R_2^1), \ldots (R_n^0, R_n^1)$ and denote Moshe's pairs by $(M_1^0, M_1^1), (M_2^0, M_2^1), \ldots (M_n^0, M_n^1)$. Ron computes $T_R = \sum_{i=1}^n R_i^{x_i} \bmod 2^k$, which is the sum of the random numbers that correspond to the bit values of his candidate. Similarly, Moshe computes $T_M = \sum_{i=1}^n M_i^{y_i} \bmod 2^k$.

Now Ron leaves the room. Moshe selects n envelopes from Ron's collection, by choosing from the ith pair the one corresponding to the value of y_i. The rest of the envelopes are put into a pile. Ron enters the room and verifies that indeed Moshe has chosen only n envelopes altogether. The unchosen envelopes are destroyed. Moshe opens the envelopes that he has chosen and sums up their contents together with T_M, i.e., computes $S_M = T_M + \sum_{i=1}^n R_i^{y_i} \bmod 2^k = \sum_{i=1}^n (M_i^{y_i} + R_i^{y_i}) \bmod 2^k$. Ron and Moshe exchange roles, do the same procedure with Moshe's envelopes, and Ron computes $S_R = T_R + \sum_{i=1}^n M_i^{x_i} \bmod 2^k = \sum_{i=1}^n (R_i^{x_i} + M_i^{x_i}) \bmod 2^k$. (Note: bitwise "exclusive or" may be used instead of sum in the above calculations.)

Ron and Moshe end the protocol as follows: Ron writes S_R on a piece of paper, Moshe writes S_M on a piece of paper, and they give each other their pieces of paper. If $S_R = S_M$, then they conclude that $x = y$, and otherwise they conclude that $x \neq y$.

It is not hard to verify the following:

- If Ron and Moshe follow the rules, and if $x = y$, then $S_R = S_M$, so Ron and Moshe conclude that $x = y$. If $x \neq y$ then $S_R \neq S_M$ with probability $1 - 2^{-k}$, so with probability $1 - 2^{-k}$, Ron and Moshe conclude that $x \neq y$.

- If the n envelopes that Moshe chose did not come from exactly the positions defined by Ron's bits $x_1, x_2, \ldots x_n$, then from Moshe's viewpoint S_R is simply a random number selected uniformly from the integers between 0 and $2^k - 1$. Thus, Moshe gains no information whatsoever from Ron's number S_R. A similar statement applies for Ron with respect to Moshe's number S_M.

A problem with the above scheme is that (say) Ron can learn whether $x = y$ but not give this information to Moshe. If Ron writes a random value on his paper, rather than S_R, then Moshe will (almost surely) conclude that $x \neq y$, whether this is correct or not, but Ron will know the true answer. We note that this problem can be corrected, but at the price of losing efficiency and simplicity (the correction involves a more complicated scheme, which among other things executes the above protocol k times).

Solution 13: DIGITAL ENVELOPES

The ENVELOPES solution can be implemented digitally (and thus converted into a cryptographic protocol). We now briefly describe how. It is based on a primitive called "one-out-of-two oblivious transfer" suggested by Even, Goldreich and Lempel [10], as a generalization of Rabin's "oblivious transfer" [15] (the notion was also developed independently by Wiesner in the 1970's, but not published till [19]). This primitive allows Ron to send two values R^0, R^1 to Moshe so that Moshe can choose to receive either R^0 or R^1. When Moshe chooses to receive R^j he gets no knowledge about R^{1-j}. Ron gets no knowledge about which R^j Moshe chose. Given such a primitive, it is not hard to see how to implement this solution digitally: for each $1 \leq i \leq n$ Ron sends to Moshe the values R_i^0 and R_i^1 in a one-out-of-two oblivious transfer. Moshe chooses to receive $R_i^{y_i}$. Similarly, Moshe does the same with the M_i's. At the end, Ron has S_R and Moshe has S_M, which they can compare (again, however, incurring the problem discussed at the end of Solution 12).

Following the execution of the protocol Ron will gain no additional knowledge as to Moshe's value, and vice versa.

In order to implement oblivious transfer, one must assume that Ron and Moshe have computational power corresponding only to probabilistic polynomial time. (That is, the computers that they have at their disposal can be simulated by a Turing machine with a source of random bits, such that the running time of the Turing machine is bounded by some polynomial in the input length.) Furthermore, given the current state of computational complexity theory, one must also make a "complexity theoretic" assumption. For instance, implementations of one-out-of-two oblivious transfer exist, under the assumption that factoring a random number of the form $N = PQ$, where P and Q are primes, is hard. (See, e.g., Brassard, Crépeau and Robert [4] for details of implementing oblivious transfer under such an assumption.)

3 Analysis

Most of the flaws in our solutions, relative to our six desired properties, will already have been spotted by the reader. We take a brief excursion through the properties with comments.

Property A: "Resolution"

All of the solutions work when there is a match, assuming no local aberrations (e.g. in MESSAGE FOR MOSHE the called party might untruthfully deny that a message was left for Moshe; and many of the solutions rely on agreement between Ron and Moshe concerning the spelling of candidates' names). Three of the solutions are capable of producing a false positive; DECK OF CARDS falls victim to accidental anagrams, and ENVELOPES and DIGITAL ENVELOPES admit a small probability of ruination by numerical coincidence.

Property B: "Leakage"

Since we make no assumptions about computational power in any of the solutions except DIGITAL ENVELOPES, we can make the strong but simple requirement that what Ron or Moshe sees as a result of the protocol (part of whose outcome is typically random) is independent of the other's complainer, assuming the complainers are different. This critical requirement is fully achieved in all the solutions except DECK OF CARDS, where there is a small possibility that no red card appears until the last few cards of the deck have been examined; then Ron and Moshe would be able to conclude that their complainers' names contain similar sets of letters.

In the DIGITAL ENVELOPES solution we do assume a limitation on the computational power of Ron and Moshe. In this case we must define what we mean by the statement "Ron and Moshe gain no knowledge about each other's value, except whether or not they are equal," since Ron and Moshe each gain an encrypted version of the other party's value. This statement can be made rigorous, as we now sketch. We consider the "ideal" solution, in terms of the information transferred to the other party, to be JOSEPHINE, where Josephine simply announces the result. Therefore what Ron (and similarly Moshe) will desire from a protocol is that Moshe's knowledge after executing the cryptographic protocol would not be greater than after executing JOSEPHINE. The idea is that no machine operating in probabilistic polynomial time should be able to distinguish the results of the cryptographic protocol from the results of JOSEPHINE. The definition of the security of such protocols and the design of protocols meeting these requirements have been the subject of intensive investigations in cryptography. A very accessible introduction to such “zero-knowledge” protocols is given in [12].

Property C: “Privacy”

Of the solutions involving a trusted Josephine, only PERMUTATION COMPOSITION gives nothing away to Josephine; RANDOM PERMUTATION reveals the resolution and RANDOM ROTATION also provides Josephine with information about the complainers when they are different. In JOSEPHINE, all is revealed.

Property D: “Security”

The issue of cheating in our protocols is quite complex. The solutions involving a trusted Josephine all suffer from possible collusions between Ron or Moshe and Josephine; this is perhaps a more critical problem than privacy in such solutions. Particularly weak, when Ron or Moshe can be tempted by the devil, are MESSAGE FOR MOSHE, AIRLINE RESERVATION and PASSWORD; all three allow one of the parties surreptitiously to try several candidates. COMPUTER PROGRAM permits fake programs (PASSWORD does also) and SPECIAL-PURPOSE DEVICE can fall victim to a phony device.

One particular method of cheating is for one of the participants to halt the protocol prematurely after learning what he needs to know (this is called the “walk-away problem”). For example, in the first version of PASSWORD, Moshe could obtain the answer without sharing it with Ron. This of course exposes the deserter as a cheater, but at times there may be excuses (such as “the line went down”). This problem does not have a completely satisfactory solution, but it is possible to limit the advantage of the deserter (see e.g. [2, 8, 10, 16]).

Property E: “Simplicity”

Admittedly, some of our proposals have gotten out of hand; PERMUTATION COMPOSITION and perhaps ENVELOPES are too complex for realistic consideration. SPECIAL-PURPOSE DEVICE will of course not be practical until such a device is marketed. COMPUTER PROGRAM and PASSWORD require computers, and the third-party solutions require Josephine, who certainly needs to be cooperative if not trustworthy.

Among the simplest solutions to execute, CUPS stands out when the list of candidates is known and small. Otherwise AIRLINE RESERVATION and DECK OF CARDS are effective, the latter better when there is more time but less trust. We suspect, though, that readers of this article will often find that PASSWORD requires the least time and preparation.

Property F: "Remoteness"

All the solutions using a trusted Josephine enjoy property F, that is, they do not require all parties to be present at the same location. AIRLINE RESERVATION and PASSWORD (except with Plagne's variation) can also be operated remotely. However, if we want a solution where Ron and Moshe are not tempted to gain more information, but where a physical solution is impossible, then we must use a cryptographic solution. In fact, if all that Ron and Moshe do is exchange messages between them, then Chor and Kushilevitz [7] show that cryptography is necessary, and Kilian [14] shows that the use of oblivious transfer is necessary. The DIGITAL ENVELOPES solution seems to be one that can be implemented with least cost (in programming effort and execution time).

4 Conclusions and End of Story

Which of these wonderful solutions did Ron and Moshe actually implement? The answer is: none of them! We now describe the solution they used.

Solution 14: REAL LIFE

The first author brought up the problem at dinner to his family. He explained that Bob had complained to him, and that he was trying to decide if Bob is the person who complained to Moshe. His then thirteen-year-old son Josh Fagin said, "Why not just ask Bob whether he complained to Moshe?" Because of its simplicity this was in fact the solution that Ron and Moshe actually implemented. Of course, this solution depends on the fact that the value that Ron and Moshe are trying to compare is actually the name of a person whom they can communicate with, rather than simply a number. It also depends on the fact that Ron knew that Bob wouldn't mind being asked (in fact, Bob was quite amused, and probably pleased at the lengths Ron went to in order to protect his confidentiality). As a humorous twist, which shows how even the best solutions can go astray in the real world, it turned out that Bob did not remember whether he had complained to Moshe! Upon further reflection, Bob decided that he probably had, and Bob and Ron agreed dynamically on a modification of the protocol: Bob asked Moshe if he (Bob) had complained to Moshe. It turned out that Bob was indeed the person who had complained to Moshe.

Acknowledgments

The Ron and Moshe of our story are the first author and Moshe Vardi, who together experienced a similar problem at IBM. A couple of years earlier the third author had been moved to consider the problem by an incident at Emory University, in which he and a colleague failed to discover that they had heard the same piece of juicy gossip.

All the authors found that the cryptographic solutions in the literature were not necessarily the best for the situation at hand. Suggestions for solutions were solicited from colleagues, primarily at the IBM Almaden Research Center and at the 22nd ACM Symposium on Theory of Computing held at St. Louis in May of 1990. Those who shared their thoughts with us but were not mentioned earlier include Richard Cleve, Tomás Feder, Steven Rudich, David Shmoys and Moti Yung. We thank Cynthia Dwork, Joseph(ine) Halpern and Moshe Vardi for comments on a draft of this paper.

Finally, we wish to thank Moshe Vardi for insisting on the simplicity of the solutions, thus forcing us to search further.

References

[1] D.A. Barrington, *Bounded-width polynomial-sized branching programs recognize exactly those languages in* NC^1, J. Computer Syst. Sci. **38**, 1988, pp. 150–164.

[2] M. Ben-Or, O. Goldreich, S. Micali and R. Rivest, *A fair protocol for contract signing*, IEEE Trans. on Information Theory **36**, 1990, pp. 40-46.

[3] M. Ben-Or, S. Goldwasser, and A. Wigderson, *Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation*, Proc. 20th ACM Symp. on Theory of Computing, 1988, pp. 1–10.

[4] G. Brassard, C. Crépeau and J.-M. Robert, *All-or-Nothing Disclosure of Secrets*, Advances in Cryptology - Crypto'86, Lecture Notes in Computer Science, No. 263, Springer Verlag, 1987, pp. 234–238.

[5] J.L. Carter and M.N. Wegman, *Universal Classes of Hash Functions*, J. Computer Syst. Sci. **18**, 1979, pp. 143–154.

[6] D. Chaum, C. Crépeau, and I. Damgård, *Multiparty Unconditionally Secure Protocols*, Proc. 20th ACM Symp. on Theory of Computing, 1988, pp. 11–19.

[7] B. Chor and E. Kushilevitz, *A zero-one law for Boolean privacy*, Proc. 21st ACM Symp. on Theory of Computing, 1989, pp. 61–72. Full version: SIAM J. Discr. Math. **4**, 1991, pp. 36–47.

[8] R. Cleve, *Controlled Gradual Disclosure Schemes for Random Bits and Their Applications*, Advances in Cryptology - Crypto'89, Lecture Notes in Computer Science, No. 435, Springer Verlag, 1987, pp. 573–587.

[9] C. Crépeau and J. Kilian, *Discreet solitary games*, Advances in Cryptology - Crypto'93, Lecture Notes in Computer Science, No. 773, Springer Verlag, 1994, pp. 319–330.

[10] S. Even, O. Goldreich and A. Lempel, *A Randomized Protocol for Signing Contracts*, Communications of the ACM **28**, 1985, pp. 637–647.

[11] U. Feige, J. Kilian and M. Naor, *On minimal models for secure computation*, Proc. 26th ACM Symp. on Theory of Computing, 1994, pp. 554–563.

[12] O. Goldreich, *Randomness, Interactive Proofs, and Zero-Knowledge–A Survey*, in **The Universal Turing Machine: A Half Century Survey**, edited by Rolf Herken, Oxford University Press, 1988, pp. 377–405.

[13] O. Goldreich, M. Micali, A. Wigderson, *How to play any mental game*, Proc. 19th ACM Symp. on Theory of Computing, 1987, pp. 218–229.

[14] J. Kilian, **Use of Randomness in Algorithms and Protocols**, MIT Press, Cambridge, Massachusetts, 1990.

[15] M. O. Rabin, *How to exchange secrets by oblivious transfer*, Tech. Memo TR-81, Aiken Computation Laboratory, 1981.

[16] M. O. Rabin, *Transaction Protection by Beacons*, J. Computer Syst. Sci. **27**, 1983, pp. 256–267.

[17] A. Shamir, R. Rivest and L. Adleman, *Mental Poker*, in **The Mathematical Gardner**, edited by David Klarner, Wadsworth International, Belmont, 1981, pp. 37–43.

[18] M.N. Wegman and J.L. Carter, *New Hash Functions and Their Use in Authentication and Set Equality*, J. Computer Syst. Sci. **22**, 1981, pp. 265–279.

[19] S. Wiesner, *Conjugate coding*, SIGACT News **15**, 1983, pp. 78–88.

[20] A.C. Yao, *Protocols for Secure Computations*, Proc. of the 23rd IEEE Symp. on Foundations of Computer Science, 1982, pp. 160–164.

[21] A.C. Yao, *How to Generate and Exchange Secrets*, Proc. of the 27th IEEE Symp. on Foundations of Computer Science, 1986, pp. 162–167.