Weakly Secret Bit Commitment: Applications to Lotteries and Fair Exchange

Paul Syverson
Center for High Assurance Computer Systems
Naval Research Laboratory
Washington, DC 20375
USA
syverson@itd.nrl.navy.mil

Abstract

This paper presents applications for the weak protection of secrets in which weakness is not just acceptable but desirable. For one application, two versions of a lottery scheme are presented in which the result of the lottery is determined by the ticket numbers purchased, but no one can control the outcome or determine what it is until after the lottery closes. This is because the outcome is kept secret in a way that is breakable after a predictable amount of time and/or computation. Another presented application is a variant on fair exchange protocols that requires no trusted third party at all.

1. Introduction

Secure computing and communication in distributed environments have typically assumed a foundation of strong cryptography. Indeed, it is a well known fallacy that security rests entirely in the strength of the cryptographic algorithms used. Weak cryptography is tolerated in practice because of a number of problems associated with stronger stuff: availability, cost (both monetary and computational), and legal or policy restrictions. It may seem surprising that weak cryptography is something not only to be grudgingly tolerated but also to be embraced as useful. In this paper we will explore applications of weak cryptography. Specifically, we will look at applications where the desired goals can only be attained if some of the cryptographic components are practically breakable at a predictable cost of time and/or computation. All of our applications will make use of some type of bit commitment where the confidentiality of the bit commitment is breakable in this way. We call this *weakly secret bit commitment* (WSBC).

We will consider two applications. The first is a means to hold a form of lottery wherein the winner is determined entirely by a set of public inputs from the bettors, but that determination cannot be made until after the inputs have been publicly identified. Thus, no one, not even the lottery organizer, can manipulate who wins. The second application is a variant on fair exchange. It combines some features of fair exchange with some features of contract signing protocols. Its main contributions are that it is relatively efficient (unlike contract signing) but requires no trusted third party (unlike fair exchange). It accomplishes this by a short sequence of messages that combine a diminishing disincentive to cheat with either increasing ability to complete the exchange without help or evidence that cheating has occurred. It thus functions not by preventing cheating but by structuring incentives so that enlightened, self-interested parties would prefer to complete the protocol than to cheat. For this reason, we call it *rational exchange*.

The remainder of the paper is organized as follows: In section 2 we will present the basic properties and concepts involved in WSBC. In section 3 we will present background information on the structure of lotteries. In section 4 we present general properties and assumptions for the lotteries presented in this paper. In section 5 we present a lottery scheme based on a collected WSBC. In section 6, we will describe another lottery scheme that shifts some of the trust and verification cost associated with the basic scheme by placing the WSBC on the individual tickets rather than on the collection of tickets. In this lottery, if all tickets are properly constructed, the result can be quickly confirmed by anyone once it is publicly determined. The lottery schemes presented earlier require anyone who wants to confirm the result to repeat the long, costly calculation of the result. In section 7, we describe how to nonetheless verify this calculation at least somewhat more quickly if multiple processors are available. In section 8 we will explore an entirely different application of WSBC, variants on fair exchange protocols. In section 9 we will discuss related work and will conclude the paper.

2. Weakly Secret Bit Commitment

In this section, we will set out the basic concept of WSBC and give some examples. But first, we briefly sketch the more basic concept of (secret) bit commitment.

Bit commitment is a way of requiring a principal to commit to a value without revealing that value. Here is a simple example taken from [12]. Alice generates two random-bit strings R_1 and R_2. She commits to a message M by creating $h(R_1, R_2, M)$ and sending $R_1, h(R_1, R_2, M)$ to Bob. When she wants to reveal M to Bob, she sends him R_2, M. By the properties of hash functions, Bob cannot determine M from the first message Alice sent. Also by the properties of hash functions, Alice cannot find R_2', M' such that $h(R_1, R_2, M) = h(R_1, R_2', M')$. First we note that, as the example illustrates, 'bit commitment' is a slight misnomer. This technique could be used to commit to a single bit, but it obviously can be used to commit to much more. (In the example as presented in [12], a single bit is committed to, making the need for R_2 all the more obvious. It should still be clear that it is needed for commitment to any message from a relatively small, predictable set of candidates.)

The idea of WSBC is similar to that of bit commitment. The difference is that we want the secrecy of the bit commitment to be breakable within an acceptable bound on time and/or computation.

The general properties that a WSBC function w should have are:

1. 2^{nd}**-preimage resistance:** Given x, it should be computationally infeasible to find $x' \neq x$ such that $w(x) = w(x')$

2. **weak-preimage resistance:** For any prespecified value y of w it should be *moderately hard* to compute any x such that $y = w(x)$

The first of these is the same as a standard desired property of hash functions taken from [9] (*q.v.* for more properties and discussion). The second is a variant of preimage resistance where 'computationally infeasible' is replaced by 'moderately hard'. The latter term is borrowed from [4]. Here, we mean to imply both upper and lower bounds on the ease of computation. We will say more about this below.

Depending on the application, we may wish to require other properties.

3. **collision resistance:** It should be computationally infeasible to find any x, x' such that $w(x) = w(x')$.
   This is stronger than 2^{nd}-preimage resistance.

4. **near-preimage resistance:** Given $y = w(x)$, it should be hard to find x' such that x and x' differ by a small number of bits.
   This is not directly similar to any of the hash function properties of [9]; although it is probably related to the non-correlation property.

2.1. Temporarily Secret Bit Commitment

For the applications we envision, it is important that it not be too easy to compute x from $y = w(x)$. Nonetheless, it would be nice if, given x and $w(x)$ one could quickly verify that x is in fact the argument of $w(x)$. This is related to the generally desirable property of hash functions that they be easy to compute. In fact, ease of computation implies this property. However, we must be careful in how we pursue it. For example, suppose we wish to show commitment to a random fixed-length binary value r from a relatively small space. A natural suggestions would be to use a hash of $s\hat{}r$, the concatenation of a fixed binary string s to r.[1] The length or r should be fixed so that the time to find r by brute force search on $h(s\hat{}x)$ is within the upper and lower bounds required by weak-preimage resistance. This or a related approach may be appropriate for applications where the goal is simply to have some computational load involved in breaking the bit commitment. For example, the pricing functions of [4], designed to complicate the sending of junk email, appear to have this property. And, this approach has the advantage of quick verifiability; however it may not be suited to all applications. The problem is that it is easy to break up the search space. Thus, any accuracy we may have on bounds on computation time cannot be assumed to indicate realtime bounds: the more processors that run the computation, the faster they will find r. Also, we can only predict an average time to find r. There is no way to guarantee that it cannot be found earlier. This would suggest a more restricted class of WSBC in which the breaking of the bit commitment can be more accurately related to time.[2] Following [10] we suggest that the important features are that it be calculated by an "inherently sequential" computation of predictable length. By 'inherently sequential' we mean a sequential computation for which the results of the preceding computations are needed to calculate the next value. In this way, we may speak of temporarily secret bit commitment (TSBC). For a TSBC function w we replace the requirement of weak-preimage resistance with the following more precise requirement.

4. **temporary-preimage resistance:** For any prespecified value y of w, to find any x such that $y = w(x)$,

[1] A similar use of hashes is described in [1], ironically for strengthening rather than weakening secrecy.

[2] I thank Gene Tsudik for first pointing this out to me. Similar observations can be found in [10].

should require an inherently sequential computation of length that is within prespecified bounds.

It is relatively easy to construct a means to commit to a value that is not known in advance. For example, we can commit to the result of performing n hashes on some value. In this case, if $y = h^n(x)$ then $w(x) = y$. For example, the collected lottery of section 5 might be designed in this way. But, this function has no means to be quickly verified, nor can it be used to actually commit to a known value, as is often needed. Fortunately, there are TSBC functions without these drawbacks. One such example, is the time-lock puzzles of [10].

2.1.1. TSBC via Time-lock puzzles

This is a very brief summary of a technique set out by Rivest, Shamir, and Wagner in [10]. Roughly, they take a secret s and encrypt it with a strong key K. They then perform repeated squares of a value wrt a composite modulus, the two large prime factors of which are known to the individual. They then effectively encrypt K with the result of those squarings. Knowing how to factor the modulus allows the individual to get the result of the repeated squaring much more efficiently. Anyone else has no choice but to perform all the squarings. This does not appear to be parallelizable in any way. The only advantage to be gained is by doing it on a faster processor, and the range of available processor speeds can be readily determined. Assuming the speed of the processor, this has very predictable computation time. Once, the squarings have been performed, the key K is revealed and s can be uncovered.

Note that there is a straightforward generalization from this technique to quickly verifiable TSBC for any message M. First encrypt M with some strong key K. Then, given $\{M\}_K$, take any TSBC function w and commit to K by forming $w(K)$. The TSBC to M is thus, $\langle \{M\}_K, w(K) \rangle$.

However, in general, there is no guarantee that it will be any quicker to form $w(K)$ than to derive K from it. The function chosen in [10] is especially nice because it allows quick calculation of $w(K)$ by the person who possesses the factors of the modulus. Thus, the TSBC is not only quickly verifiable but quickly calculated in the first place.

All of the applications we describe in the rest of the paper will require TSBC rather than WSBC in general. However, we will typically use the more general 'WSBC' when describing these applications. We now turn to look at lotteries.

3. Lotteries

We begin with some general background on lotteries before getting into the structure of the lotteries we have designed. Lotteries can have a variety of structures. There are options concerning the determination of prizes, the means of entering, and the determination of winners. Perhaps the simplest form of lottery is one where the names of entrants are placed in a pool and the winner(s) selected at random from this pool. For example, the names of all participants can be written on slips of paper that are placed in the proverbial hat. These are mixed to ensure randomness and then drawn blindly. Of course, the names can be replaced by pseudonyms or even one time tokens (such as a sequence of numbers). And, the physical selection of the winning entry can be replace by a means for identifying that entry (as in a policy or numbers game). If tokens or numbers are used, it can also happen that the pool of possible tokens for selection be a superset of those entered in the lottery.

Any of these variants is easily vulnerable to cheating on the part of those determining the winners, particularly if this is done out of public view. Solution to this problem include having the determination made by a trusted party, making the determination publicly visible, and/or leaving the determination to some public and unpredictable events. The use of trusted parties is limited not least by the trust that must be placed in the party. Nonetheless, this remains a widely used form of lottery (e.g., Publishers Clearinghouse giveaways). The trust necessary can be diminished if we watch the watchers, either by assigning other parties to evaluate the fairness of the drawing, or making it generally and publicly visible. For example, this was the case for policy rackets and remains the structure of most U.S. state lottery drawings. Even here there is a potential for abuse. The mechanism for randomizing the choice can be tampered with (assorted variants on weighting dice) so that even if the determination is properly conducted and public, it uses devices under the exclusive control of those determining the winner. Besides careful public inspection and more use of trusted parties, another approach is to leave the determination up to some event that is not under the control of the lottery organizers, preferably one that is widely visible. For example, traditional numbers games used such values as the n least significant digits in the result of adding together the winnings paid in a previously announced set of races at a race track, some n digits from announced digit columns in the total volume of stock traded on some exchange, etc. Here it is important that the event be unpredictable by and not under the control of the organizers or anyone else connected to the lottery.

Another issue is the nature of the winnings. The winnings may be predetermined independent of the entrants in the lottery. In this case, they may be monetary or other prizes. The winnings may be determined by the amount bet by the players, as in parimutuel betting. There may also be questions of whether or not all of the prizes are guaranteed to be awarded. As noted above, this is possible if the pool of entry tokens or numbers is a proper subset of the pool from which winners are drawn. If, however, the drawing is not considered complete until a winner is chosen, or if a winner is chosen by some other means when a winner is not drawn from the pool, then it is still possible that all prizes are guaranteed to be awarded. Also, if there is some suitable metric available, winners may be determined by closeness to the token or number drawn. If all the prizes are not awarded, they might no longer be considered as possible winnings. Alternatively, they might be added to the available prizes for the next instance of the lottery. An example of this is the "lotto" game run by most U.S. state lotteries, which roll the "jackpot" first prize over from one running of the lottery to the next if no one wins it in a that run. In this way, all prizes are ultimately awarded; however, all prizes are not awarded on each running of the lottery.

4. General Properties of Our Lotteries

In the next two sections, we will discuss two different lottery schemes. Nonetheless, they share certain properties. In this section, we first define properties they share. Then we describe in more detail some of the assumptions and features common to both of them.

1. **committed:** The lottery outcome is entirely determined by the ticket numbers of the purchased tickets.
2. **simple parimutuel:** $W \leq B$, where W is the total value of the winnings in a given run of the lottery and B is the total purchase cost of all tickets.
   'Parimutuel' means that the winnings are all from money bet in the lottery. 'Simple' means that all prizes are awarded; there is no rollover (unlike in U.S. state lotto games).
3. **weakly secret:** The outcome of the lottery cannot be determined before the lower time bound and can be determined by the upper time bound. The lower time bound is clearly after the termination of ticket sales.
4. **random and fair:** Even one random ticket number makes the outcome random, and no ticket is predictably more likely to win than any other.
5. **verfiable:** All of the above can be determined by anyone.
   Also the total ticket sales, the numbers of sold tickets, and the criteria for assessing the outcome and distributing winnings is publically verifiably determined clearly before the lower bound on determining the outcome.

4.1. Tickets

Before discussing tickets themselves, we set out assumptions about the environment in which the sale of tickets takes place. First, we assume that methods are independently available for authentication and integrity of ticket sales. Thus, the rightful purchaser of a ticket can be identified as such. (We leave aside questions of anonymous purchase. If there is no way to link a winnings claimant to a ticket purchaser, purchased tickets are bearer instruments. Thus we must also assume a means to protect the ticket from theft when it is transferred from vendor to customer, presumably encryption.) And, we assume tampering with a ticket purchase is detectable. This does not meant that denial-of-service is in any way prevented. It only means that attempts to alter a ticket by attacking communication between the customer and the vendor will be detected and that only unaltered tickets will be entered in the lottery. These are important assumptions, and any workable implementation of the lottery must have some means of satisfying them. However, there are various well known mechanisms for authentication and integrity protection that can be employed independent of the basic lottery design. So, we just assume them. Likewise we assume weak fair exchange of tickets, i.e., that when a ticket sale takes place the vendor gets his money and the customer gets his ticket, or at least that each can prove that the other received what was sent. It is possible, and even likely, that there are advantages to integrating mechanisms for fair exchange, authentication, and/or integrity into the lottery design itself. We leave this as a topic for future work.

Tickets themselves are an abstraction. There may be no ticket per se. For us *ticket* refers to either (1) a means of reliably associating a number, the *ticket number*, with a given individual (typically the purchaser) or (2) a token that associates the bearer with the ticket number. Tickets are assumed to be unforgeable. There are no valid tickets that were not issued by the ticket vendor. We leave aside discussion of how tickets are authenticated as not central to our design. Note that the vendor is free to issue tickets to itself or its friends, even without charging for them. For example, it is even possible for all but one of the tickets to belong to the vendor. We will see that if the disbursement of winnings is properly structured, the vendor gains no advantage by doing this over any other ticket purchaser.

There is a publicly announced *open of lottery*, a time after which customers can contact the ticket vendor for purchase of tickets. There is also a publicly announce *close of lottery*, a time after which no new tickets can be purchased. Both of these are announced well in advance of the actual running of the lottery.

As tickets are purchased, their ticket numbers are placed by the vendor on a *ticket list*. This list is to be publicly available, and it should be possible to tell that this list is authentically posted by the vendor. It may be periodically or occasionally updated during the sale of tickets. The final ticket list must be clearly labeled as such to differentiate it from lists that may not include all the tickets entered before the close of lottery.

Tickets are unique. For simplicity, we will assume this is accomplished by requiring tickets numbers to be unique. Of course the vendor cannot be allowed to choose ticket numbers for individuals or he could use values for which he has precomputed the lottery outcome. (He would actually have to compute several unless he can predict/control the total number of tickets sold. This is not an important distinction.) Individuals must be allowed to choose their own numbers, preferably in a random manner or at least one not predictable by the vendor. The vendor can attempt to control the ticket numbers to some extent by occasionally rejecting numbers that customers submit. If the determination of winners is fairly based on all tickets sold, and is unpredictable until after the lottery is closed, and the payoff is as described below, he can gain no advantage by doing so unless he can choose whether to accept or reject every single ticket number, which would be equivalent to letting him choose the ticket numbers.

4.2. Determining the Winner

The lottery outputs a single result that determines the winning ticket(s). As an example, one simple way to determine the winner(s) would be based on the ticket(s) that matches the lottery output in the most places. For simplicity, in the rest of the paper we will assume only primary winners. It is easy enough to establish secondary and tertiary winners, etc., as those whose ticket numbers match the lottery output to specified degrees less than those of the primary winners. This approach can be seen as a generalization of the usual lotto approach in which the primary winners are those who match the lottery output exactly and secondary winners match the output to a prespecified degree. If ticket numbers and lottery output are represented as binary numbers of the same size and ticket numbers are randomly chosen, then the winners can easily be determined by several means. Perhaps most naturally, the winners can be those whose ticket numbers differ from the output in the least number of bits (i.e., the Hamming distance from the output). We will see below that if ticket numbers chosen tend to be 'clustered' in the space of possible ticket numbers, this may not be a good way to determine the winner. We will return to this presently, and consider alternatives.

Lottery output is based entirely on the ticket numbers of purchased tickets, i.e., committed. If we were able to assume a publicly verifiable means of producing the lottery output that was also unpredictable by anyone who purchases tickets or awards prizes, then our design could be much simpler, in that it would not require the use of an algorithm that delays determination of the output until well after the lottery is closed. (As noted above, the means for computing the lottery output in traditional numbers games was assumed to be such a means.) We assume that such a means is not available, is not acceptable, or in any case is not used. Lottery output should be fair in the sense that any ticket is just as likely to win as any other. Of course it may not be possible to prove that the algorithm for computing the output is truly fair. As noted above, lottery winners should be determined by output. Thus, we would like something like a strict avalanche criterion [13] in which a change to any bit of input from the ticket list yields a 50-50 chance of change for each bit of the lottery output.

Time to compute the lottery output is acceptably bounded above and below. Specifically, The time to compute the lottery output is sufficiently long that the (final) ticket list has been publicly posted unquestionably before anyone could compute the lottery output. To be truly secure in this regard, the time to compute the output should be such that, even if there is only one ticket issued at the open of lottery, the time from when it was sent by the customer until the soonest the lottery output could be determined is sufficiently long after the close of lottery that the vendor could not possibly affect the outcome by issuing other tickets. And, time to compute the lottery output is not so long as to cause an unreasonable delay in determining winning tickets and distributing winnings.

We assume that clock skew and communication delays are strictly dominated by the bounds on time to compute the lottery. That is, we assume that the time to compute the lottery is orders of magnitude longer than any clock skew or communication delay. Relatedly, we assume that it is not possible for the vendor to mount an attack of the following type without getting caught. The vendor sells some number of tickets. He then breaks connections (at least to those customers who have bought tickets). Next he picks several sets of ticket numbers and calculates the lottery outcome based on the actually sold ticket numbers and his own sets. If he has chosen sufficiently many sets, odds are good that he should have at least one outcome that makes one of his tickets the winner. He builds the final ticket list based on the actually sold tickets and his own chosen set, and then he resumes communication. He claims that those were the tickets sold before the close of lottery and keeps all the money. There are a number of ways that this might be prevented. The vendor might be required to maintain a copy of the ticket list with a timestamp from a trusted time server. More generally, if the vendor cannot demonstrate by some acceptable means that the ticket list that yielded the lottery output was publicly available between close of lottery and some time adequately before the minimum time to calculate the outcome based on the open of lottery time, then the lottery is cancelled and all customers are issued a full refund for tickets purchased.

4.3. Payoff

Winnings are based on a simple parimutuel scheme. The winnings are also assumed to be a monetary value proportional to a fixed percentage of the total number of tickets on the list. This percentage is assumed to have been publicly posted before the beginning of the lottery. As we noted above, for simplicity we are assuming only primary winners. So, the winnings will simply be equally split between them.

If the payoff were not simple, then it would be possible for the expected value of tickets to rise enough that any increase in payoff is dominated by the amount rolled over (or otherwise exogenously added). In the simplest case, the vendor could simply do a run of the lottery where all tickets are his own or his collaborators. They would thus be guaranteed to collect all of the prize money. So, in the case of a lottery that is not simple parimutuel, some mechanism must be available to prevent this attack by the vendor.

Since for us the payoff is simple and is a fixed percentage of the number of tickets issued, it seems that the expected value of a ticket is fixed regardless of the number of tickets issued. Thus, the vendor would gain no expected advantage by issuing tickets to itself (or covertly to its friends). While doing so will decrease the probability that others will win, it will proportionally increase the payoff those others will receive if they do win. The vendor may even issue tickets without charging for them. The point remains that the expected value of a ticket remains constant. The only 'cheating' the vendor can do is to shift the extent to which that expected value is due to the probability of winning vs. the value of the payoff. But appearances can be deceptive. We now consider a way that the vendor *can* gain an advantage by issuing tickets.

4.4. Spatial Distribution of Ticket Numbers

The above claim about expected value would be true if ticket numbers were uniformly distributed throughout the space of possible ticket numbers. If this is not the case, and the winner is determined by Hamming distance, the vendor can mount an attack that lowers the expected value of the sold tickets, thus increasing his expected earnings in the lottery. As a very simple example, suppose a lottery is run with $W = .5B$. Thus, if tickets cost \$1.00, the basic expected value of a ticket should be 50 cents. Suppose also that there are only four possible ticket numbers, $\{00, 01, 10, 11\}$. If only one ticket was purchased, with ticket number 00, and the vendor issues two false tickets with numbers 01 and 10, then the expected value of the sold ticket is just under 19 cents. The vendor's attack becomes all the more profitable if several tickets are purchased with value 00. This attack cannot be defeated by simply requiring that ticket numbers are unique or that a large number of different people buy tickets. As long as the distribution of tickets is skewed in such a way that the vendor can issue tickets with numbers that are closer to more of the ticket number space than are the numbers on sold tickets, he will be able to run a version of this attack.

An important question is the complexity of such an attack. If choosing ticket numbers to bend the lottery to the vendor's advantage is at least on the order of the complexity of calculating the lottery outcome, then it may not seem to be a threat. However, unlike the exact outcome of a given lottery, generic distribution calculations may be run in advance, especially if typical ticket distributions are somewhat predictable. Building on this, the vendor might even be able to construct a meta-attack that need not be executed each time the lottery is run. The vendor may have heuristics that point out to him when particularly glaring skews of ticket numbers for sold tickets are present. He may then employ his attack and not employ it otherwise. In section 6, we will consider a lottery in which the vendor cannot make use of such heuristics even if he had them, thus rendering the complexity questions about the meta-attack moot.

Although, we will see a way to prevent the vendor from running heuristic checks on lottery runs in progress, there is nothing to prevent him from assessing the the typical distribution of tickets over several runs of the lottery and taking advantage of any pattern he sees or even tailoring the space of possible ticket numbers to facilitate the attack or increase the advantage. If customers often choose numbers based on dates for example, the vendor may be able to choose a space and a set of numbers for which it is relatively easy to approximately bound off the 'date' subspace from the rest of the ticket number space. The vendor may even be able to encourage clustering of purchased tickets by any number of ways. Like lottery vending machines of today, he might provide 'random' ticket numbers for those who do not want to pick their own. Or he might be more indirect, e.g., by being covertly involved in the publication of books on lottery playing strategies.

Of course the space of possible ticket numbers and sets of ticket numbers chosen in previous runs should all be public information. And, anyone can take such advantage should he recoginize it. Thus, in some sense there is no 'attack' here at all. Savvy bettors should gradually occupy the ticket number space, always having a certain advantage over their less savvy counterparts, much as in some other games of chance, such as poker (or, in another way, go).

However, we can also have a more intellectually egalitarian game. One way would be to require people to use random numbers. This has some disadvantages: it would be inconvenient at best to enforce, and it would require people to trust and/or maintain psuedo-random number generators. They could still choose their own favorite numbers, but they would have to salt them with something random, which may be desirable in any case.

Another possibility is to set the winner not by the minimum Hamming distance, but by Hamming distance within some boundary. Anyone within this boundary wins. Again, this is like U.S. state lottery games, but the Hamming distance for first place there is 0. This will remove the advantage for those who attempt to draw boundaries around empty or low density parts of the space. However, when there is clustering in small parts of the space of ticket numbers, it may still be advantageous to scatter tickets throughout the rest of the space, depending on the amount of clustering and the size of the boundary. Note also, that there may be no winner under this approach. Thus, to use this means of calculating the winner, we must violate either our assumption that the total amount awarded W is a fixed ratio of the total purchase B or our assumption that the lottery is simple parimutuel.

Alternatively, the winner could be determined by other means than the Hamming distance from the lottery output. For example, the output could be taken modulo the total number of tickets sold N, where they are considered in the order they were entered in the ticket list before the lottery close. (Any secondary and tertiary winners could now be determined by closeness to the winning number. Both Hamming distance and the ticket list order appear adequate measures of closeness for this purposes; although I have not examined this carefully.) If the outcome were unpredictable mod N, this would remove any spatial bias because the randomness is over the ticket numbers played rather than over the space of all potential ticket numbers. (I am assuming that the lottery output is always potentially much larger than N.) This is one suggestion. Any other mechanism that chooses winners in true lottery fashion, i.e., by picking randomly and without bias from the entered ticket numbers, could be substituted. We need make no judgement here about which type of game should be played (one that picks fairly from the domain of ticket numbers vs. one that picks fairly from the actually played ticket numbers). To be fair, this only need be clearly explained to potential players in advance.

5. Lottery with Collected WSBC

For our first lottery design, we need some additional assumptions. Specifically, we assume that we can produce from the list of ticket numbers L via a 2^{nd}-preimage resistant function, such as a hash, a value $h(L)$ in the range of a WSBC function w. (Note that it is conceivable, though unlikely, that h here is simply identity.) Thus, by the properties of w, there exists o such that $w(o) = h(L)$ and such that it is moderately hard to find o given $w(o)$. This o is the output of the lottery.

This is perhaps the most straightforward way to build a lottery employing WSBC. This way of determining lottery outcome is also more similar to the one given in [7] than the one in section 6. So, we will now discuss some of the differences. One obvious difference is that the lottery of [7] employs a delaying function, as it is called there, in place of our use of WSBC. Like inverting WSBC, delaying functions are meant to be moderately hard to compute. And they are meant to be entropy preserving, in the information theoretic sense. This, is obviously similar to the collision resistance or 2^{nd}-preimage resistance of WSBC. Delaying functions may simply be inverses of a class of WSBC (perhaps even precisely inverses of TSBC). However, this is speculation, and we leave the exact relation between the two as a topic for future work.

Another important difference is that the lottery in [7] requires authentication of ticket purchasers and that a minimum number of different purchasers obtain tickets during the “critical phase” of the lottery. This is basically a period beginning during the sale of tickets and ending at the close of sale. It is designed to guarantee that neither the lottery organizer nor anyone else is ‘cornering’ the lottery by buying tickets himself or using a handful of accomplices. By insuring that a minimum number of distinct purchasers buy during this period, and assuming that the probability of collusion between that number or more is low enough, we can be sure that the probability of such ‘cornering’ is sufficiently low.

Of course, as we have noted, in the lotteries presented in this paper, anyone is free to corner the lottery in this way. This also means that there is no need for registration, authentication, or other complications. So, why bother? One answer is that this allows for lotteries that are not simple parimutuel. They may either have outside prizes or may be parimutuel with rollover. When the payoff on such lotteries gets high enough, they may attract disproportionate numbers of customers (and purchases per customer). Thus, this has advantages for the vendor. Also, even if the expected dollar value of a ticket goes down under these circumstances, that the potential prize is so large may make playing advantageous for customers. (It is a mistake to measure expected utility of playing as linearly related to expected monetary value of tickets. If this were not true, there would be no point in playing a lottery at all.) There are thus both advantages and disadvantages to requiring authentication.

Because the lottery in [7] is parimutuel with rollover. It is thus possible for this lottery to use the Hamming-distance-within-a-boundary method of calculating winners described in section 4.4 without violating any other constraints we have set on running a lottery. As noted there, in the presence of clustering, the vendor may still theoretically be able to take advantage of a distribution attack on a lottery if this method is used to calculate the winner(s). But, this may be exacerbated in a parimutuel lottery with rollover. If he has set the Hamming distance boundary for winning properly and perhaps done things to encourage clustering, he should be able to create fairly high jackpots and then play a spread of ticket numbers with either a guarantee or a high probability of winning, at the same time not dropping below the threshold of purchasers during the critical purchase phase. In fact, if he only plays his 'spread' during high jackpots, he can be sure that there will be a larger than average number of (noncolluding) players. An analysis that is beyond the scope of this paper would be necessary to determine if building such an attack is really worth the risk and trouble. If it should be the case, then other methods of determining the winner should be used, e.g., the modulo N method given above.

6. Lottery using Individual WSBC

The basic collected WSBC lottery does not require anyone to trust the vendor or anyone else concerning the fairness of the lottery. Trust lies solely with the algorithms that transform lottery input to lottery output (and their implementations). However, there is still an element of trust vs. convenience that must take place. Specifically, if anyone wishes to confirm the outcome of the lottery, she must reproduce the presumably time consuming and/or expensive calculation on her own trusted platform and wait for the results. A step on the way to overcoming this is to have an algorithm operate not on the entire conglomerate list of ticket entries but on each ticket number individually. The advantage here is twofold. First, the ticket holder can in this way know the result of applying the algorithm to his ticket number ahead of time. More importantly, since the algorithm is the inverting of a WSBC, he does not have to perform it at all: he only has to perform the WSBC itself, which may be much easier.

We will call the input s he chooses for w the *private ticket number* and the output $w(s)$ the *public ticket number*. The final calculation of the lottery output can be a fast calculation, e.g., via a good hash function, from the private ticket numbers. These can be obtained by breaking the secret bit commitments $w(s)$ that constitute the public ticket numbers. (Private ticket numbers would be taken as input to this calculation in some fixed order, presumably the same as that of the public ticket numbers in the ticket list ordering.)

Perhaps the most important advantage of the splitting of the WSBC is that it allows the individuals to precompute their own commitment value. This has several advantages, but the paramount one is that it can be based on a secret that the individual possesses. This means that the lottery result can be quickly and easily verified by anyone. It also means that the lottery result can potentially be determined more quickly without any threat to fairness.

6.1. Quick Verification of Lottery Results

We can use the quickly verifiable technique of section 2.1.1 to commit to each private ticket number. So, $w(s) = \{s\}_K$, the encryption of s with K. (More precisely, $w(s) = \langle\{s\}_K, w'(K)\rangle$, where w' is a TSBC function used to commit to K.) Once, K has been calculated for each of the tickets, all the keys can be published. Thus, anyone can quickly verify the lottery result by performing the encryptions and computing the lottery output. The lottery output may be quickly verified, but it is still necessary to wait for the result of breaking the secret bit commitments before the lottery outcome can be determined. However, depending on the cooperation of the lottery participants, this too can be accelerated.

Once the lottery has been publically determined to be closed, individual ticket holders can reveal their private ticket numbers and the keys used to encrypt them. If everyone does this, then the lottery outcome can be determined—presumably more quickly than by breaking the bit commitments, which must include a temporal safety margin. There is no guarantee that everyone will cooperate. However, no one can know that they have won (or lost) until this is done. So, it is in the interest of the participants to cooperate. (The calculation is similar in this respect to rational exchange, which will be discussed below.) Even allowing for the irrational, sociopathic, or incompetent ticket holders, and the system errors that may occur, it is probably still in the interest of individual ticket holders to reveal their secrets. This is because the vendor probably has only so many processors to devote to the breaking of the bit commitments. Thus, the more people send in their secrets, the faster the outcome can be determined. This may even imply a solution (for this case only) to a classic coordination problem; when something requires the cooperation of lots of people, it is rational to think that someone will fail to do his part. Thus, it is not worth doing your part. Since doing your part here helps even if some others don't, it is a better bet to cooperate. This may lead to a higher number of lottery runs where everyone actually does send in his secrets.

6.1.1. Malformed Tickets

The above discussion all assumes that each of the tickets is valid. If any of the tickets was not produced via encryption using an appropriate key and the correct algorithm, this can be determined (provided there is enough redundancy to determine that a given message was encrypted using a given key, e.g., by including a hash of the message within the encryption). Such a ticket might be declared invalid. Alternatively, the technique for extracting the commited key will yield a unique key whether or not this is a key that was used to encrypt the private ticket number; thus, the private ticket number can be declared to be the decryption, using that key, of the public ticket number, even though that private ticket number was clearly not chosen and committed to in this way. The latter choice has the advantage of avoiding many, though not all, of the customer relations problems accompanying the first choice. However, whatever choice is made, the existence of any such improperly formed tickets has the effect of destroying the quick confirmation of the lottery outcome. The lottery outcome can be quickly determined, ignoring the malformed tickets, but it is the determination that such tickets are malformed that cannot be quickly verified.

Malformed tickets should be relatively rare. Indeed, it is conceivable that most lottery runs will not contain any. Customers are not likely to submit malformed tickets since it will cost them money to do so. (Recall that we assume integrity protection of ticket purchase; so, the ticket chosen by a customer can assumed to be the ticket entered in the ticket list.) If necessary, ticket purchase can even be authenticated as in [7], as a means to reveal anyone who submits a malformed ticket. That individual can then be sanctioned by, at least, excluding him from any future lottery runs. And, assuming customer generated malformed tickets rarely occur, a vendor is not likely choose invalid ticket numbers since this will make her lottery less appealing to the public than that of another vendor. Nonetheless, it is still possible that malformed tickets will occur enough to remove the advantage of quick verification.

7. Parallel Verification of Computations

As noted, slow verification is a limitation on the collected lottery and is potentially a limitation on the individual lottery as well. But, we can speed up the verification by parallelizing it. This is somewhat surprising since the computations themselves are designed to resist parallelization. However, even an "intrinsically sequential" computation must have intermediate results.

Any sequential computation requiring S steps can be broken into arbitrarily many smaller pieces. For example, if $S = nk$, and c_i is the i^{th} intermediate result, the verification of a computation of c_S can be broken into n pieces of the form $\langle c_i, c_{i+k} \rangle$. The verification of this piece is then to confirm that k computations on c_i yields c_{i+k}. And, the verification time of calculating c_S can be thus reduced by a factor of n (assuming n processors working in parallel).

Using this method, it is possible to split the verification of the collected lottery outcome or to split the verification that any malformed ticket is indeed malformed. Individual ticket purchasers are unlikely to have many processors at their disposal; however, this would allow mutually trusting customers to split the duties.

Moreover, by splitting things up in this way, it is possible to verify computations probabilistically. This should also ameliorate the limitation on easy verification. If an error were equally likely to come in any segment, then checking some number of segments at random would reduce the probability that the computation is incorrect proportionately. However, in our case, the probability of an error is higher towards the end. Thus, it makes sense to check the last segments and then check backwards with increasingly less likelyhood of checking as the initial segment is approached.

8. Fair Exchange

As the name implies, fair exchange protocols are designed to provide some guarantee to a participant that she cannot be cheated by someone who agrees to exchange some goods and then takes what she sends but does not send his own agreed upon goods. (Cf., e.g., [6, 2].) Purists reserve 'fair exchange' for those protocols that guarantee to each participant (eventual) delivery of the agreed goods. This may also be called *strong fair exchange*. *Weak fair exchange* guarantees to a participant that she receive either the agreed upon goods or evidence that the other principal has access to the goods she sent.

Somewhat related to fair exchange protocols are nonrepudiation protocols [15, 16]. These provide either *nonrepudiation of origin*, proof of who the sender of a message is, *nonrepudiation of receipt*, proof of who received the message, or both. Some protocols provide both some form of fair exchange and some form of nonrepudiation. The protocol we describe below is such a protocol.

The protocols we present do not, in general, provide even weak fair exchange. As such, the title of this paper is a misnomer. However, the protocols result in a form of exchange that appears similar to fair exchange. What they do is to provide incentives so that principals operating out of enlightened self interest have more reason to proceed with the protocol at each point than to abort. It is for this reason that we call them rational exchange protocols. Also, if someone sends a message that is supposed to provide some value or goods after a fixed amount of time or computation, but sends something else instead, this can be shown. Thus, the protocol provides this feature of weak fair exchange. At the same time anyone wishing to prove that he has proceeded to a given point in the protocol must therefore also show that his counterpart has proceeded to the immediately prior point. Thus, there is a degree of nonrepudiation evidence provided.

One of the bottlenecks for fair exchange (and nonrepudiation) protocols is the trusted third party (TTP) that typically is necessary to guarantee the protocol has the desired properties. Much work has been done, e.g., in the above cited papers, to reduce this bottleneck. Improvements include moving the TTP offline so that it need only be contacted if one of the principals fails to cooperate. The protocols we describe require no TTP at all. We will explain this below.

Related to fair exchange is the phenomenon of fair contract signing. (Cf. [12] for a high level description and references.) Like the lottery and unlike the fair exchange and non-repudiation protocols already discussed, in fair contract signing there is typically no trusted third party at all. Very roughly, parties exchange bits or larger parts of their signatures on a contract in an alternating fashion. Thus, assuming roughly equal computing power, each party has at all points roughly the same chance of committing the other party to the contract with about the same amount of effort. (Ben-Or et al. [3] present a fair contract signing protocol without many of the limitations and assumptions of the basic contract signing protocols we describe here. However, it requires at least a limited use of a TTP.) The main problem with contract signing is that to maintain this parity only small amounts of information can be conveyed with each message. Thus, these protocols tend to be highly inefficient. Our goals are slightly different, and we are thus able to be much more efficient.

8.1. Rational Exchange Using WSBC

We begin with a simple example where a vendor V is selling some *Goods* to a customer C. (We follow the somewhat usual conventions that $[X]_{K_P^{-1}}$ is the a digital signature on X using principal P's private key K_P^{-1}, and $\{X\}_K$ is the encryption of X using key K.) The main mechanism we will employ is to encrypt a value using a strong key K and then commit to K, in other words the quickly computable TSBC technique of section 2.1.1.[3]

Message 1 $\quad V \rightarrow C:\quad$ [*Description of Goods*, $\{$*Goods*$\}_K, w(K)]_{K_V^{-1}}$

Message 2 $\quad C \rightarrow V:\quad$ [*Payment, Message 1*]$_{K_C^{-1}}$

Message 3 $\quad V \rightarrow C:\quad$ [K, *Message 2*]$_{K_V^{-1}}$

What might such a protocol be used for? (1) If the vendor is selling relatively low value items, it may typically not be worth it for the customer to break the encryption to recover the item. This may be literally true, in terms of the cost of computation, or may be based on the value of the inconvenience of delay. If cost of computation is itself enough of a factor, we may rely on general WSBC rather than TSBC as presented. (2) Alternatively, the vendor may have some item that is of timely and diminishing value such as short term investment advice or regularly changing lists of bargain items for sale. The commitment to K can be structured so that, without K, the *Goods* will be of greatly diminished value by the time they are recovered. If structured properly, it may be worth it to the vendor to take a chance on unknown customers in this way, but to refuse to service repeat customers who fail to pay. (The protocol might begin one step earlier with a signed request for *Goods* sent by the customer.)

After the first message, the customer has something which will either turn out to be what he wants or evidence that the vendor has sent the wrong thing. The customer now sends payment along with acknowledgement of the first message. The vendor can thus only show acknowledgement that he sent the goods (at least in weakly encrypted form) if he also shows that he received payment. Once the third message is received, the customer can only prove that he sent payment by also proving that the vendor sent information to quickly get to the *Goods*. Even though this information is just a key, the included *Message 2* will show that the key reveals the *Goods*.

The vendor might not send the third message for a long time. A way to counter that possibility would be to add timestamps to the protocol. But, that would defeat the point of having no TTP. For applications we envision the primary deterrent against the vendor delaying sending of *Message 3* is that the vendor gains nothing by doing so except a bad reputation that could ruin her business. In any case she will not prevent the customer from getting the *Goods*, albeit after a slightly longer wait and/or greater effort, unless she has really not sent $\{$*Goods*$\}_K$ at all in the first message. But, in that case she has provided evidence to the customer that she sent something else, evidence that could be used to show a judge that she has not fulfilled her part of the protocol. If the penalty for this form of cheating greatly exceeds the value of the goods, then it is even less in the vendor's interest to cheat.

We could also add a fourth message in which the customer acknowledges timely receipt of *Message 3*. Customers generally have no incentive to send such a message, however. In an account based system, the vendor might offer an incentive to customers that turn in a certain number of acknowledgments. Here the vendor would have to be trusted to keep track of the number of acknowledgments sent or we would need still another message, etc.

8.2. Generic Fair Exchange Using WSBC

We now describe a more generic protocol for fair exchange between two principals. Alice will exchange her

[3]David Goldschlag has suggested a similar version of this protocol, using delaying functions, as an improvement on an earlier version.

goods $G(A)$ for Bob's good $G(B)$. The protocol uses the same WSBC commitment to a key as in the previous example. However, the commitment is more quickly, or more easily, breakable in each successive message. By the end of the protocol each party should have an encryption of the other party's goods that is accessible very quickly.

Message 1

$$A \rightarrow B : \quad [\textit{Descrip. of } G(A), \textit{Exch. parameters}, \{G(A)\}_K, w_1(K)]_{K_A^{-1}}$$

Message 2

$$B \rightarrow A : \quad [\textit{Descrip. of } G(B), \textit{Exch. parameters}, \{G(B)\}_{K'}, w_2(K'), \textit{Message } 1]_{K_B^{-1}}$$

Message $2n+1$

$$A \rightarrow B : \quad [w_{2n}(K), \textit{Message } 2n]_{K_A^{-1}}$$

Message $2n+2$

$$B \rightarrow A : \quad [w_{2n+1}(K'), \textit{Message } 2n+1]_{K_B^{-1}}$$

Exchange parameters should include such things as the length of computation in each message to find the encryption key. They may take into account such things as assumed asymmetry in processor speed, in trust, e.g., between customer and vendor, and asymmetry in the goods being distributed, e.g., money for investment advice. They might also include some minimum time that principals are held responsible for maintaining evidence of all the messages sent in the protocol. As before, if the cost of finding the preimage to the bit commitment can be compared to the value of the goods, then it may be possible to use general WSBC instead of TSBC. In either case, it is generally important to use a bit commitment function w that is quickly computable, such as the one in [10], cited above in section 2.1.1. Otherwise, it will take impractically long for the principals to prepare their messages.

The protocol has a structure similar to contract signing; nonetheless, it is typically likely to be much more efficient. This is because we need not proceed by such tiny increments. For example, suppose for simplicity that Alice and Bob are exchanging items each of which will be diminishingly useful for about the next three days after which they will be useless. The protocol could be structured so that the first message should contain encryption of goods that would take on average four days for Bob to break. A second message could be breakable by Alice in about a day. A third message could be breakable by Bob in about four hours. A fourth message could be breakable by Alice in about forty minutes. A fifth message could be breakable by Bob in about a minute. A sixth message could reveal K' completely.

Only three messages are sent each way. And, at each stage of the game parties have an easier time cheating, but what they can get will be worth less than if they proceed. Perhaps more importantly, they are providing the other party with evidence that they agreed to the protocol and received previous messages. Thus, if, e.g., Bob does not send the fourth message, then (1) Alice can show a judge messages 1 through 3, and the judge can rule that Bob did not comply with the exchange if he does not either respond with message 4 in a timely manner or eventually show that he received message 5. (2) Alice can still break the encryption in the second message in about a day. Thus, she can still get part of the value of Bob's goods, while he is presumably responsible for having failed to complete the protocol.

What if one of them is sending completely random garbage? This should also be provable in a short amount of time. Even the first message can be shown to be such. As in the purchase protocol above, penalties and compensation for this form of cheating could well exceed the value of what is exchanged, thus making it worthwhile for people to go through the computation to demonstrate it and making it not worthwhile to cheat in this way.

9. Related Work and Conclusions

The basic idea of using WSBC seems related to a number of other ideas that have arisen lately: the time-lock puzzles of [10], the pricing functions of [4], and the delaying functions of [7], among others. As noted in the text, pricing functions appear to be related to general WSBC, while time-lock puzzles and delaying functions are related more specifically to TSBC. Also, in [5] values based on repeated hashing, as above in section 2.1, are used as a measure of elapsed time.

There have been several previous cryptographic treatments of lotteries. However, they have generally been used as forms of payment mechanisms rather than to run actual lotteries (cf., e.g., [11, 14]). The most closely related work to our general approach to lotteries is [7]. In fact many of the basic ideas arose in joint discussion with those authors. However, development of the basic ideas was done independently so there are some important differences. These have been discussed in section 5.

The rational exchange protocols discussed are perhaps most similar to the work of Jakobsson in [8]. His main idea is to allow the cryptographic equivalent of ripping a bill, say a $20 bill, in half and giving half to the vendor before the vendor provides the goods. The second half is given afterwards. Similar to our approach, part of the idea is not to force compliance with the protocol but to remove incentives to cheat.

In this paper we have introduced the concept of weakly secret bit commitment and shown two applications. We have shown how to construct lotteries that are entirely determined by the tickets purchased but are such that this determination cannot be made before the lottery is closed because the lottery result takes too long to calculate. For the individual lottery, we have shown how to quickly verify the results provided that there are no malformed tickets. For both the collected lottery and the individual lottery in the presence of malformed tickets, we showed how to speed up the verification of lottery results by parallelization. This also allows one to perform verification that is probabilistic in its assurances but still faster too perform. We have also introduced the idea of rational exchange and showed how to use WSBC in exchange protocols that are efficient and require no TTP.

Recall that we assumed weak fair exchange in the purchase of lottery tickets. Since lottery tickets themselves are of small purchase value, it seems possible to have a use of WSBC integrating rational ticket purchase into the lottery design. We may explore this in the future. For now, we observe that the presented examples have shown weak protection of secrets can sometimes achieve things that strong protection cannot.

Acknowledgements

Many of the basic ideas for the collected lottery scheme were developed during conversations with David Goldschlag and Stewart Stubblebine at FC'97 in Anguilla, BWI in February 1997. I thank them both greatly for these conversations. I am thankful to Cathy Meadows for many helpful conversations while writing this paper. I also thank Gene Tsudik and the anonymous referees for their helpful comments. This work supported by ONR.

References

[1] Martín Abadi, Mark A. Lomas, and Roger Needham. "Strengthening Passwords", SRC Technical Note 1997 - 033, Sept. 4, 1997 (with minor revisions on Dec. 16, 1997).

[2] N. Asokan, M. Schunter, and M. Waidner. "Optimistic Protocols for Fair Exchange", *Proceedings of the* 4^{th} *ACM Conference on Computer and Communications Security (CCS'97)* (ACM Press), pp. 7–17, April 1997.

[3] M. Ben-Or, O. Goldreich, S. Micali, and R. Rivest. "A Fair Protocol for Signing Contracts", *IEEE Transactions on Information Theory* , vol. 36, no. 1, pp. 40–46, Jan. 1990.

[4] Cynthia Dwork and Moni Naor. "Pricing via Processing or Combatting Junk Mail", Weizmann Institute Tech. Report CS95-20, 1995. (Preliminary version presented at CRYPTO '92.)

[5] Matthew K. Franklin and Dahlia Malkhi. "Auditable Metering with Lightweight Security", in *Financial Cryptography: FC '97, Proceedings*, R. Hirschfeld (ed.), Springer-Verlag, LNCS vol. 1318, pp. 151–160, 1998.

[6] Matthew K. Franklin and Michael K. Reiter. "Fair Exchange with a Semi-Trusted Third Party", *Proceedings of the* 4^{th} *ACM Conference on Computer and Communications Security (CCS'97)* (ACM Press), pp. 1–6, April 1997.

[7] David M. Goldschlag and Stuart G. Stubblebine. "Publically Verifiable Lotteries: Applications of Delaying Functions (extended abstract)", *Financial Cryptography (FC'98): Preproceedings*, Anguilla BWI, February 1998. Final proceedings forthcoming from Springer-Verlag.

[8] Markus Jakobsson. "Ripping Coins for a Fair Exchange", *Advances in Cryptology—EUROCRYPT '95* (Springer-Verlag LNCS 921), pp. 220–230, 1995.

[9] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*, CRC Press, 1997.

[10] Ronald L. Rivest, Adi Shamir, and David A. Wagner. "Time-lock puzzles and timed-release Crypto", Unpublished, March 1996.

[11] Ronald L. Rivest. "Electronic Lottery Tickets as Micropayments", in *Financial Cryptography: FC '97, Proceedings*, R. Hirschfeld (ed.), Springer-Verlag, LNCS vol. 1318, pp. 307–314, 1998.

[12] B. Schneier. *Applied Cryptography, Second Edition: Protocols, Algorithms and Source Code in C*, John Wiley and Sons, 1996.

[13] A.F. Webster and S.E. Tavares. "On the design of S-boxes", *Advances in Cryptology—CRYPTO '85* (Springer-Verlag LNCS 218), pp. 523–534, 1986.

[14] David Wheeler. "Transactions Using Bets", in *Security Protocols:* 4^{th} *International Workshop*, M. Lomas (ed.), Springer-Verlag, LNCS vol. 1189, pp. 89–92, 1996.

[15] Jianying Zhou and Dieter Gollmann. "A Fair Non-repudiation Protocol", *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, (IEEE CS Press), pp. 55–61, 1996.

[16] Jianying Zhou and Dieter Gollmann. "An Efficient Non-repudiation Protocol", *Proceedings of the 1997 IEEE Computer Security Foundations Workshop (CSFW 10)*, (IEEE CS Press), pp. 126–132, 1997.