Implementation of a Digital Lottery Server on WWW

Kazue Sako

C&C Research Laboratories, NEC Corporation
4-1-1 Miyazaki Miyamae Kawasaki 216-8555 Japan
sako@ccm.cl.nec.co.jp

Abstract. This paper presents the implementation of a digital lottery server on WWW. The aim of this server is to offer an outcome that a group of users can agree to be random. The server allows users to define and start a lottery session, participate in that session, and verify its outcome.
While the scheme employs Cryptographic tools for purposes of security, it keeps their complexity to a minimum, so as not to complicate actual operations or adversely effect the ease of actual use.

Keywords: lottery, fairness, multiparty protocol, implementation, hash function

1 Introduction

The widespread use of digital networks has created a kind of cyber-society whose citizens are increasingly able to participate in the full range of activities associated with a real community. One convenience which they still appear to lack, however, is a handy system for drawing lots. In a real community, a group of people can gather in one place to draw lots by themselves or to observe that a representative indeed drew in a fair manner. Being physically present is essential to be assured of fairness, i.e., that there was no cheating in the process, but this is essentially impossible in a cyber-society.

Cryptographic protocols provide a theoretical basis for achieving assured fairness, and studies on secure multi-party computation demonstrate the possibility of determining a winner randomly [1,2]. Goldschlag and Stubblebine present a simple lottery scheme based on a delaying function[3]. These schemes can be used in principle to build a lottery system, given their specifications. However, an actual tool to support a flexible, *universal* lottery, whose design and purposes might be modified, has, to the author's best knowledge, yet to be reported.

This paper presents the implementation of a convenient digital lottery server to be used on the WWW. It offers an outcome that a group of users can agree to have been determined randomly. The server may be used for selecting at random a single winning participant or multiple winners at various levels of winning, (e.g. a single 1st prize winner and five 2nd prize winners, etc.) It can also be used for the random ordering of participants (e.g. to determine a draw for an athletic

R. Baumgart (Ed.): CQRE'99, LNCS 1740, pp. 101–108, 1999.
© Springer-Verlag Berlin Heidelberg 1999

competition.) The server allows an initiator of a lottery session to determine its purpose and design its rules.

In designing such a lottery server, security requirements and applicability features often come in conflict with each other. We have carefully designed the lottery scheme that the server adopts to meet the following requirements from the aspect of both security and applicability.

Fairness: Users can be assured that outcomes are generated in a fair manner, under reasonable assumption.
Verifiability: The server provides users with verification means to detect any inconsistencies made during the session.
Simplicity of procedures: The round complexity of the scheme is kept as low as possible.
Robustness: The server does not fail to conduct an outcome, even in the presence of lazy players, i.e. those who for some reason fail to participate as they should.
Flexibility: The server can handle many types of lotteries that would be carried out in general.
Descriptive Menu: The server provides a simple template menu by which the initiator can describe in detail the characteristic of the lottery desired.
Feeling of reality: A consideration must be taken in designing appropriate human-interface to increment a sense of "reality" to the act of participation.

We believe thus constructed lottery servers will be useful on the Internet in a number of ways: an individual might use one, for example, to choose from friends distributed over the net persons to whom to give spare movie tickets; managers of public facilities might use one to choose among applicants for use of those facilities; newly developed electronic auction systems or electronic voting systems might use one to hold a lottery in case of a tie; etc.

The rest of the paper is organized as follows: in Section 2, we describe the basic process involved in a lottery session, and in Section 3, we present the implementation more specifically.

2 Lottery Scheme

The lottery scheme that we have employed involves two kinds of users: a *dealer*, who initiates a lottery session on the lottery server, and *players*, who access the server and participate in the lottery session.

On the other hand, the lottery server manages and carries out multiple lottery sessions initiated by the users. More specifically, the trustworthy server:

- provides users a means to start new lottery sessions and become dealers,
- maintains the secrecy of initial values of each session until its closure,
- provides users a means to participate in sessions they wish to particpate in,
- on a due date, executes a lottery and computes its outcome, and
- displays the outcome of each executed lottery session in a verifiable manner.

2.1 Outline

The outline of how a session proceeds is as follows:

1. The dealer initiates a lottery session on the lottery server. He describes its design and purpose using a template, then determines an initial value x and submit it.
2. The server, on accepting the request, determines a server's initial value y for the session using a random number generator. The server assigns a unique session ID sid.
3. The server adds to a session list the description of the new session, which is published on the web, together with commitments of the dealer's initial value x and the server's initial value y. Here, the commitments are computed as $H(sid \circ x)$ and $H(sid \circ y)$ using a cryptographically secure hash function[4], H.
4. Each player i chooses the session he is to participate in, and enrolls his name together with a string r_i freely created by himself.
5. On the date of execution, the server computes the outcome from two initial values and strings created by each player. The result of the session is mapped from a hashed value $H(x \circ y \circ r_1 \circ \cdots \circ r_n \circ DESC \circ sid)$, where $DESC$ denotes a string uniquely converted from the description of the session[1]
6. The outcome is published on the web, together with decrypted initial values x, y and strings r_i created by each players.
7. Each players may verify the following: the string he has created is indeed included, the outcome has been correctly computed, and the initial values had been correctly committed, by computing $H(x \circ y \circ r_1 \circ \cdots \circ r_n \circ DESC \circ sid)$, $H(sid \circ x)$ and $H(sid \circ y)$.

2.2 Features

We assume that a cryptographically secure and ideal hash function achieves the following properties:

- *One-wayness.* Given $H(x)$, it is hard to compute x. Moreover given $H(x)$ and any partial string constituting x, it is hard to recover x entirely.
- *Collision-free.* It is hard to find x and x' that yields $H(x) = H(x')$.
- *Random oracle model.* Distribution of $H(x)$ can be regarded as random.

Based on the properties of the hash function, the lottery scheme described above provides the following features:

- Strings chosen by the players *equally* contribute to computing the outcome. The output of an ideal hash function is dependent on each bit of the input. Controlling a part of the input can not control the output.

[1] In the implementation, we added the name of the dealer, a name and a registration number of each participant to the input to the hash function.

- Players are not allowed to gain any unfair advantage. In order for a player to select a string advantageous to himself, knowledge of all the other players'strings and initial values x and y are required. However, public information regarding x and y is $H(sid \circ x)$ and $H(sid \circ y)$, which leaks no information on x and y due to the one-wayness of the hash function. Even if a player colludes with the dealer, the value of y is never leaked beyond the trustworthy server.
- Neither dealers nor the server can alter the initial values without being detected. In order to alter the initial value x by x', or y by y', $H(x) = H(x')$ or $H(y) = H(y')$ must hold if not detected. These collisions are hard to find due to the collision-free property of the hash function.

2.3 Security Enhancement and Its Cost

One possible concern in the proposed system is the centralized power at the server, i.e., it knows all the initial values of the sessions. One way to decentralize its power is to keep some of these values from the server and distribute them among a dealer and/or some of the participants at each session. For example, a dealer may not submit its initial value x in plain text when initiating a session, but instead submit $H(sid \circ x)$ from the beginning. Thus the server is prevented from knowing x. Even the participants, some or all of them, can voluntarily act in setting initial values. That is, they can send the hashed value of a string of their choice, which will be published, before the session begins. They will reveal the string only after the closing of the session. In this case, each participant can be completely assured of fairness, because no cheating is possible as long as he keeps the string to himself.

The biggest drawback to this approach is that both dealer and volunteer participants must be available when computing the outcome. This affects the scheme's robustness, that the session may terminate without an outcome.

We feel that a more suitable approach is to introduce multiple independent entities within a server, who is always assumed to be present in executing a lottery. On initiating a session, each entity generates an initial value and broadcasts its hashed value to other entities. A set of all hashed values can be considered as the server's hashed values, or more conveniently, the hashed value of all hashed values from the entities can be used. This improvement does not cause any change in user procedure. Other conventional techniques on threshold schemes can also be employed.

3 Digilot: WWW Lottery Server

We have constructed a WWW lottery server on which to implement the proposed scheme.

For the implementation, we have specifically designed:

- a template menu which allows a dealer to describe the design and the purpose of his lottery, and
- a lottery engine which specifies the input to the hash function and mapping of a hashed value to the outcome of the lottery.

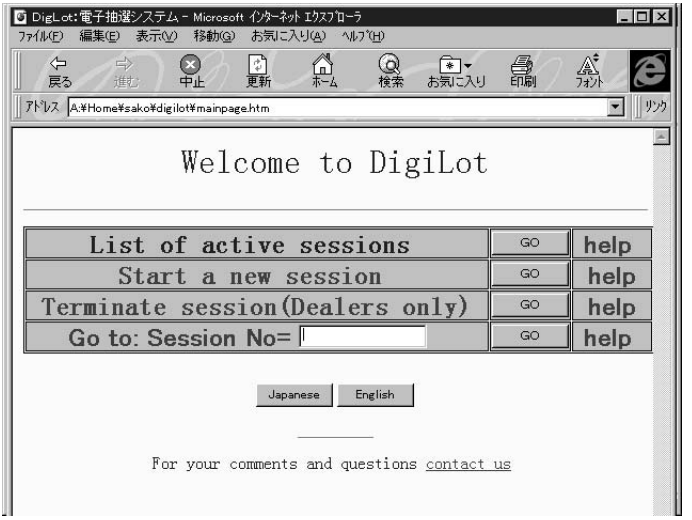

Fig. 1. Main Page of Lottery Server

Starting a new session
A dealer, the person who opens a new lottery session, specifies the following using the template menu:

- the title of the session
- the aim of the session
- participants' qualifications
  Sessions may be either open all or limited to members. In the latter case, a list of members must be given.
- selection field
  Selections can be made from a field comprising participants, limited members, or others.
  In the third case, a list of items in the selection field must be given, for example, spades, hearts, clubs, or diamonds for a card game, or 1-6 for a dice game.
  The second case applies when one needs to select among the members with equal probability, despite possible *laziness* of each member. That is, even if a member fails to participate, he still has a chance of winning (or, similarly, losing) in the lottery.
- outcome to be determined: its item description and number.
  Items can be plural, e.g. one 1st prize winner and five 2nd prize winners.
- opening and closing dates
- date to execute lottery
- an initial value x

Although this template menu does not serve to represent all types of possible lotteries, we believe it compactly describes most of the designs and purposes of lotteries.

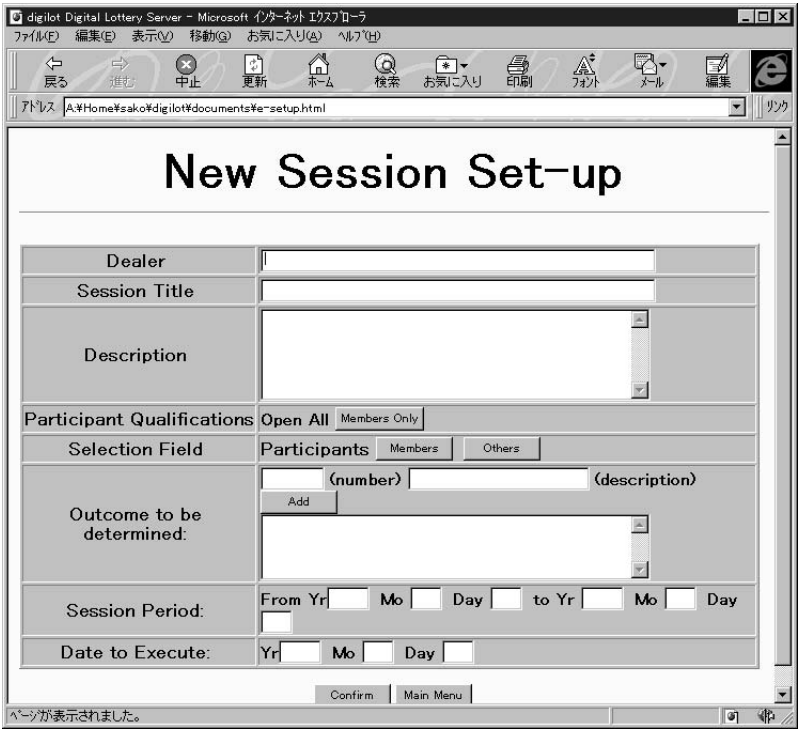

Fig. 2. Starting of a New Lottery Session

When a request for a new lottery session with its description is submitted, the server asks the dealer to choose an initial value x. The server then generates a random string y to be the server's initial value for that session, and assigns a session number (session ID, sid). The description of new session with values of $H(sid \circ x)$ and $H(sid \circ y)$ is published on WWW.

Participating in a session

Each participant specify a session number, either one obtained from a list of public sessions, or one provided by the dealer or another participant. The server displays the aim and description of the session, together with hashed values $H(sid \circ x)$ and $H(sid \circ y)$. By typing in his name and a freely chosen string, he completes the participation procedure.[2]

[2] If only qualifed members are to participate, they need to enter a secret password which is informed by the server beforehand.

On accepting his entry[3], the server displays his registration number with the string he has submitted.

Result of a session
After the closing date, the lottery server computes the outcome and publishes the result. The server provides a verification tool so that all participants can verify that the result is consistent with the specified procedure. The tool also allows users to edit an input to the outcome-computing function. Thus one can observe how an alternative choice of his string would have changed the outcome.

Specification of the lottery engine
The following are inputs to the lottery engine:

- dealer's name
- dealer's initial value and its published hashed value
- server's initial value and its published hashed value
- each participant's name, his registration number, and his string in the order of participation
- a list of items in the selection field in a specified order
- a list of outcome descriptions and their number in the announced order
- session ID

The engine concatenates the above input in the specified order and computes its hashed value. The hashed value is then mapped to a number in a list of selection field items using modular computation. When multiple items are to be selected, the hashed values are sequentially hashed to obtain necessary distinct winning items.

Implementation details
We have implemented a demonstration system which works on Windows 95 and Windows NT server/Workstation with Pentium CPU 100MHz. The system requires a webserver such as Personal Webserver or Internet Information Server, and a Netscape Navigator. The program is written in C and has 6K lines, less than a tenth of which is devoted to describe the lottery engine. For actual use, we are also developing a system that uses Oracle 8 database for the session management, with e-mail services for users to notify the status of one's lottery session.

4 Concluding Remarks

In this paper, we have presented the design and implementation of a lottery server on WWW. This is an actual tool to support a flexible, *universal* lottery,

[3] The current open all session only identifies and distinguishes users by their e-mail address. For more rigorous user identification, use of digital signature techniques is appropriate.

where a user can define and start a lottery session of his purpose. The server provides users a verification means, which helps them to be assured of fairness. Through the use of a trustworthy server that maintains secrets, the scheme does not complicate actual operations or adversely effect the ease of actual use.

References

1. Manuel Blum *Coin Flipping by Telephone.* In *IEEE COMPCON*, pages 133–137. 1982.
2. Oded Goldreich, Silvio Micali and Avi Wigderson *How to Play Any Mental Poker.* In *STOC*, pages 218–229. 1987.
3. David M. Goldschlag and Stuart G. Stubblebine *Publicly Verifiable Lotteries: Applications of Delaying Functions* In *Financial Cryptography*, 1998.
4. Douglas Stinson *Cryptography–Theory and Practice–*, CRC Press, 1995.
5. Ronald Rivest *Electronic Lottery Tickets as Micropayments* In *Financial Cryptography*, 1997.
6. Eyal Kushilevitz, Yishay Mansour, and Micheal Rabin *On Lotteries with Unique Winners* In *SIAM Journal on Discrete Mathematics*, Vol.8, No.1, p.93-98, 1995.