Electronic National Lotteries

5 authors, including:

Elisavet Konstantinou
University of the Aegean
49 PUBLICATIONS **348** CITATIONS

SEE PROFILE

Vasiliki Liagkou
University of Ioannina
39 PUBLICATIONS **87** CITATIONS

SEE PROFILE

Paul Spirakis
University of Liverpool
416 PUBLICATIONS **3,721** CITATIONS

SEE PROFILE

Yannis C. Stamatiou
University of Patras
92 PUBLICATIONS **840** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Project Secure and Privacy-preserving Computation View project

Project FOCUS (Foundations of Dynamic Distributed Computing Systems) View project

Electronic National Lotteries

Elisavet Konstantinou[1,2], Vasiliki Liagkou[1,2], Paul Spirakis[1,2],
Yannis C. Stamatiou[1,3,4], and Moti Yung[5]

[1] Research and Academic Computer Technology Institute,
P.O. Box 1122, 26110 Patras, Greece
[2] Department of Computer Engineering and Informatics,
University of Patras, 26500 Patras, Greece
{konstane,liagkou,spirakis}@ceid.upatras.gr
[3] Department of Mathematics, University of the Aegean,
Karlovassi, 83200, Samos, Greece
stamatiu@aegean.gr
[4] Joint Research Group (JRG) on Communications and Information Systems
Security (University of the Aegean and Athens University of Economics and Business)
[5] Computer Science, Columbia University, New York, NY, USA
moti@cs.columbia.edu

Abstract. We describe the design and implementation of secure and
robust protocol and system for a national electronic lottery. Electronic
lotteries at a national level are a viable cost effective alternative to me-
chanical ones when there is a business need to support many types of
"games of chance" and to allow increased drawing frequency. Electronic
lotteries are, in fact, extremely high risk financial application: If one dis-
covers a way to predict or otherwise claim the winning numbers (even
once) the result is huge financial damages. Moreover, the e-lottery process
is complex, which increases the possibility of fraud or costly accidental
failures. In addition, a national lottery must adhere to auditability and
(regulatory) fairness requirements regarding its drawings. Our mecha-
nism, which we believe is the first one of its kind to be described in
the literature, builds upon a number of cryptographic primitives that
ensure the unpredictability of the winning numbers, the prevention of
their premature leakages and prevention of fraud. We also provide mea-
sures for auditability, fairness, and trustworthiness of the process. Besides
cryptography, we incorporate security mechanisms that eliminate vari-
ous risks along the entire process. Our system which was commissioned
by a national organization, was implemented in the field and has been
operational and active for a while, now.

1 Introduction

Generating numbers for the support of lotteries is a utility that needs to pro-
duce *unpredictable numbers* with additional protection (e.g., against premature
disclosure) and a secure system supporting various sorts of fraud prevention
mechanisms throughout the entire lottery process. A huge amount of money is
at stake for the lottery operator in case of malicious intervention in the number

drawing mechanism or anywhere else in the system (e.g., introducing a winning coupon "after the fact"). Thus, what we need is assurances of robustness that will make sure the desired unpredictability properties while facing a new set of attacks perhaps by insiders or other parties with access to various partial relevant data. In the real world, achieving these aims can be traditionally accomplished with mechanical lotteries performed live on TV with a certified auditor present. However, due to recent business needs, the use of such lotteries is not suitable. For instance, there may be a requirement for very frequent drawings (every 5 minutes in KENO-like games) or drawings that should be accomplished within a very short time interval. In such cases the use of electronic lotteries is inevitable. Also, dynamically games may change as new games are introduced and the cost of new mechanical device for each game is quite large, whereas an electronic device producing random bits is much more easily adapted to new games, merely by re-interpretation of the random stream of bits. Regarding the consumer side, auditability is required and acceptable assurances are required to make sure that the lottery result is fair. Assuring that the process is indeed a "game of chance" may also be required by regulation. This is in contrast with "Internet casino sites" that rather than playing a fair game (with some agreed upon bias, perhaps), might secretly study specific user behavior and tune their games accordingly, to maximize profit.

Related Work

A number of designs that seem to lack scalability (which is a must in a national lottery game) are available in the literature. In [9], a lottery protocol is presented that allows the support of e-casinos with secure remote gambling. An interesting feature of the protocol is that the initial randomness (seeds) is chosen through the collaboration of two or more players in such a way that the final choice is essentially random to all of them. The protocol also includes various auditing functions that build trustworthiness between the casino owner and the players. However, the need for collaboration of players and the overhead in the required protocol steps make it rather unsuitable for large scale electronic lotteries with a large expected player participation and a requirement for fast operation. The paper also present some interesting practical issues pertaining to the design and operation of remote electronic lotteries. In [6], another electronic lottery protocol is proposed based on the concept of *delaying functions*. A delaying function is a function that cannot be computed in time less than a predetermined time limit. Although these functions ensure fairness and public verifiability of the whole process, the time required for the verification is as long as the time to compute the function and this can be unacceptably low in applications where the drawings are frequent. Also, the status of the best time to compute a function may change as our knowledge changes (due to lack of solid lower bounds in complexity theory), thus the delay may not be robust enough over time (one can, of course, always adjust the parameters to handle algorithmic advances but it requires awareness of the advances on the designers' side). In addition, the protocol puts an upper bound on the number of lottery coupons one can buy

which is unacceptable in a scalable nation-wide design. In [24], a protocol based on a bit-commitment scheme where a secret (the winning numbers) is committed to and can be read by the lottery players only after a predetermined amount of time. This still can be unacceptable in applications that require frequent drawings and unsuitable for large scale lotteries. In [22], a protocol for Internet based lotteries is presented. It generates the winning numbers using the played coupons with cryptographic primitives such as hash functions. This protocol is actually incorporated in a tool that supports user-initiated drawings and verification of the generation process. However, this cannot be used in large-scale lotteries with many participating users where each user is part of the auditing function. In [13] a protocol is presented that involves a bit-commitment scheme on the part of the electronic lottery so that the chosen seed cannot later be changed. It allows users to participate in the drawing of numbers indirectly by incorporating their numbers in the chosen seed. A scalability issue with this protocol is that it requires some computation steps that need to be performed by the players so that they can decide whether they win or not.

In [26], another protocol is presented whose main features are the preservation of the anonymity of the players and the existence of a mechanism for paying the winners. However, this protocol still requires the users to participate in the identification of the winners and, also, adds the complication of payments that usually is not an issue since the winners can claim their prizes later at the lottery organization or some designated bank out of band. In essence this protocol is about the front end user handling over the Internet, whereas we concentrate on the back end support (that can be augmented to include a front end over the Internet). The protocol presented in [12] uses as primitives a bit-commitment scheme and a hash function and it is suitable for a large-scale Internet operation (but it is essentially a protocol for Internet betting rather than lottery). It attempts to minimize the transactions between involved parties for security and efficiency reasons. The protocol mainly addresses Internet security and it focuses on resolution of conflicts among parties as well as prevention of collaboration among them towards forgery. Thus, it does not focus on the crux of the system, namely on the number generation process.

Finally, practical ideas and techniques on the frequently neglected but highly critical issue of generating and managing securely the "true" randomness necessary for various components of cryptographic applications can be found in [10] and the references contained therein, as well as Chapter 10 of the book of Ferguson and Schneier [7].

Our Design

The protocol we describe in this paper has been implemented in a real nationwide electronic lottery environment that requires *frequent* drawings per day with *strict* drawing times. Thus, the large number of expected players and the hard timing constraints essentially preclude the explicit participation of users in the number generation and winner identification processes. Our design is scalable, since it is

a nationwide application that is alive. In contrast with all previous approaches, to build up people's trust in the electronic lottery we have done the following:

1. We focused on the core number generation process and our protocol incorporates several interacting cryptographic primitives that ensure the credibility of the process. Each element of the generation combines various independent technologies concurrently to assure **cryptographic robustness**.
2. We provided protection against various manipulations of the process assuring the necessary security level, so as to avoid a huge financial loss for the lottery organization if one manages to interfere with or prematurely learn the process. We used bit commitments, signatures and encryptions to protect various pieces of information and bind the results to the bidding data. We protected the process against premature or future manipulation by binding it to the system's state via a process we call **state stamping.**
3. We designed extensive real-time auditing facilities. We made sure that some independent processes will monitor/ audit other critical components as much as possible, so that actions can be verified after the fact due to logging, signing, etc.
4. We took into account performance (time constraint) requirements.
5. We incorporated security mechanisms (since cryptography alone is never a complete security solution). We isolated parts of the network and employed network security tools, and designed for independent actions and logs to take place. We also took care of physical and operational security.
6. In addition, since delays or cancellations of the drawings may damage the reputation of the lottery organization, there is a provision for replication (fault tolerance) at all system levels (hardware and software) in order to increase reliability and achieve high-availability.
7. We assured modularity, enabling the protocol to be suitable plug-in component in, e.g., as part of an Internet lotteries that also take care of many interacting parties such as banks, lottery organization, coupon sellers, etc.

In what follows, we will describe the protocol by describing its basic primitives and the way they interact with each other. In Section 2 we motivate and discuss the requirements posed by an electronic lottery design. In Section 3 we provide a design proposal to meet the set requirements. In Section 4 we provide a high-level functional description of the components and the drawing protocol used by the electronic lottery. In Section 5 we provide the details of the implementation of each of the components. Finally, in Section 6 we summarize the main feature of the electronic lottery protocol and discuss possible practical improvements and extensions of a real implementation of the protocol.

We believe that this work will serve as a starting point for triggering thoughts and proposals on how application-driven protocols for the production of random numbers should be designed, in terms of security, robustness and efficiency, for use in other electronic lottery settings and similar scenarios "where true pseudorandomness counts."

2   Operational Environment and Requirements

2.1   The Environment

The operational context in which the electronic lottery operates is the following (see Figure 1). The players submit their coupons (on which they mark their number choices) at one of 6000 lottery agencies. Then the agencies send (via telephone lines) the coupon data to the central computer of the lottery organization where the support software stores them in a special coupon file. Exactly 5 minutes before the predetermined time of the next the drawing, no more coupons are permitted in the system and the bidding stops. Note that in the future the agencies and phone lines can be augmented with Internet servers that collect coupons from individual Internet users in another lottery distribution channel (the rest of our mechanism constituting the back-end and result publication component stay as is).

At this stage, the electronic lottery is initiated by the central computer and produces the numbers at the exact time of the drawing, sending them over to the TV channel as well as to the central computer. Every element in the system has an independent backup and many channels are replicated. Various auditing and monitoring is performed on-line. Reliability is another issue we provide, which implies replication of components.

In more details, due to high-availability and security requirements, the electronic lottery is composed of three components: two computers, the generators, interconnected in a master-slave configuration so as the slave automatically takes over in case of failure of the master plus one computer, the verifier, that acts as an intermediate between these computers and the central computer. The generators are totally isolated from the environment and they only communicate with the verifier, to which they send the numbers plus other auditing information. Then the verifier checks the integrity of the drawing and if all checks are successful then the numbers can be safely transmitted to the central computer. Moreover, for auditing purposes, the generators send the numbers also to a printer and a computer monitor placed at a supervisor's office for cross-checking and they store them on CD-recorder and hard disk for later verification.

Finally we assure physical security. The electronic lottery system (i.e the two generators plus the verifier) is enclosed in a shielded room with biometric access control system which is under a 24-hour per day camera surveillance (and there is a plan for periodically updating the physical protection).

2.2   Sets of Requirements for the Electronic Lottery

The great financial risks involved in the building and operation of the electronic lottery necessitated a very careful consideration of all possible security aspects as well as environmental factors that may disturb the normal operation of the lottery (e.g. power and network failures).

Generally speaking, the main "operational requirements" placed on a system capable of supporting any lottery can be summarized in the following:
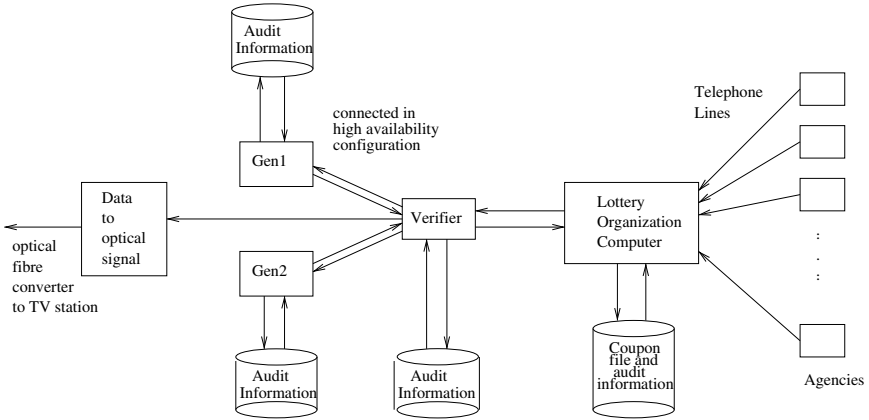
Fig. 1. An overview of the system

1. The produced numbers should obey the <mark>uniform distribution</mark> over their intended range.
2. It should be infeasible for anyone (either the lottery operator, the lottery designers or the players) to <mark>guess the next number of the lottery, given all the lottery history, with chances better than if the next number was to be chosen at random.</mark>
3. It should be infeasible for anyone (either the lottery operator, the lottery designers or the players) to interfere with the drawing mechanism and with the choice of winners (even after the drawn numbers are known and published). In case such an interference eventually occurs, then this fact is detectable.
4. The drawing mechanism should be designed so as to obey a number of standards and, in addition, there should also be a process through which it can be officially certified (by a lottery designated body) that these standards are met by the chosen mechanism.
5. The drawing mechanism should be under constant scrutiny (by a lottery designated body) so as to detect and rectify possible deviations from either of the above requirements or potential tampering with it.
6. The details of the operation of the drawing mechanism should be publicly available to people so as to ensure their trust and interest in the games. In addition, this ensures a publicly open lottery auditing protocol (which may be required by regulations).

Ensuring these requirements is accomplished (under a reasonable physical model) through the use of the traditional drawing mechanism of balls that are shuffled by some random physical process and then chosen from within an urn, where this entire process is performed publicly and with auditors present and is pre-authorized by a state authority. This appears to achieve all requirements set above plus people's satisfaction that the whole mechanism is trustworthy and fair to them.

The situation changes dramatically, however, if business requirements necessitate the use of electronic means as is our case. There are many reasons why one

would prefer this option over the traditional drawing mechanism. For example, to increase players' interest for participation in lottery games, many lottery organizations (such as the one using the protocol we describe in this paper) desire to perform many drawings each day instead of the traditional end-of-the week drawing. This is done in KENO-like games where drawings may be performed as frequently as every 5 minutes, increasing the lottery organization's chances for profits. This is because people play more frequently since they are attracted by the fact that if the do not win at the current draw, they may well be winners at the next draw in a few minutes. Another reason for not using the traditional drawing mechanism could be the fact that the lottery organization desires the introduction of a variety of games with numbers drawn in different ways (e.g. repetitions allowed or not, different ranges, etc.), a thing that would require building many drawing machines with high cost and construction time. Indeed, in our case, the lottery organization desired the introduction of two *different* games, one with the selection of three numbers from 0 to 9 (repetitions allowed) and another one with the selection of 5 numbers from 1 to 35 (repetitions not allowed). Also, there was a requirement for four electronic drawings per day at predetermined times. In addition, for publicity and publication (as well as marketing) benefits, each drawing should be sent over to a private TV channel that displays the numbers in real time at the lower third of the screen. The publication method integrity is assured on-line. The numbers are also sent over to the computer that stores the played coupons so that the winner can be selected and various statistics calculated.

Finally, a very important requirement with regard to all the entities involved, was the *high-availability* of the electronic lottery. No delay or cancellation of a drawing was acceptable due to failures of the electronic lottery as this would jeopardize the success of the new games as well as the reputation of the entities.

From the above considerations, it is clear that traditional drawing mechanisms are costly and cumbersome, if not impossible, to use and support the business operational requirements.

Taking into account the above discussion on the "security and safety requirements" of the whole system, the security requirements for our application can be described more precisely as follows:

1. Confidentiality: Information should be disclosed only to the intended recipients and no leaks of information occur before predetermined time points. Confidentiality can be achieved through encryption methods as well as secure random number sources that prevent estimation of their evolution.
2. Integrity: No unauthorized changes should be made, both in stored and transmitted data. Integrity of data can be achieved through the use of computation of hash and MAC functions.
3. State stamping: The lottery outcome is a function of a given state representing all the coupons of the current drawing and the internal (randomly chosen) state of the lottery mechanism, some of it secret. The system should stamp the state using cryptographic tools so that no future modifications are possible without detection.

4. Availability: The system should be ready for operation any time it is needed and should not turn down authorized service requests (subject to given service/ performance requirements). Availability can be achieved through component and data path replication.

5. Accountability: All access to, or modification of, specific information in the system should be detected and, possibly, traced to specific sources (this also include identification schemes). Non-repudiation of an action is the lack of capability of an actor to deny an action, and as a security property, it is closely related to accountability. These requirements can be achieved using mechanisms of electronic signing and commitment.

In the next section we will provide the design of the electronic lottery and justify the design choices.

3   Design Considerations

In this section we will discuss the components of the solution to the drawing process. We explain how we met the security requirements, confidentiality, state stamping, integrity, availability and accountability described above. More specifically, we describe the specific electronic lottery choices we made, indicating for each of them the specific requirement it meets. We employed numerous cryptographic primitives and protocols at the low level of the generation making it robust according to the requirements.

3.1   Randomness Sources

One component of the confidentiality requirement concerns the use of a good source of random numbers. There are three approaches, other than the traditional one, for producing sequences of numbers: (i) Using an appropriate algorithmic scheme – *pseudorandom number generator,* (ii) Using some physical process such as, for example, semi-conductor noise – *truly random number generators,* and (iii) Using a combination of (i) and (ii). There is much debate going on as to which is the best approach. Approach (i) has been subjected to the criticism, that an algorithm has a limited, although huge, number of possible states and, as it follows a well defined set of steps, it may be amenable to some clever educated guess attack. Although the introduction of, the so called*, cryptographically secure* pseudo-random number generators can handle this criticism the fact remains that an algorithm is deterministic and, thus, its output can always be guessed in principle, given the initial state. Approach (ii) on the other hand receives the criticism that physical processes often obey specific distribution laws that may enable one to limit the range of possible future evolutions of a generator based on them. In addition, physical devices often malfunction or deviate from their initial statistical behavior depending on environmental factors such as temperature, humidity, magnetic fields etc. This may, also, enable one to easily interfere with the number generation process. In addition, physical randomness,

if not backed-up correctly (for auditing purposes), is hard to reproduce and, thus, check appropriately. Finally, according to approach (iii), software-based pseudorandomness and truly random generators are both used (as a hybrid) in a way that amplifies their advantages and diminishes their disadvantages. It is exactly this approach that, we believe, is the most beneficial for designing a system for the needs of a lottery and this is the approach we followed and describe below. We also replicate generators of both types, and combine streams of independent generators to achieve cryptographic robustness, in case some method or technology fails.

Of course, approach (iii) alone is not, by itself, sufficient to guarantee that the lottery system obeys all the requirements. We still have to consider the exclusion of many possible threats such as post-betting, malicious intervention, system observation, system access protection and many more.

3.2   Seed Commitment and Reproduction of Received Numbers

An issue that arises when a drawing is performed is whether the seeds claimed to have been used by the generation process were actually used. Ensuring that the seeds were actually used, entails the use of a bit-commitment protocol and is related to the integrity and accountability requirements. The commitment is performed by the number generator and the related information is transmitted to the verifier that performs the necessary checking. This commits to a verifiable state, yet keeps the confidentiality of the seed to prevent premature leakage.

3.3   State Stamping: Prevention of Post-betting

A major requirement from the random number generation protocol was to be in position to detect post-betting, i.e. to detect whether a coupon was inserted into the coupon file after the current drawing is closed. This meets the integrity and accountability requirements as it guards against illicit coupon file modification. In addition, when this especially bad situation is detected, the protocol should be terminated immediately and report it, essentially canceling the current drawing. One way to detect post-betting is to use a fingerprint (hash value) of the coupon file after the bidding time is over and check that it still has the same value. If not, an error condition is raised and a supervisor is notified.

3.4   Seed Processing

We used the Naor-Reingold pseudorandom function (see [20]) for processing the combination of the seeds that are obtained from the physical random number generators with the hash value of the coupons file. The NR function is initially seeded with a strong random key. With the use of the NR function the resulting processed seeds is made not directly dependent on the on-line drawn physical bits, an act which guards against malfunctioning of the physical randomness sources by "rectifying" deviations (and makes the process independent of the manufacturers of the physical devices).
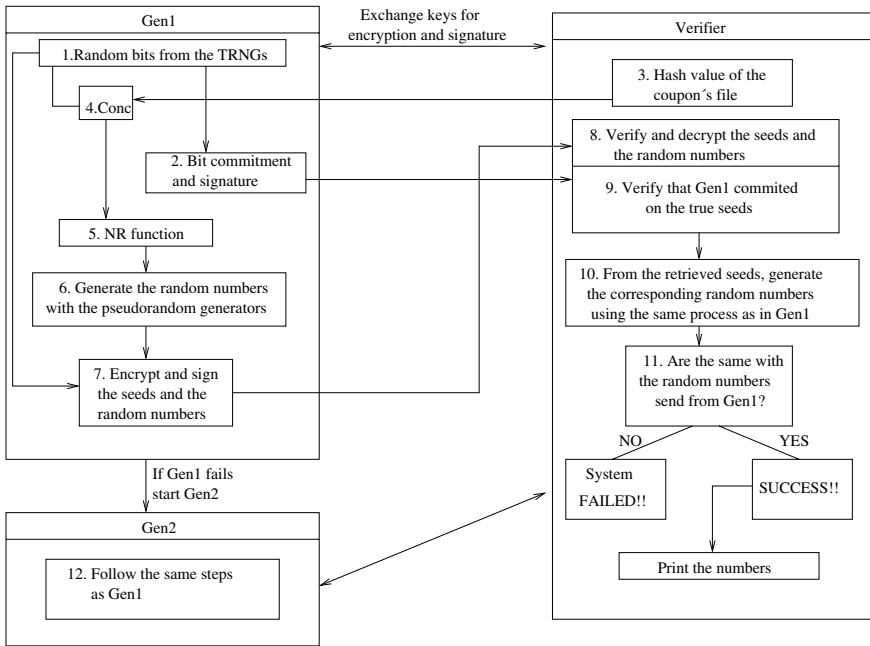
Fig. 2. The architecture of the random number generation system

3.5 Signing and Authenticating

To boost confidentiality and accountability, each time a drawing is performed the seeds and the produced numbers sent over from the generation source, should be signed by the source (using, e.g., a public key cryptographic scheme) and verified by the recipient. In this way, the source of the drawing is authenticated and the drawing can be considered valid.

4 A High-Level Description of the Protocol of the Electronic Lottery

Before we detail each component of the protocol, it will be useful to give a high-level description of these components as well as their interaction as summarized in Figure 2. The protocol is based on two basic interacting agents: the Generator and the Verifier. The Generator is replicated for high-availability purposes. (Of course, other components can be replicated as well, but this is easy to do since they are mainly general purpose computers whereas the generator is a complex and highly secured component that we replicated.) First, the Generator and the Verifier execute a key-exchange protocol in order to end-up sharing a secret key to enable secure communication between them. They also generate a private/public key pair for signature purposes. At this point, the Generator starts idling and waits for a drawing initiation signal from the Verifier. The Verifier knows the times of the daily drawings since they are communicated to it by the central

computer. Only coarse time synchronization is needed since there is enough time for all time-based operations in the system, since the mechanism operates four times a day. The following are the steps in generation (see Figure 2):

1. Upon initiation, the Generator first draws a sequence of truly random bits as seeds from a set of physical random numbers sources (reseeding occurs at sufficiently frequent time intervals).

2. Then the Generator executes a bit-commitment protocol on the seed bit sequence and signs it. The packet that results from these actions is sent over to the Verifier. As a result the seeds are committed to and the commitment is signed.

3. The Verifier, in turn, gives the Generator the hash of the file of coupons.

4. Next, the Generator mixes the random bits produced by the physical random number generators with the hash value of the file containing all the coupons played up to the allowed time (contained in the coupon file) in order to inextricably bind together the seed with the coupon file. This mixing can be effected by simply concatenating the random bits with the hash value given by the Verifier, essentially "freezing" the coupon file (this is the state stamping for post-betting prevention). Since the randomness is committed to and signed (and verified) and the coupon file hash is a commitment to the file, this is a state stamping which is present and logged at the Verifier.

5. The first portion of the initially produced (i.e. bits from the physical random-ness s ources with concatenated hash value of the coupon file) bit-sequence is then fed through the Naor-Reingold (pseudorandom) function as a post-processing precaution to further decouple any direct biases of the random sources (provided by external manufacturers) from totally influencing the randomness (based on separation of duties principle). The result is inter-preted as a bit stream.

6. The bit stream result of the previous stage is XORed with a second por-tion of the initially produced bit-sequence (to mix a physical stream and a pseudorandom stream for cryptographic robustness). Then, this final bit sequence is used as the seed to a set of software based pseudorandom gener-ators, in order to stretch the seed and assure that enough random numbers are generated. In our protocol we are currently using algebraic as well as block cipher based generators. We employed the algebraic pseudorandom number generators RSA and BBS (which have a proof of security in the lit-erature) and two generators based on the block ciphers DES and AES. These generators can operate alone or in various output combinations. Although we have used the specific generators (as they are widely known or accepted within the cryptographic community) any number or type of secure software random number generators could be used instead. Note that basing our final generation on variety of functions adds to cryptographic robustness.

7. Then the Generator executes a bit de-commitment (opens the initial phys-ical random seed bits). The seed and the numbers are encrypted and their encrypted form is signed. The packet that results from these actions is sent over to the Verifier and the Generator stops.

8. The Verifier first authenticates the packet (using the public key of the Generator), decrypts the received packet and recovers the numbers plus the seed presumably used by the Generator to produce these numbers.
9. Then, the Verifier checks that the Generator has committed to the sent seed.
10. Then, the Verifier checks that the seed was used properly by repeating the same generation process used by the Generator and interpreting the outcome in the range of numbers to announce.
11. If the produced numbers match the numbers sent by the Generator then the drawing is considered successful and the drawing is completed by announcements of the winning numbers. This check against the commitment and the hash of the file of coupons verifies that no interference occurred in the drawing process in the Generator. If the numbers do not match the numbers sent, then an alarm is raised, the drawing is canceled and the Verifier initiates it again (or some other action is taken to assure integrity and continuity). In this way the Verifier acts as an auditor of the entire drawing process.
12. If the system fails the Verifier activates the second Generator, that repeats the process of the first one.

Finally, we also included a provision for *on-line statistical testing* that estimate the entropy of sources and long sequences of (i) the produced random numbers, and (ii) the hardware random number generators. Algorithms implemented in software need only be checked once, at the system installation phase (assuming the entire system is not being changed since no one gets access to it). However, given the fact that physical random number generators are vulnerable to environmental conditions or even subject to aging and physical malfunctions, one can never be certain that they are operating properly once installed on a computer. A very informative discussion on testing (on-line as well as off-line) physical random number generators can be found in [23] where a number of tests are prescribed for such generators able to meet their special requirements. These considerations discussed in that paper are already incorporated in the German AIS 31 national standard (as well as prerequisite for device approval) for testing physical random number generators.

5   Implementation Choices

5.1   Randomness Sources

The software random number generators. Our decision to incorporate several different algorithms for the generation of the numbers was to base the whole design on generators relying on different principles of operation and security so as to increase the difficulty of attacks aiming at guessing the number sequence. Different security principles imply that an attacker would face more difficulties in guessing as it would be necessary to break all of these principles. All the cryptographic primitives that will be described are fully reconfigurable in terms of their key sizes (i.e. the sizes modifiable at the implementation level). These sizes can be changed sufficiently frequently, depending on cryptanalytic advances.

We included two algebraic generators. One of them, BBS was proposed by Blum, Blum, and Shub (see [3]) is one of the most frequently used cryptographically strong pseudorandom number generators. The second generator involved in the random number generation protocol is the RSA/Rabin generator and it is based on the RSA function (see in [1]). These works have shown proof of security, but these are typically too slow in many applications, but we had enough time to include them in our mix.

The other two secure generators we used are based on the encryption algorithms DES and AES which are more often used. We used the implementations provided by version 2.5.3 of the mcrypt library (available at [18]). The key sizes were 64 bits for DES (56 bits effective) and 192 bits for AES. In order to have number generators from the encryption algorithms, DES and AES are used in their CFB (Ciphertext Feed-Back) mode and they operate as follows: an initial random vector is constructed from a seed processed as described in Section 4 and then DES and AES are invoked as many times as the bytes we wish to generate. In particular, every time the encryption function is called, a byte with all zeros is encrypted (always in CFB mode, which means re-encrypting the ciphertext) and the encryption result is the output of the generator. The cryptographic strength of the two generators is based on their encryption counterparts assuming the entire block is unpredictable.

We further employed two techniques that are used to fortify and increase variability of weak generators (even though our generators are strong). One of them employs two shuffling algorithms for combining the output of the random number generators: *Algorithm M*, proposed by MacLaren and Marsaglia [15] and *Algorithm B* proposed by Bays and Durham [2]. Algorithm M takes as input two sequences X_n and Y_n, and outputs a more random sequence. The algorithm actually is shuffling the sequence X_n using elements of the sequence Y_n as indexes into the sequence X_n. Thus, the elements of the new sequence are the same with those of X_n but in different order. Algorithm B is similar to M, but it requires only one sequence as input. The output is, again, a shuffled instance of the input. Both algorithms are described in detail in Knuth's book [11].

Another technique for achieving extended variability in the lottery is to combine the four generators (which are viewed as "independent functions") using the bit-wise XOR operation, and to allow the protocol to swap, periodically (according to some predetermined internal schedule), to different sub-set combinations of the generation of the random numbers.

Physical random number generators. The seeds of any software random number generator must, eventually, be drawn from a physical source of randomness. After considering the various physical sources of randomness within a computer (e.g. /dev/random in LINUX, fluctuations in hard disk access times, timing crystal frequency jitter, etc.) and evaluating the trade-off between easiness in using and quality of output, it was decided to use commercial hardware generators known to pass a number of demanding statistical test (e.g. DIEHARD). Moreover, it was important to include more than one physical sources (with outputs XORed) as it is not uncommon to have, after some time, harmful devi-

ations in the physical characteristics of the devices from which noise is drawn. These fluctuations, in turn, may cause the appearance of detectable biases in the produced number sequences. XORing, however, a biased physical source with a good one helps decreasing the bias.

The component of the implementation responsible for physical randomness generation is actually comprised of three separate hardware-based random number generators: (i) One based on the phase differences of the clocks of the computer's motherboard. These differences are tapped by the function VonNeumannBytes(), written by Adam L. Young, which produces a stream of random bytes based on these phase differences. As their authors state, this function is based on truerand() by M. Blaze, and J. Lacy, D. Mitchell, and W. Schell [14] (ii) A commercial device placed on one of the ISA slots of a computer that produces random bits on demand via appropriate system calls: this device is the ZRANDOM random number generator built by of the German company Westphal Electronics (for product overview, consult [25]), and (iii) The commercial device SG100, which is a hardware random number generator connected to the serial port of the computer. It is provided by the Swedish company Protego Information AB (for product overview consult [21]). We used three sources of physical randomness for increased security and in order to guard against malfunction of any one of them.

5.2   State Stamping

We needed to assure that the drawing is based on a given random seed independent of the coupons, yet that no modification of state would be acceptable. A simple way to meet this requirement was to mix the coupon file as is with the truly random (and committed to) seeds drawn from the hardware devices and *then* drive the software generators. As the coupon file is, generally, too big to be combined in a usable way with the seeds drawn from the physical random number generators, we used its much smaller hash value instead (see Figure 2, Gen1). The hash function we used is RIPEMD-160 (see [5]). The Verifier can later check that the right random bits (committed to earlier) were used in this mixing. The commitment makes the mixing non-malleable in the sense that the system's state cannot be changed given the record at the Verifier.

5.3   Seed Processing

As we mentioned above (Section 3.4), we used the Naor-Reingold function, or NR for short, for processing the seeds to assure that whatever biases still exist, a pseudorandom function will process the random seed. (This assumes that we made sure at the start that seeding the NR function key is done with very strong random bits). The seed processing via a pseudorandom function seed can be common to the Generator and the Verifier who both possess the NR key. The Verifier can privately make sure the function was applied correctly. This check can also be done via a non-interactive zero-knowledge proof. We further note that an alternative approach to this stage can be the use of the more recent "verifiable

(pseudo)random functions" as defined in [19], where audit of the correctness of a pseudorandom function can be checked as a public verification utility.

Regarding the NR function, a key for it is a tuple $\langle P, Q, g, \boldsymbol{a} \rangle$, where P is a large prime, Q is a large prime divisor of $P-1$, g is an element of order Q in Z_P^* and $\boldsymbol{a} = \langle a_0, a_1, \ldots a_n \rangle$ is an uniformly distributed sequence of $n+1$ elements in Z_Q. For every input x of n bits, $x = x_1 \ldots x_n$, the NR function is defined as:

$$\tilde{f}_{P,Q,g,\boldsymbol{a}} = (g^{a_0})^{\prod_{x_i=1} a_i} \bmod P.$$

In our implementation, the size of P is 1000 bits and the size of Q is 200 bits. Note that we applied a pseudorandom function for this both randomizing the result and so that this serves as a commitment that based on its inputs its outputs can be checked if and when an internal audit is required (one can verify the output based on given inputs, directly relying on the security of the verifier. As mentioned above, alternatively, audit can be achieved in a zero-knowledge fashion, given the NR function's public description). Since the NR function is pseudorandom, revealing input-output relationships does not hurt its security in this case. The portion processed via a pseudorandom function, is combined with a physical random source portion in order to take advantage of physical randomness as well, in case it is the best source we have.

5.4   Encryption, Signing and Authenticating

The Generator and the Verifier, are each equipped with an RSA key pair. At start-up they construct this pair and exchange their public keys. Then the commitments, the sets of numbers, as well as the seeds, that originate from the Generator are first encrypted and then signed using the shared secret key. This keeps any transmission private within the mechanism as a layer of protection. Before the encryption, however, we used the *Simplified Optimal Asymmetric Encryption Padding*, or SAEP [4]. With the SAEP protocol, a padding on the bits of the message packet is performed aiming at achieving semantic security and chosen ciphertext security (in the random oracle model). Note that typically, a hybrid encryption is used in applications, but given our performance requirements and the specific nature of our messages within the application, we can afford using public key encryption. A possible key size for the encryption is 1000 bits and for the signature 2000 bits. The number of zeros in the SAEP protocol can be equal to 100 and the random number required can, also, have 100 bits. Note that signatures are performed over committed encrypted (thus random) values. The Verifier, after receiving the packet, decrypts it using the same secret key and verifies that it originated from the legal Generator. This avoids the risks associated with authentication through, e.g., IP address checking, password phrase exchange etc. The RSA pairs used for signing can be refreshed after a drawing is completed. This is perhaps an exaggerated precaution due to the large key sizes. In general, refreshment can be less granular. Note that a new public RSA key can be certified by being signed with the old keys this achieves forward secrecy (namely, when the system is compromised past signatures are valid).

5.5   Seed Commitment and Reproduction of Received Numbers

In order to ensure that the Generator actually used the seeds it claims to have used for the current drawing, the Generator and the Verifier execute a bit-commitment protocol based on the RSA encryption scheme and the hash function RIPEMD-160. After this processing, the commitment is sent to the Verifier and then the Generator can get the hash of the file of coupons and then the Generator produces and sends to the Verifier the seed and the resulting numbers of the current drawing. Upon receipt of the numbers, the Verifier uses the seed to which the Generator committed itself to in order to reproduce the received drawn numbers. If the reproduced numbers match the ones sent by the Generator, the numbers are deemed legal and are made public. Otherwise, the protocol stops and issues a warning.

6   Discussion

In this paper we have described a general protocol for the support of a national electronic lottery system. To the best of our knowledge this is the first publicly described system focusing on the core number generation and auditing process.

We have argued why electronic lotteries are an exceptionally challenging type of financial applications, and that there are many factors that should be considered for a robust protocol designed to support an electronic lottery. The generation of sequences that are exceptionally difficult to guess is only one such factor, but one should take measures against many possible attacks on the generation as well as on the entire system operation and management process.

References

1. W. Alexi, B. Chor, O. Goldreich, and C. Schnorr, RSA and Rabin Functions: Certain Parts are as Hard as the Whole, *SIAM J. Computing* **17(2)**, pp. 194–209, April 1988.
2. C. Bays and S.D. Durham, Improving a Poor Random Number Generator, *ACM Trans. Math. Software* **2(1)**, pp. 59–64, March 1976.
3. L. Blum, M. Blum, and M. Shub, A Simple Unpredictable Pseudo-Random Generator, *SIAM J. Computing* **15(2)**, pp. 364-383, May 1986.
4. D. Boneh, Simplified OAEP for the RSA and Rabin Functions, in: *Proc. Crypto '01*, LNCS 2139, pp. 275–291, Springer-Verlag, 2001.
5. H. Dobbertin, A. Bosselaers, and B. Preneel, RIPEMD-160: A Strengthened Version of RIPEMD, in: *Proc. Fast Software Encryption 1996*, LNCS 1039, pp. 71–82, Springer Verlag, 1996.
6. D.M. Goldschlag and S.G. Stubblebine, Publicly Verifiable Lotteries: Applications of Delaying Functions, in: *Proc. Financial Cryptography 98*, LNCS 1465, pp. 214–226, Springer Verlag, 1998.
7. N. Ferguson and B. Schneier, *Practical Cryptography*, John Wiley & Sons, 2003.
8. M.P Ford and D.J. Ford, Investigation of GAUSS' Random Number Generators, a report prepared for Aptech Systems, Inc., *FORWARD Computing and Control Pty. Ltd.*, NSW Australia, 2001.

9. C. Hall and B. Schneier, Remote Electronic Gambling, in: *Proc. 13th ACM Annual Computer Security Applications Conference*, pp. 227-230, 1997.

10. M. Jakobsson, E.A.M. Shriver, B. Hillyer, A. Juels, A Practical Secure Physical Random Bit Generator, in: *Proc. of the 5th ACM Conference on Computer and Communications Security 1998*, pp. 103–111, 1998.

11. D.E. Knuth, *Seminumerical Algorithms*, Third Edition, Addison-Wesley, 1997.

12. K. Kobayashi, H. Morita, M. Hakuta, and T. Nakanowatari, An Electronic Soccer Lottery System that Uses Bit Commitment, in: *IEICE Trans. Inf. & Syst.*, Vol. E 83-D, No. 5, pp. 980–987, 2000.

13. E. Kushilevitz and T. Rabin, Fair e-Lotteries and e-Casinos, in *Proc. CT-RSA 2001*, LNCS 2020, pp. 100–109, Springer Verlag, 2001.

14. J.B. Lacy, D.P. Mitchell, and W.M. Schell, CryptoLib: Cryptography in Software, in *Proc. 4th USENIX Security Symposium*, USENIX Assoc., pp. 237–246, 1993.

15. M.D. Maclaren and G. Marsaglia, Uniform Random Number Generators, *JACM* **12(1)**, pp. 83–89, January 1965.

16. G. Marsaglia, A current view of random number generators, keynote address in: *Proc. 16th Symposium on the Interface between Computer Science and Statistics*, pp. 3–10, 1985.

17. G. Marsaglia, DIEHARD: A battery of tests for random number generators, available at `http://stat.fsu.edu/~geo/diehard.html`

18. Mcrypt cryptographic library: `ftp://mcrypt.hellug.gr/pub/crypto/mcrypt`

19. S. Micali, M. Rabin, and S. Vadhan, Verifiable Random Functions, in: *Proc 40th IEEE Symp. on Foundations of Cumputer Science*, pp. 120–130, 1999.

20. M. Naor and O. Reingold, Number-theoretic constructions of efficient pseudo-random functions, *Proc. 38th IEEE Symp. on Found. of Computer Science*, 1997.

21. Protego, product information, `http://www.protego.se/sg100_en.htm`

22. K. Sako, Implementation of a digital lottery server on WWW, in: *Proc. CQRE '99*, LNCS 1740, pp. 101–108, Springer Verlag, 1999.

23. W. Schindler and W. Killmann, Evaluation criteria for true (physical) random number generators used in cryptographic applications, in *Proc. 4th International Workshop on Cryptographic Hardware amd Embedded Systems (CHES 2002)*, pp. 431–449, 2002.

24. P. Syverson, Weakly Secret Bit Commitment: Applications to Lotteries and Fair Exchange, in: *Proc. IEEE Computer Security Foundations Workshop (CSFW11)*, pp. 2–13, 1998.

25. Westphal Electronics, product information, `http://www.westphal-electronic.de`

26. J. Zhou and C. Tan, Playing Lottery on the Internet, in: *Proc. ICICS 2001*, LNCS 2229, pp. 189-201, Springer Verlag, 2001.