Quantum Permanent Compromise Attack to Blum-Micali Pseudorandom Generator

Elloá B. Guedes, Francisco M. de Assis, Bernardo Lula Jr.
IQuanta – Institute for Studies in Quantum Computation and Quantum Information
Federal University of Campina Grande
Rua Aprígio Veloso, 882 – Campina Grande – Paraíba – Brazil
elloaguedes@gmail.com, fmarcos@dee.ufcg.edu.br, lula@dsc.ufcg.edu.br

***Abstract*— This paper presents a quantum permanent compromise attack to the Blum-Micali pseudorandom generator whose security is based on the assumption of intractability of the discrete logarithm problem. The proposed attack makes use of the Grover's quantum search extension for multiple solutions and of quantum parallelism to recover the generator's internal state with high probability. This attack compromises the unpredictability of the Blum-Micali cryptographic secure pseudorandom generator, since it recovers all previous and future output, as well as the generator's seed. Compared to the classical equivalent attack, the quantum algorithm proposed has a quadratic speedup, and represents a menace against the security of a pseudorandom generator used in many real-world cryptosystems.**

Keywords*— *Blum-Micali Generator, Quantum Attack, Grover Algorithm

I. Introduction

Randomness is an essential requirement for any well-designed cryptographic application. Session keys, initialization vectors, salts to be hashed with passwords, unique parameters in digital signatures, and nonces in protocols are examples of randomness usage in cryptography [1]. However, random sources, like Geiger counts or radioactivity decay, are not available in all cryptographic systems. In face of this limitation, it is acceptable the use of an algorithmic alternative: the *pseudorandom numbers generators* (PRNGs).

The most useful type of pseudorandom processes updates a current sequence of numbers in a manner that appears to be random. Such a deterministic generator, f, yields numbers recursively, in a fixed sequence. The previous numbers (often just the single previous number) determine the next number:

$$x_i = f(x_{i-1}, \ldots, x_{i-k}). \tag{1}$$

Considering that the set of numbers directly representable in the computer is finite, the sequence will repeat. The set of values at the start of the recursion is called the *seed*. Each time the recursion is begun with the same seed, the same sequence is generated. A common requirement of PRNGs is that they possess good statistical properties, meaning their output approximates a sequence of true random numbers. Those properties can be assessed via statistical tests [2], [3].

An special type of PRNG with unpredictable outputs is the *cryptographically secure pseudorandom number generators* (CSPRNGs), i.e., given n consecutive output bits of a CSPRNG, there is no polynomial time algorithm that can predict the next bit with better than 50% chance of success [4]. Since randomnesss plays a major role in cryptographic systems, it is essential to analyze the vulnerability of PRNGs specially those that are used in real world cryptosystems, i.e., the CSPRNGs.

A family of CSPRNGs widely adopted in cryptography was stamped by Blum and Micali [4]. According to them, each generator from this construction must be composed by a function f, that is a permutation over a domain $\mathcal{D}$, and a binary function. Examples of generators founded on the Blum-Micali construction are Blum-Blum-Shub [5], Kaliski [6], and the Blum-Micali generator [4] whose security is based on the assumptions of intractability of the factoring, elliptic curve discrete logarithm and discrete logarithm problems, respectively.

However, *quantum computers* can implement efficient algorithms to certain problems where an efficient classical algorithm is not known. An example is the Shor's quantum factoring that may break the whole security of RSA algorithm when quantum computers become scalable [7]. Another relevant algorithm, the quantum search, was proposed by Grover [8] and performs a search in an unsorted N-sized database with $O(\sqrt{N})$ cost.

In this paper, it is proposed a quantum permanent compromise attack, based on the quantum search, to the Blum-Micali CSPRNG. Such attack retrieves the internal state of the generator and endanger all future and previous output, compromising the unpredictability. Moreover, this attack has a quadratic speedup to its classical counterpart.

The rest of this paper is organized as follows. In Section II the Blum-Micali generator is described and a simple exponential-order permanent compromise attack to this generator is introduced. In Section III the Grover algorithm is reviewed. In Section IV is introduced a quantum permanent compromise attack to the Blum-Micali pseudorandom generator. In the Section V related works are briefly discussed and in the Section VI conclusions and future work are presented.

II. Blum-Micali Generator

Let p be a large prime and $n = \lceil \log p \rceil$ the binary length of p. The set $\mathbb{Z}_p^* = \{1, 2, \ldots, p-1\}$ stands for the cyclic group under multiplication $\mod p$. Let g be a generator of $\mathbb{Z}_p^*$ and $x_0 \in_R \mathbb{Z}_p^*$. We denote $a \in_R A$ the random choice of an element a out of the set A.

The Blum-Micali generator (BM) is prepared with parameters (p,g,x_0) as previously described where x_0 is the seed of the generator, and p and g are the parameters publicly known. This generator produces pseudorandom bits using an one-way function $x_i = g^{x_{i-1}} \mod p$ over the domain $\mathbb{Z}_p^*$, and a hard-core predicate for the permutation, denoted by δ, as shown above:

$$x_i = g^{x_{i-1}} \mod p \qquad \text{exponential map} \tag{2}$$

$$b_i = \delta(x_i) \tag{3}$$

where δ is a binary function with the following definition:

$$\delta(x) = \begin{cases} 1 & \text{if } x > \frac{p-1}{2} \\ 0 & \text{otherwise.} \end{cases}$$

To illustrate the usage of BM, suppose a generator configured with parameters $(7, 3, 1)$, denoting p, g, and x_0, respectively. The diagram (4) shows the evolution of internal states and the bits outputted during this process.

$$\begin{array}{ccccccccc} 1 & \rightarrow & 3 & \rightarrow & 6 & \rightarrow & 1 & \rightarrow & \dots \\ \downarrow & & \downarrow & & \downarrow & & \downarrow & & \\ 0 & & 0 & & 1 & & 0 & & \end{array} \tag{4}$$

The security of the BM generator relies on the hardness of inverting the one-way function $x_{i+1} = g^{x_i} \bmod p$. This inversion is equivalent to solve the discrete logarithm problem and no efficient algorithm to perform this operation is known in classical computing [9]. In the following subsection we introduce a simple algorithm to attack to the Blum-Micali generator.

A. Permanent Compromise Attack to Blum-Micali Generator

A permanent compromise attack happens to a pseudo-random generator when its internal state is recovered and, in consequence, all previous and future output become predictable. This kind of attack compromises the unpredictability of the cryptographic secure PRNG, and also recovers the seed.

Some concepts are needed before the permanent compromise attack description. The BM generator's internal (unknown) state in time i is defined as an ordered set $X(i) \triangleq \{x_i, x_{i-1}, x_{i-2}, \dots, x_0\}$, where x_0 is the seed. Each $x_k \in X(i)$ is associated to a bit outputted by the generator, i.e., $b_k = \delta(x_k), k = 0, 1, \dots, i$. We call x_i the *representative* of the internal state $X(i)$.

Since the evaluation of the exponential in (2) is computationally efficient and considering that at some point the sequence will repeat, if any single element of a set $X(i)$ along its index are discovered then all the previous and future internal states can be recovered, i.e., all elements of $X(i)$ are found out.

To attack a BM generator used in a cryptosystem, an adversary has knowledge of the public parameters p and g, and has previously discovered a sequence of output bits ($\mathbf{b} \triangleq \{b_i, i = 1, 2, \dots\}$) produced by the generator under attack.

In the attempt to recover the internal state of the generator, the adversary needs to estimate an element $x_k \in X(i)$ along its index. Without loss of generality, consider $k = i$. Let $\hat{X}_i$ be the set of guesses to $x_i \in X(i)$. The set $\hat{X}_i$ will be referred as the *estimator set relative to* x_i, or plainly *estimator set*. For example, as stated in the BM initialization, the seed x_0 is randomly chosen from $\mathbb{Z}_p^*$, so the adversary would start in the attempt to attack the generator with the estimator set $\hat{X}_0 = \mathbb{Z}_p^*$.

With a given estimator set $\hat{X}_i$ that contains x_i and the knowledge of the bit b_{i+1} outputted by the generator, the adversary would proceed as follows to produce $\hat{X}_{i+1}$. Compute $A_{i+1} = g^{\hat{X}_i} \bmod p$, i.e., the image of $\hat{X}_i$ under action of the map $x \rightarrow g^x \bmod p$. Let $A_{i+1} = A_{i+1}^{(0)} \cup A_{i+1}^{(1)}$ be the partition of A_{i+1} due to the BM rule, that is, $x \in A_{i+1}^{(0)}$ iff $\delta(x) = 0$ and similar to $A_{i+1}^{(1)}$. The estimator set $\hat{X}_{i+1}$ will be given by:

$$\hat{X}_{i+1} = A_{i+1}^{(b_{i+1})}. \tag{5}$$

Notice that $\hat{X}_{i+1},\ i = 0, 1, 2, \dots$ only contains elements out of the class defined by $A_{i+1}^{(b_{i+1})}$. The Algorithm 1 synthesizes the procedure followed by the adversary.

Algorithm 1 Pseudocode of the algorithm used by the adversary to promove a permanent compromise attack to a BM generator. In this algorithm, δ denotes the function shown in (3); j is the number of bits in **b**; and every set $\hat{X}_i$ used for the first time is an empty set.

```
i ← 1
X̂_0 ← {1, 2, ..., p − 1}
while (i ≤ j) do
  for all x ∈ X̂_{i−1} do
    if δ(g^x mod p) = b_i then
      X̂_i ← X̂_i ∪ {g^x mod p}
    end if
  end for
  if i ≠ j then
    i ← i + 1
  end if
end while
print X̂_i
```

To exemplify the described permanent compromise state attack, suppose that the adversary eavesdropped 2 sequenced output bits, $\mathbf{b} = \{10\}$, from a BM with parameters $p = 7$ and $g = 3$. He would start his estimators to the seed with the set $\hat{X}_0 = \{1, 2, \dots, 6\}$. With the information that the first bit observed was $b_1 = 1$, he would discard the numbers lesser than $(7 - 1)/2 = 3$ and perform the operation $g^x \bmod p$ in the remaining ones, resulting in the set of estimators to x_1, $\hat{X}_1 = \{4, 5, 6\}$. Proceeding the same way, but with the next observed bit $b_2 = 0$, the intruder would update from $\hat{X}_1$ his estimators to x_2, resulting in $\hat{X}_2 = \{1\}$. In this case, with only two intercepted bits, the adversary could retrieve, with 100% of sure, the internal state of the generator.

B. Correctness and Analysis of the Algorithm (1)

To verify that the permanent compromise attack to BM recovers the internal state, it must be proved that: (i) every estimator set $\hat{X}_i$ contains $x_i \in X(i)$; and (ii) given sufficient bits in **b**, the size of the set $\hat{X}_i$ is unitary for i large enough or the adversary will easily to predict the next generator's output. The skecth of the proofs to the requirements (i) and (ii) are presented below. In order to prevent trivialities we assume that x_0 is member of a larger cycle in the functional graph [10] of the exponential map (2). This assumption is clearly acceptable since short cycles correspond to high predictability of the sequence.

Proof of (i) *– Every set* $\hat{X}_i$ *of estimators contains the correspondent representative* $x_i \in X(i)$*:* Induction in i. Clearly, the claim is valid for $i = 0$ as $x_0 \in \hat{X}_0 = Z_p^*$. Assume $x_i \in \hat{X}_i$ (induction hypothesis). It must be shown that $x_{i+1} \in \hat{X}_{i+1}$. Indeed, from (5), $y \in \hat{X}_{i+1}$ if only if (1) $y = g^x \bmod p$ for some $x \in \hat{X}_i$ and, (2) $\delta(y) = b_{i+1}$. But,

by the induction hypothesis, $x_i \in \hat{X}_i$ and, according to BM productions, $\delta(x_{i+1} = g^{x_i} \bmod p) = b_{i+1}$ so x_{i+1} attains both requirements (1) and (2). We have done. ■

Proof of (ii) (sketch) – Cardinality $|\hat{X}_i| = 1$ for i enough large or the next bit is fully predictable: In the $i-$th step, prior observing b_i, the adversary calculate $A_i = g^{\hat{X}_{i-1}} \bmod p = A_i^{(0)} \cup A_i^{(1)}$. Define $q = |\hat{X}_i|/|\hat{X}_{i-1}|$, then observe that $q = \Pr[b_i = 0|\hat{X}_{i-1}]$ or $q = \Pr[b_i = 1|\hat{X}_{i-1}]$ due to the BM rule. Then the uncertainty of b_i is $\mathcal{H}(q)$ where $\mathcal{H}(q)$ stands for the binary entropy calculated in q. Note that if $A_i^{(0)} = \emptyset$ then $q = 1$ and b_i is completely predictable and similar is true if $A_i^{(1)} = \emptyset$. Thus we assume that a good choice of parameters x_0, g and p should hold $\mathcal{H}(q) > 1 - \epsilon$, where ϵ is a small positive number. Therefore $q = |\hat{X}_i|/|\hat{X}_{i-1}|$ is near enough to $1/2$ or the bit b_i is fully predictable. This conclude the sketch of the proof of (ii). ■

To analyze the complexity of the permanent compromise attack described it is important not forget that p in practical applications is a large prime, i.e., $p \approx 2^n$, and that evaluations of (2) have unitary cost of the exponent. In the adversary's algorithm the modular exponentiation is evaluated to modular residues of p, thus the complexity is $O(p) \approx O(2^n)$ to a classical computer.

III. Grover's Quantum Search Algorithm

The Grover's quantum search algorithm performs a generic search for a solution in an unsorted database [8]. This algorithm is adequate to many problems where the best-known algorithm is to naively search through the potential solutions until one is found.

Suppose that an unsorted database contains N elements and M solutions. To represent the N elements of the database in binary, there are necessary $n = \lceil \log N \rceil$ bits, i.e., n is the problem input size. To perform the Grover's search in this database the steps described below must be followed:

1) Preparation of two registers: the first containing n qubits initialized as $|0\rangle$, and the second register containing a single $|1\rangle$ *ancilla* qubit;
2) Hadamard transformation: must be applied in both registers, producing $\frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle$ and $|-\rangle$;
3) Grover iteration: Repeat k times the following steps:
   a) Phase inversion: Invert the phase of the states in the first register that are solutions to the search problem. Use the second register as an auxiliary to this operation;
   b) Amplitude amplification: Re-invert the phases that were inverted in the previous step over the total average. Due to the fact that a quantum system is described by an unitary vector, this process will increase the amplitudes of the solutions and decreases the others;
4) Measurement: Perform a measurement in the first register.

The step 2 consisted in the Hadamard transformation of the input, and thus the state before the Grover's iteration can be written considering the solutions (subspace $|\psi_{good}\rangle$) and the non-solutions (subspace $|\psi_{bad}\rangle$):

$$
\begin{aligned}
|\psi\rangle &= H^{\otimes n+1} |0\rangle^{\otimes n} |1\rangle && (6)\\
&= \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} |k\rangle && (7)\\
&= \sqrt{p_{good}} |\psi_{good}\rangle + \sqrt{p_{bad}} |\psi_{bad}\rangle && (8)\\
&= \sin(\theta) |\psi_{good}\rangle + \cos(\theta) |\psi_{bad}\rangle && (9)
\end{aligned}
$$

where $p_{good} = \sum_{x \in X_{good}} |\alpha_x|^2$, $\alpha_x = \frac{1}{\sqrt{2^n}}$, $p_{bad} = 1 - p_{good}$, $\theta \in \left(0, \frac{\pi}{2}\right)$, and $\sin^2(\theta) = p_{good}$.

The operator G, denoting the Grover iteration, must be applied $k \approx \frac{\pi}{4}\sqrt{\frac{N}{M}}$ times in order to amplify the amplitudes of the solutions to the problem:

$$G^k |\psi\rangle = \cos((2k+1)\theta) |\psi_{bad}\rangle + \sin((2k+1)\theta) |\psi_{good}\rangle . \quad (10)$$

The Grover's quantum search algorithm provides a quadratic speed-up over its best-known classical counterpart. The extremely wide applicability of searching problems makes this algorithm interesting and important. The next section shows an application of the Grover's algorithm to endanger the security of a BM generator.

IV. Quantum Permanent Compromise Attack

The quantum algorithm that performs the permanent compromise attack to Blum-Micali is composed of two main parts: the first part, illustrated in the circuit of Figure 1, that is responsible for the identification of the estimator set to the representative of $X(i)$; and the second part, that follows the first and is illustrated in the Figure 2, that performs the amplitude amplification with the Grover's quantum search algorithm.

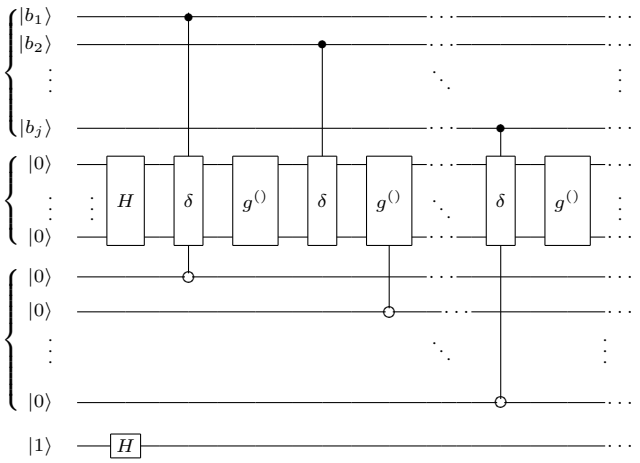

Figure 1. First part of the quantum circuit that implements the quantum permanent compromise attack to Blum-Micali pseudorandom generator.

This algorithm starts creating a superposition of qubits, representing all the elements in $\hat{X}_0 = \mathbb{Z}_p^*$, and uses the information of the **b** bits discovered to mark which of these elements could be in the internal state $X(i)$. After this phase, the Grover algorithm is run and the amplitude of

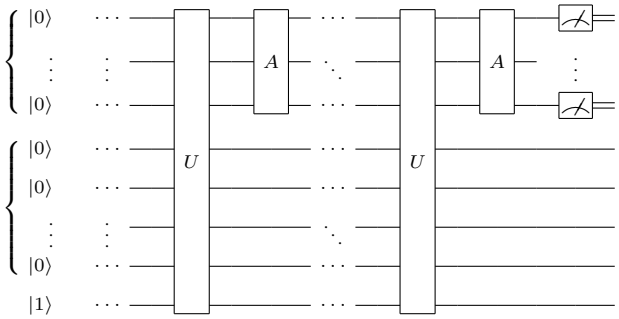

Figure 2. Second part of the quantum circuit that implements the quantum permanent compromise attack to Blum-Micali pseudorandom generator. This part follows the first one and implements the Grover's quantum search.

the elements marked as solutions are amplified, being able to be measured with high probability. The input for the circuit that implements the quantum permanent compromised is described as follows:

1) The first register is composed by the sequence of bits discovered by the intruder simply codified as qubits;
2) The second register, containing $\lceil \log p \rceil$ qubits, are used to represent the elements of $\hat{X}_i$;
3) Containing as much qubits as the first register, this third register is composed of *ancilla* qubits, that will help in the identification of the estimators for $x \in X$;
4) The fourth register contains the auxiliary qubit for the Grover's quantum search algorithm.

The gates δ and $g^{()}$ in the first part, implement the following operations:

$$\delta_c \left|x\right\rangle \left|0\right\rangle = \begin{cases} \left|x\right\rangle \left|0\right\rangle & \text{if } x \notin \mathbb{Z}_p^* \\ \left|x\right\rangle \left|1\right\rangle & \text{if } x > \frac{(p-1)}{2} \quad \text{and } c = 1 \\ \left|x\right\rangle \left|1\right\rangle & \text{if } x \leq \frac{(p-1)}{2} \quad \text{and } c = 0 \\ \left|x\right\rangle \left|0\right\rangle & \text{otherwise} \end{cases}$$

$$g^{()} \left|x\right\rangle = \begin{cases} x & \text{if } x \notin \mathbb{Z}_p^* \\ g^x \bmod p & \text{otherwise} \end{cases}$$

Both gates are reversible and can be implement efficiently in a quantum computer, since they are already implemented efficiently on a classical computer. After this first part, given j bits in $\mathbf{b}$, it is expected that the estimators to x_j will be the states associated to $\left|11\ldots1\right\rangle$ in the third register. The gates U and A of the quantum searching part denotes, respectively, the phase inversion and amplitude amplification of Grover's algorithm. The U oracle implements the function $f(x) = 1$ if the third register is equal to $\left|11\ldots11\right\rangle$, and $f(x) = 0$ otherwise. The applications of U and A are equivalent to the Grover iterations from (10).

In the attempt to exemplify the quantum permanent compromise attack, suppose that an intruder recovered the bits $\mathbf{b} = \{001\}$ and knows that $p = 7$ and $g = 3$ for a certain BM. He also possess a quantum computer that implements the quantum permanent compromise attack algorithm described, and wants to use this algorithm to recover the internal state of the generator. The adversary would prepare the first register with the state $\left|001\right\rangle$; the second register with 3 qubits initialized with $\left|0\right\rangle$; the *ancilla* register with the state $\left|000\right\rangle$; and the Grover auxiliary *ancilla* qubit with $\left|1\right\rangle$, resulting in the following initial state:

$$\left|\psi_0\right\rangle = \left|001\right\rangle \left|000\right\rangle \left|000\right\rangle \left|1\right\rangle \tag{11}$$

Since the first register is just used to control the application of δ, it will be omitted in the following steps. According to the circuit in the Figure 1, a Hadamard gate must be applied in the second and fourth registers, and the other registers must be left unchanged, resulting:

$$\begin{aligned} \left|\psi_1\right\rangle &= H^{\otimes 3} \otimes \mathbb{1}^{\otimes 3} \otimes H \left[\left|000\right\rangle \left|000\right\rangle \left|1\right\rangle\right] && (12) \\ &= \frac{1}{\sqrt{2^3}} \sum_{k=0}^{2^3-1} \left|k\right\rangle \left|000\right\rangle \left|-\right\rangle && (13) \\ &= \frac{1}{\sqrt{8}} \left(\left|0\right\rangle + \left|1\right\rangle + \ldots + \left|7\right\rangle\right) \left|000\right\rangle \left|-\right\rangle && (14) \end{aligned}$$

The first application of δ is controlled by a 0 qubit and targets the first *ancilla* qubit to 1 if the second register is lesser or equal than $(p-1)/2$. As consequence of δ application, the $\left|\psi_1\right\rangle$ state evolves to $\left|\psi_2\right\rangle$:

$$\begin{aligned} \left|\psi_2\right\rangle &= \delta_0 \left|\psi_1\right\rangle && (15) \\ &= \frac{1}{\sqrt{8}} \left[\left(\left|0\right\rangle + \left|4\right\rangle + \ldots + \left|7\right\rangle\right) \left|000\right\rangle + \right. \\ &+ \left. \left(\left|1\right\rangle + \left|2\right\rangle + \left|3\right\rangle\right) \left|100\right\rangle\right] \left|-\right\rangle && (16) \end{aligned}$$

The next step is to perform the transformation $g()$:

$$\begin{aligned} \left|\psi_3\right\rangle &= g^{()} \left|\psi_2\right\rangle && (17) \\ &= \frac{1}{\sqrt{8}} \left[\left(\left|0\right\rangle + \left|4\right\rangle + \left|5\right\rangle + \left|1\right\rangle + \left|7\right\rangle\right) \left|000\right\rangle + \right. \\ &+ \left. \left(\left|3\right\rangle + \left|2\right\rangle + \left|6\right\rangle\right) \left|100\right\rangle\right] \left|-\right\rangle && (18) \end{aligned}$$

The expression of the state $\left|\psi_3\right\rangle$ tells that $\hat{X}_1 = \{3, 2, 6\}$. But, the intruder knows two more bits, and also the estimators set $\hat{X}_2$ and $\hat{X}_3$ can be obtained.

The application of δ_0 controlled by the second qubit of the first register results:

$$\begin{aligned} \left|\psi_4\right\rangle &= \frac{1}{\sqrt{8}} \left[\left(\left|0\right\rangle + \left|4\right\rangle + \left|5\right\rangle + \left|7\right\rangle\right) \left|000\right\rangle + \right. \\ &+ \left. \left|1\right\rangle \left|010\right\rangle + \left|6\right\rangle \left|100\right\rangle + \left(\left|3\right\rangle + \left|2\right\rangle\right) \left|110\right\rangle\right] \left|-\right\rangle && (19) \end{aligned}$$

The result of $g^{()}$ to $\left|\psi_5\right\rangle$ is given below:

$$\begin{aligned} \left|\psi_5\right\rangle &= g^{()} \left|\psi_4\right\rangle && (20) \\ &= \frac{1}{\sqrt{8}} \left[\left(\left|0\right\rangle + \left|4\right\rangle + \left|5\right\rangle + \left|7\right\rangle\right) \left|000\right\rangle + \right. \\ &+ \left. \left|3\right\rangle \left|010\right\rangle + \left|1\right\rangle \left|100\right\rangle + \left(\left|6\right\rangle + \left|2\right\rangle\right) \left|110\right\rangle\right] \left|-\right\rangle && (21) \end{aligned}$$

To conclude the first phase, the δ gate, controlled by the third qubit of the first register and targeting the third qubit of the third register, needs to be applied to the state $\left|\psi_5\right\rangle$:

$$\begin{aligned} \left|\psi_6\right\rangle &= \delta_1 \left|\psi_5\right\rangle && (22) \\ &= \frac{1}{\sqrt{8}} \left[\left(\left|0\right\rangle + \left|7\right\rangle\right) \left|000\right\rangle + \left(\left|4\right\rangle + \left|5\right\rangle\right) \left|001\right\rangle \right. \\ &+ \left. \left|3\right\rangle \left|010\right\rangle + \left|1\right\rangle \left|100\right\rangle + \left|2\right\rangle \left|110\right\rangle + \left|6\right\rangle \left|111\right\rangle\right] \left|-\right\rangle && (23) \end{aligned}$$

After the execution of the first phase of the algorithm, the state $|6\rangle$ in Eq. (23) is the only one associated to $|111\rangle$, identifying that the set $\hat{X}_3$ is unitary and, therefore, $x_3 = 6$. Although the representative is already identified, this information is accessible only at the quantum level – at the final of the first phase a measurement would return any number from 0 to 7 with the same probability. Therefore, it is needed to perform Grover iterations, the second phase of the algorithm, to increase the probability that the representative will be returned after a measurement.

To simplify the calculus, the state $|\psi_6\rangle$ can be rewritten in the following manner:

$$|\psi_6\rangle = \sqrt{\frac{7}{8}}\left|\neg\hat{X}_i\right\rangle|y\rangle|-\rangle + \frac{1}{\sqrt{8}}\left|\hat{X}_i\right\rangle|111\rangle|-\rangle \quad (24)$$

$$= \sin(\theta)\left|\psi_{\hat{X}_i}\right\rangle + \cos(\theta)\left|\psi_{\neg\hat{X}_i}\right\rangle \quad (25)$$

where $i = 3$; $\left|\hat{X}_i\right\rangle = |6\rangle$; $\left|\neg\hat{X}_i\right\rangle$ denotes all the states that aren't in the estimator set $\hat{X}_3$; $y \neq 111$; $\left|\psi_{\hat{X}_i}\right\rangle = |6\rangle|111\rangle|-\rangle$; $|\psi_{\neg\hat{i}}\rangle = \left|\neg\hat{X}_i\right\rangle|y\rangle|-\rangle$; $\theta \in (0, \frac{\pi}{2})$ satisfies $\sin^2(\theta) = \frac{1}{8}$; and, thus $\theta \approx 0.36$ radians.

The optimal number k of iterations to be performed is $\frac{\pi}{4}\sqrt{\frac{8}{1}} \approx 2$. So, two Grover iterations on $|\psi_6\rangle$ results:

$$|\psi_7\rangle = G^2|\psi_6\rangle \quad (26)$$

$$= \cos[(2\cdot 2+1)\theta]\left|\psi_{\bar{i}}\right\rangle + \sin[(2\cdot 2+1)\theta]|\psi_i\rangle \quad (27)$$

$$= \cos(5\theta)\left|\psi_{\bar{i}}\right\rangle + \sin(5\theta)|\psi_i\rangle \quad (28)$$

$$= \cos(1.8)\left|\psi_{\bar{i}}\right\rangle + \sin(1.8)|\psi_i\rangle \quad (29)$$

This result states that a measurement in the second register of $|\psi_7\rangle$ will return the representative to $X(3)$ ($x_3 = 6$) with probability $|\sin(1.8)|^2 = 94,53\%$ of sure. With this information the intruder will be able to retrieve all the set $X(i)$ of internal states from the generator under attack, endangering its unpredictability.

The quantum permanent compromise attack to BM described in this section has total complexity equal to the sum of the complexities of each phase. In the first one, the cost is equal to $O(1)$, because of $2\cdot j\cdot O(1)$ oracle's operations performed; the second part has total cost equal to the Grover's algorithm, $O(\sqrt{p}) = O(\sqrt{2^n})$ because $p \approx 2^n$. So, the total cost of the quantum algorithm is $O(1)+O(\sqrt{2^n}) = O(\sqrt{2^n})$. This result represents a quadratic speedup over its classical counterpart.

V. Related Work

Classical attacks to pseudorandom generators have been studied for many researchers, not only with cryptographic purposes. Kelsey et al. [1] proposed a taxonomy that classifies attacks to PRNGs in six different categories and also discuss their extensions. According to these authors, the study of cryptanalytic attacks on PRNGs has practical and theoretical applications because (i) there aren't any widespread understanding of the possible attacks to PRNGS; (ii) PRNGs are single point of failure in many real-world cryptosystems; and, (iii) many systems use badly-designed PRNGs or use them in ways to make various attacks easier that they need to be.

So far, regarding quantum attacks to pseudorandom generators, no references were found in the literature that present such attacks. Since the proposed algorithm characterizes a speedup over its classical counterpart, it opens up the possibility to use quantum computation to endanger the security of such generators.

VI. Conclusions and Future Work

This paper introduced a quantum permanent compromise attack to the Blum-Micali generator. This algorithm uses quantum searching with multiple solutions and parallelism to retrieve the generator's internal state with high probability. The algorithmic complexity is $O(\sqrt{2^n})$ contrasting with $O(2^n)$ of the best classical algorithm known, where n is the input size.

To the authors's knowledge, the attack introduced in this work is the first proposal of quantum attacks against pseudorandom numbers generators. Furthermore, this attack brings results against the security of a generator used in real-world cryptographic applications.

In future works, the authors intend to analyze the adoption of some elements from the quantum algorithm to the discrete logarithm problem into the permanent compromise attack.

Acknowledgements

The authors gratefully acknowledge the financial support rendered by the Brazilian National Council for the Improvement of Higher Education (CAPES), and the useful discussions with Francine Melo and Gilson O. Santos.

References

[1] J. Kelsey, B. Schneider, D. Wagner, and C. Hall, "Cryptanalytic attacks on pseudorandom number generators," *Lecture Notes in Computer Science*, vol. 1372/1998, pp. 168–188, 1998.

[2] A. Rukhin, "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications," National Institute of Standards and Technology, Tech. Rep., 2008.

[3] G. Marsaglia, "The Marsaglia Random Number CDROM, including the DIEHARD Battery of Tests of Randomness," 1995.

[4] M. Blum and S. Micali, "How to generate cryptographically strong sequences of pseudorandom bits," *SIAM J. Comput.*, vol. 13 (4), pp. 850–864, 1984.

[5] L. Blum, M. Blum, and M. Shub, "A simple unpredictable pseudo-random number generator," *SIAM Journal on Computing*, vol. 15, pp. 364–383, 1986.

[6] B. S. Kaliski, "Elliptic curves and cryptography: A pseudorandom bit generator and other tools," Ph.D. dissertation, MIT, Cambridge, MA, USA, 1988.

[7] P. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Journal on Computing*, vol. 26, pp. 1484–1509, 1997.

[8] L. K. Grover, "Quantum mechanics helps in searching for a needle in a haystack," *Physical Review Letter*, vol. 79, pp. 325–328, 1997.

[9] A. Sidorenko and B. Schoenmakers, "State recovery attacks on pseudorandom generators," in *Western European Workshop on Research in Cryptology*, 2005, pp. 53–63.

[10] D. Cloutier, "Mapping the discrete logarithm," Senior thesis – Rose-Hulman Institute of Technology, 2005.