# A Generalized Quantum Permanent Compromise Attack to the Blum-Micali Construction

**Elloá B. Guedes, Francisco M. de Assis, Bernardo Lula Jr.**

[1]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.*** *The Blum-Micali construction defines cryptographically secure pseudorandom generators widely adopted in many real-world cryptosystems. By endangering this construction, the security of certain cryptographic applications may become vulnerable. In the attempt to menace the security of Blum-Micali construction, this paper presents a generalized quantum permanent compromise attack to it. This attack is based on Grover's quantum search and on quantum parallelism to retrieve the generator's internal state. Experimental results are also provided to support the number of bits the adversary must know to perform the attack successfully against the Blum-Micali generator. The attack may bring new security requirements to be imposed on pseudorandom numbers generators.*

## 1. Introduction

There is a close relationship between cryptography and randomness. The security of cryptographic algorithms usually depends on the random choice of keys and bit sequences. Ideally, this random data should be originated from a non-deterministic phenomena, i.e., whose behavior cannot be predicted.

However, digital computers cannot generate random numbers, and it is generally not convenient to connect a computer to some external source of random events. It is possible to overcome this disadvantage if there is some source of *pseudorandom numbers* – algorithms that yield numbers that appears to be random. Many methods have been suggested in the literature for generating such pseudorandom numbers [Gentle 2003].

A *pseudorandom numbers generator* (PRNG) is a function $f$ that 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 [Rukhin 2008, Marsaglia 1995].

In cryptography, the requirements for randomness are more stringent than in other domains: the known conditional probability of the next random number occur, given the previous history, is no different from the known unconditional probability

[Blum and Micali 1984]. This requirement defines a special class of pseudorandom generators – the *cryptographically secure pseudorandom number generators* (CSPRNGs).

These generators can be derived from one-way permutations with hard-core predicates as defined by the *Blum-Micali construction*. Examples of generators founded on the Blum-Micali construction are Blum-Blum-Shub [Blum et al. 1986], Kaliski [Kaliski 1988], and the Blum-Micali generator [Blum and Micali 1984] whose security is based on the assumptions of intractability of the factoring, elliptic curve discrete logarithm and discrete logarithm problems, respectively. Since attacks focus on the parts of the cryptosystems that are most susceptible, it is essencial to analyze the vulnerability of such generators against menaces.

Quantum computing, a computational paradigm based on Quantum Physics, has been proposing efficient algorithms to certain problems where an efficient classical algorithm is not known. One of the examples is related to the security of the RSA, that is based on the assumption of hardness of factoring to a classical computer. However, Shor proposed a quantum factoring algorithm capable to solve this problem efficiently [Shor 1997]. This result opened up the possibility to endanger the security of classical cryptosystems with this computational paradigm.

Towards attacks to pseudorandom numbers generators with quantum computing, Guedes et al. [Guedes et al. 2010b] proposed a permanent compromise attack to the Blum-Micali generator. This attack was based on the Grover algorithm, a quantum procedure to perform search in an unsorted database [Grover 1997]. According to the authors, the proposed attack has a quadratic speedup over its classical counterpart.

This paper intends to generalize the quantum permanent compromise attack proposed by Guedes et al. to the whole Blum-Micali construction, showing how the quantum gates can be extended according to each generator. Furthermore, this paper presents an experimental study to estimate the number of bits the adversary needs to know to perform the attack successfully against the Blum-Micali generator. The results obtained increase the confidence in boundaries previously estimated.

The rest of this paper is organized as follows. The Section 2 presents the concepts of the Blum-Micali construction, the permanent compromise attack to it, and also an experimental study regarding attacks to the Blum-Micali generator. In Section 3 the two phases and number of Grover's iterations from the generalized quantum attack to the Blum-Micali construction are characterized, an example is presented to illustrate such attack. In the Section 4 related works are briefly discussed and in the Section 5 conclusions and future work are drawn.

## 2. The Blum-Micali Construction

The Blum-Micali construction is a family of pseudorandom numbers generators widely adopted in cryptography [Blum and Micali 1984]. The generators from this family are defined by a one-way permutation $\rho$ over a domain $\mathcal{D}$ and a hard-core predicate $\phi$ for the one-way permutation. The seed $x_0$ is obtained through a random choice of an element in the domain, denoted by $x_0 \in_R \mathcal{D}$. The productions (bits) from these generators are obtained as follows:

$$x_i = \rho(x_{i-1}) \quad (2)$$

$$b_i = \phi(x_i) \quad (3)$$

Three generators are the main representatives of this family: the Blum-Blum-Shub (BBS), Kaliski, and Blum-Micali (BM) generators.

The Blum-Blum-Shub generator's one-way permutation is based on quadratic residues modulo $M$, where $M$ is the product of two primes both congruent to $3 \bmod 4$. Its $\phi$ function is characterized by the simple extraction of the $j$-th bit, where $j$ is determined a priori. The security of the BBS generator is based on the assumption of the hardness of factoring [Blum et al. 1986].

The Kaliski generator is founded on the elliptic curve discrete logarithm problem. Its seed is a random point at the curve and the hard-core predicate for the one-way permutation is a binary test to the $x$ abscissa, comparing it to the value of a large prime previously fixed [Kaliski 1988, Sidorenko and Schoenmakers 2005].

The Blum-Micali generator is based on modular exponentiation. This generator will be used to illustrate the examples and concepts along this paper. For this reason, its characteristics will be presented in details in the next section. After that, a permanent compromise attack to the Blum-Micali construction will be presented.

### 2.1. The 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 $\bmod p$. Let $g$ be a generator of $\mathbb{Z}_p^*$.

The Blum-Micali generator 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 the one-way function $x_i = g^{x_{i-1}} \bmod p$ over the domain $\mathbb{Z}_p^*$, and a hard-core predicate for the permutation, denoted by $\delta$, as shown below:

$$x_i = g^{x_{i-1}} \bmod p \tag{4}$$
$$b_i = \delta(x_i) \tag{5}$$

where $\delta$ is a binary function with the following definition:

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

The security of the BM generator relies on the hardness to solve the inverse of the one-way function (4). This inversion is equivalent to solve the discrete logarithm problem and no efficient algorithm to perform this operation is known in classical computing.

To illustrate how the bits are outputted by the BM generator, suppose that it was initialized with parameters ($p = 7$, $g = 5$, $x_0 = 6$). The bits produced are shown in the Diagram (6).

$$\begin{array}{ccccccccc} 6 & \rightarrow & 1 & \rightarrow & 5 & \rightarrow & 3 & \rightarrow & \ldots \\ \downarrow & & \downarrow & & \downarrow & & \downarrow & & \\ 1 & & 0 & & 1 & & 0 & & \end{array} \tag{6}$$

## 2.2. Permanent Compromise Attack to the Blum-Micali Construction

A permanent compromise attack happens to a pseudorandom 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 CSPRNG, and also recovers the seed.

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

Since the evaluation of the one-way permutation $\rho$ 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 one of the generator from the BM construction, an adversary has knowledge $(i)$ of the type of the generator, $(ii)$ of the public parameters, and $(iii)$ has previously discovered a sequence of output bits ($\mathbf{b} \triangleq \{b_i, i = 1, 2, \ldots\}$) 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 previously, the seed $x_0$ is randomly chosen from $\mathcal{D}$, so the adversary would start in the attempt to attack the generator with the estimator set $\hat{X}_0 = \mathcal{D}$.

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 $A_{i+1} = \rho(\hat{X}_i)$, i.e., the image of $\hat{X}_i$ under action of the map $x \mapsto \rho(x)$.

Let $A_{i+1} = A_{i+1}^{(0)} \cup A_{i+1}^{(1)}$ be the partition of $A_{i+1}$ due to the $\phi$ rule, that is, $x \in A_{i+1}^{(0)}$ iff $\phi(x) = 0$ and similar to $A_{i+1}^{(1)}$. The estimator set $\hat{X}_{i+1}$ will be given by:

$$\hat{X}_{i+1} = A_{i+1}^{(b_{i+1})}. \tag{7}$$

Notice that $\hat{X}_{i+1},\ i = 0, 1, 2, \ldots$ only contains elements out of the class defined by $A_{i+1}^{(b_{i+1})}$. The proofs of correctness of this algorithm can be seen in the paper of Guedes et al. [Guedes et al. 2010b]. These authors also suggest boundaries to the number of bits the adversary must know to perform the attack successfully, stating that this number is bounded by $\log p$ to the BM generator. The next section will present an experimental study performed in order to verify if practical results would follow this theoretical predicition

## 2.3. Experimental Study

This section presents an experimental study with the BM generator. This experiment can be explained as a game played between the adversary and the generator. To each generator

configured with parameters ($p$, $g$,$x_0$), the adversary would ask for many bits as necessary until he could predict the generator's next output with $100\%$ of sure.

The parameters ($p$, $g$,$x_0$) to initialize each generator were configured as follows:

1. The generator $g$: four values of $g$ where chosen ( 3, 5, 17, and 19). The requirement for different values of $g$ is explained because if a single value were chosen it could bias the number of bits resultant;
2. The prime $p$: all prime values in the range $[257, 17863]$ were up to be used, however only those where $p - 1$ and $g$ were coprimes were considered;
3. The seed $x_0$: uniformly sampled in the range $[1, p - 1]$ as suggested by the initialization of the BM generator.

After each game played between a generator and an adversary, the number of bits demanded was stored along with the value of $p$. This number of bits asked by the adversary until he could predict the generator's output with a $100\%$ sure is the response variable of the experiment. It should be noticed that situations where the generator contained a fixed point in the discrete logarithm functional graph were not considered [Cloutier 2005].

The null hypothesis ($H_0$) of the experiment is that the number of bits required by the adversary is bounded by $\log p$, and the alternative hypothesis ($H_a$) is that this number is not bounded by $\log p$, where $p$ is a large prime and one of the parameters to the BM generator. The level of confidence chosen was $95\%$[1]. A modified zero mean will be performed to verify the hypothesis under test.

The results of the attacks were summarized taking into account the number of bits in $p$, defining $6$ groups with values of $p$ from $8$ to $14$ bits. These results can be seen in the Table 1.

**Table 1. Experimental results grouped according to the number of bits in $p$.**

| Bits in $p$ | Required Sample Size | Number of Samples | Mean | Standard Deviation | Median | Confidence Interval |
|---|---|---|---|---|---|---|
| 8 | 168 | 561 | $8,044$ | $1,333$ | $8,000$ | $(7,933, 8,155)$ |
| 9 | 351 | 453 | $9,110$ | $2,178$ | $9,000$ | $(8,909, 9,311)$ |
| 10 | 329 | 354 | $10,107$ | $2,341$ | $10,000$ | $(9,862, 10,3520)$ |
| 11 | 297 | 591 | $11,090$ | $2,441$ | $11,000$ | $(10,892, 11,286)$ |
| 12 | 241 | 1074 | $12,011$ | $2,379$ | $12,000$ | $(11,863, 12,158)$ |
| 13 | 196 | 1979 | $13,111$ | $2,533$ | $13,000$ | $(12,999, 13,222)$ |
| 14 | 183 | 1679 | $14,008$ | $2,420$ | $14,000$ | $(13,892, 14,124)$ |

To achieve statistical significance within each group, there was a minimum number of required samples, showed in the Column $2$. In all cases, this number was respected, as reported in the Column $3$. The number of bits used by the adversary to perform the attack successfully was summarized according to mean, standard deviation, and median. Considering that the mean is similar to the number of bits in $p$ and the standard deviation is a small number, the data observed reveals that in all the attacks the number of bits required are within a restricted scope. The median coincides with $\log p$ in all cases. It shows a small variation in the number of bits demanded by the adversary to perform the attack, considering each group.

Confidence intervals, shown in the Column $7$, were constructed with the data obtained in the experiment. It is possible to notice that in all intervals, the corresponding

[1]A complete description of the hypothesis tests and data summarization procedures used in this work can be seen in the book of Jain [Jain 1991].

value of bits in $p$ is included. Thus, according to the zero mean tests performed, it implies in the absence of evidence to reject $H_0$ in all cases at the level of confidence of $95\%$. Furthermore, it can be noticed that the range of the intervals is narrow, not exceeding $1$ bit. This indicates that the number of bits demanded by the adversary was estimated with a high degree of precision.

However, despite the accordance between the theoretic and practical results, it is not possible to generalized the results. More experiments should be carried because there are infinite possibilites of initialization of BM generators. The results observed just reinforce the predictions provided by the theoretical results.

The theoretical and experimental results discussed are also in accordance with Boyar [Boyar 1989] and Krawczyk [Krawczyk 1992]. According to these authors, under certain conditions, if the period of the generator is $p$, then it is possible to predict the generator's output with a number of guesses bounded by a polynomial in $\log p$.

## 3. The Generalized Quantum Permanent Compromise Attack

Quantum algorithms make use of parallelism, superposition, entanglement and other intrinsic characteristics to speed up the solution of certain problems when compared to the classical equivalent algorithms. This motivated the proposal of a permanent compromise attack to the Blum-Micali construction with Quantum Computing. The quantum algorithm presented in this section is an extension of the Guedes et al. algorithm to the BM generator [Guedes et al. 2010b].

The quantum generalized algorithm that performs the permanent compromise attack to the BM construction is composed by three registers, described as follows:

1. The space of the solution, containing $\lceil \log |\mathcal{D}| \rceil$ qubits initialized as $|0\rangle$, where $\mathcal{D}$ is the domain of the generator;
2. Ancillary qubits, one for each qubit in the first register, all of them initialized as $|0\rangle$;
3. Grover ancillary qubit, initialized as $|1\rangle$.

The proposed algorithm is composed of two main parts: the first part is responsible for the *identification of the representative* $x_j$ of the internal state $X(j)$, as described in (7), and the second part performs the *amplitude amplification* of this element with the Grover's quantum search, increasing its probability to be measured. The quantum circuit showed in the Figure 1 implements the described algorithm, where its two parts are emphasized.

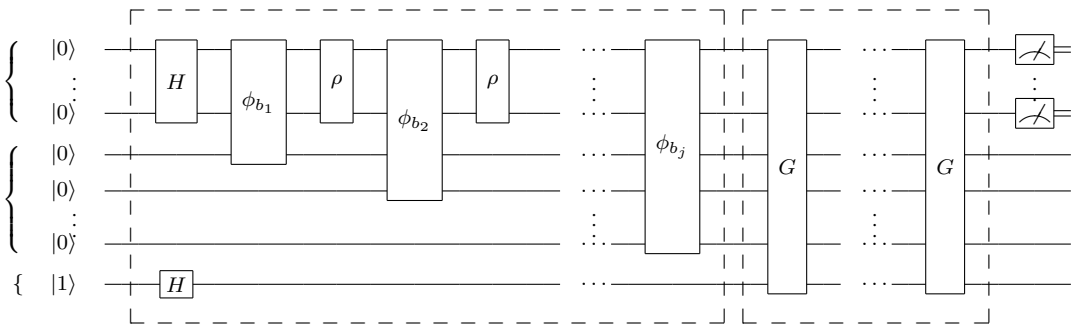


**Figure 1. Quantum circuit that implements the generalized quantum permanent compromise attack to the Blum-Micali construction.**

### 3.1. Identification of the Representative

The first part of the algorithm, composed by the gates $H$, $\rho$, and $\phi$, characterizes a procedure to *mark* the representative $x_j \in X(j)$. The state that is the representative will be associated to $1 \ldots 1$ in the third register. To perform this procedure, firstly the Hadamard gate must be applied to the first and third registers, putting them in a equally distributed superposition.

The gates $\rho$ and $\phi_{b_i}$ are defined in function of the one-way permutation and hard-core predicates of the generator under attack. They can be constructed as follows:

1. The $\rho$ gate: Implements the one-way permutation of the generator. Since public parameters and the type of the generator are available, the adversary can rebuild the rule in (2). If the input is not in the domain $\mathcal{D}$, it must be left unchanged;
2. The $\phi_{b_i}$ gate: Is analog to the hard-core predicate for the generator. It determines which members of the first register in $\mathcal{D}$ could have generated the bit $b_i$. In the affirmative cases, it *flips* the last associated qubit in the second register. The actuation of the $\phi_{b_i}$ gate is similar to a conditional statement of the classical programming languages, but with the advantage of quantum parallelism.

After this first phase, the input state can be described as the following partition:

$$|\psi\rangle = \sin(\theta) \left|\psi_{x_j}\right\rangle + \cos(\theta) \left|\psi_{\mathcal{D}-x_j}\right\rangle \tag{8}$$

where $\left|\psi_{x_j}\right\rangle$ denotes the subspace of the representative (solution), and $\left|\psi_{\mathcal{D}-x_j}\right\rangle$ denotes the subspace of the elements there are not associated to $1 \ldots 1$ in the second register (non-solutions).

### 3.2. Amplitude Amplification

Since the representative is marked, an amplitude amplification must be performed to increase its probability to be measured. This amplification will be build with the Grover's quantum search algorithm – a procedure to perform search in unsorted databases. According to it, the amplitude of the solution (in this case, the representative) will be increased and the amplitude of the other elements will be decreased, due to the restriction of unitarity in quantum states. When a measurement is performed, the solution will be returned with high probability [Grover 1997].

The execution of $k$ Grover's iterations can be summarized as:

$$G^k(|\psi\rangle) = \sin((2k+1)\cdot\theta) \left|\psi_{x_j}\right\rangle + \cos((2k+1)\cdot\theta) \left|\psi_{\mathcal{D}-x_j}\right\rangle \tag{9}$$

The required number of Grover's iterations will be discussed in the next subsection.

### 3.3. Required Number of Grover's Iterations

The required number of Grover's iterations $k$ is derived from $(i)$ the number $M$ of marked states, i.e., states associated to $1 \ldots 1$ in the third register; and from $(ii)$ the size $N$ of the database, initialized as $N = 2^n$ where $n = \lceil \log |\mathcal{D}| \rceil$. This number $k$ is given by:

$$k = \left\lfloor \frac{\pi}{4} \sqrt{\frac{N}{M}} \right\rfloor \tag{10}$$

where $\lfloor \cdot \rceil$ denotes the closest integer function. In the case of the BM generator, the number $M$ can be obtained by approximation, using the result that there is only one marked state when the adversary knows $\log p$ bits from the generator, as showed in the Section 2.3.

### 3.4. Example

To illustrate the generalized permanent compromise attack to the BM construction, let's consider an attack to a BM generator. The adversary has knowledge of the public parameters $p = 7$ and $g = 3$, and has discovered three sequential bits $\mathbf{b} = 001$. The adversary will prepare the input of the algorithm in the following state, according to the rules of definition of each register:

$$|\psi_0\rangle = |000\rangle\,|000\rangle\,|1\rangle \tag{11}$$

The next step is the application of $H$ gates in the first and third registers, resulting:

$$|\psi_1\rangle = \left[\frac{1}{\sqrt{8}}\left(|0\rangle + |1\rangle + \ldots + |7\rangle\right)\right]|000\rangle\,|-\rangle \tag{12}$$

After the identification of the representative phase, the state of the system will be the following:

$$\begin{aligned} |\psi_3\rangle &= \frac{1}{\sqrt{8}}\left(|0\rangle\,|000\rangle + |1\rangle\,|100\rangle + |2\rangle\,|110\rangle + |3\rangle\,|010\rangle + \right. \\ &+ \left. |4\rangle\,|001\rangle + |5\rangle\,|001\rangle + |6\rangle\,|111\rangle + |7\rangle\,|000\rangle\right)|-\rangle \end{aligned} \tag{13}$$

It can be noticed that the only state associated to 111 in the second register is the $|6\rangle$. Thus, the state $|\psi_3\rangle$ can be rewritten according to the following partition:

$$\begin{aligned} \left|\psi_3^{'}\right\rangle &= \frac{1}{\sqrt{8}}\,|6\rangle\,|111\rangle\,|-\rangle + \frac{1}{\sqrt{8}}\left(|0\rangle\,|000\rangle + |1\rangle\,|100\rangle + |2\rangle\,|110\rangle + \right. \\ &+ \left. |3\rangle\,|010\rangle + |4\rangle\,|001\rangle + |5\rangle\,|001\rangle + |7\rangle\,|000\rangle\right)|-\rangle \qquad (14) \\ &= \frac{1}{\sqrt{8}}\,|x_j\rangle\,|111\rangle\,|-\rangle + \sqrt{\frac{7}{8}}\,|\neg x_j\rangle\,|y\rangle\,|-\rangle \qquad (15) \\ &= \sin(\theta)\left|\psi_{x_j}\right\rangle + \cos(\theta)\left|\psi_{\mathcal{D}-x_j}\right\rangle \qquad (16) \end{aligned}$$

where $j = 3$; $|x_j\rangle = |6\rangle$; $\neg x_j$ denotes all states that aren't the representative $x_j$; $y \neq 111$; $\left|\psi_{x_j}\right\rangle = |6\rangle\,|111\rangle\,|-\rangle$; $\left|\psi_{\mathcal{D}-x_j}\right\rangle = |\neg\hat{x}_j\rangle\,|y\rangle\,|-\rangle$; and, $\theta \in \left(0, \frac{\pi}{2}\right)$ satisfies $\theta = \arcsin\left(\frac{1}{\sqrt{8}}\right) = 0.36$ radians.

The next step of the algorithm is the amplitude amplification. Before this step, it is necessary to determine how many Grover's iterations are necessary. The size of the space of the solution is $N = 8$, and $M = 1$ because just the state $|6\rangle$ is marked. Then $k = \left\lfloor \frac{\pi}{4}\sqrt{\frac{8}{1}} \right\rceil = 2$ iterations.

Therefore, 2 Grover's iterations on $\left|\psi_3^{'}\right\rangle$ will result:

$$\begin{aligned} |\psi_4\rangle &= G^2\left|\psi_3^{'}\right\rangle \qquad (17) \\ &= \sin((2\cdot 2+1)\cdot\theta)\left|\psi_{x_j}\right\rangle + \cos((2\cdot 2+1)\cdot\theta)\left|\psi_{\mathcal{D}-x_j}\right\rangle \qquad (18) \\ &= \sin(5\theta)\left|\psi_{x_j}\right\rangle + \cos(5\theta)\left|\psi_{\mathcal{D}-x_j}\right\rangle \qquad (19) \\ &= \sin(1.8)\left|\psi_{x_j}\right\rangle + \cos(1.8)\left|\psi_{\mathcal{D}-x_j}\right\rangle \qquad (20) \end{aligned}$$

A measurement in (20) will result in 6 with probability of $|\sin(1.8)|^2 \approx 94.83\%$. With this information the adversary successfully concluded the permanent compromise attack to the BM generator, endangering completely its unpredictability.

## 4. Related Work

Classical attacks to pseudorandom generators have been extensively explored by the literature. Kelsey et al. [Kelsey et al. 1998] 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 interests 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 than they need to be.

Regarding attacks to the BM construction, Sidorenko and Schoenmakers analyze the security of those generators against classical state recovery attacks, or permanent compromise attacks according to the Kelsey's taxonomy. They conclude that as the seed length increases, no polynomial-time adversary can retrieve the seed of the pseudorandom generator. Furthermore, they show that there are no tight reductions to this asymptotic statement [Sidorenko and Schoenmakers 2005].

Guedes et al. opened up the possibility to endanger the security of pseudorandom generators with Quantum Computing [Guedes et al. 2010b]. These authors proposed a quantum permanent compromise attack to the BM generator with a quadratic speedup over its classical counterpart. The present work generalizes this algorithm to the whole Blum-Micali construction and implies in the vulnerability of, at least, three different generators under quantum attacks – the BBS, Kaliski and BM pseudorandom generators.

Aside from the conclusions of Sidorenko and Schoenmakers, results to the BM generator showed that its security does not rely in the length of the seed, but in the public parameter $p$ that define the domain $\mathcal{D}$.

## 5. Conclusion and Future Work

This paper introduced a generalized quantum permanent compromise attack to the Blum-Micali construction. This attack is based on Grover's quantum search and on quantum parallelism to recover the generator's internal state with high probability. Furthermore, this paper presented an experimental study that analyzed the number of bits an adversary needs to perform the attack successfully against the BM generator.

The proposed generalized quantum permanent compromise attack algorithm has complexity of $O(\sqrt{|\mathcal{D}|})$, contrasting with $O(|\mathcal{D}|)$ from the classical algorithm. This quadratic speedup over the classical equivalent solution is explained by the use of Grover's quantum search. Since the BM construction is widely adopted in real-world cryptosystems, the attack proposed may bring new security requirements to be imposed on pseudorandom numbers generators.

The reader is referred to [Guedes et al. 2010a] in order to obtain examples of the proposed attack against the BBS and Kalsiki generators.

In future works, in the case of the BM generator, the authors aim 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).

## References

Blum, L., Blum, M., and Shub, M. (1986). A Simple Unpredictable Pseudo-Random Number Generator. *SIAM Journal on Computing*, 15:364–383.

Blum, M. and Micali, S. (1984). How to Generate Cryptographically Strong Sequences of Pseudorandom Bits. *SIAM J. Comput.*, 13 (4):850–864.

Boyar, J. (1989). Inferring Sequences Produced by Pseudo-Random Number Generators. *Journal of the Association for Computing Machinery*, 36:139–141.

Cloutier, D. (2005). Mapping the Discrete Logarithm. Senior thesis – Rose-Hulman Institute of Technology.

Gentle, J. E. (2003). *Random Number Generation and Monte Carlo Methods*. Springer.

Grover, L. K. (1997). Quantum Mechanics Helps in Searching for a Needle in a Haystack. *Physical Review Letter*, 79:325–328.

Guedes, E. B., de Assis, F. M., and Lula Jr., B. (2010a). Examples of the Generalized Quantum Permanent Compromise Attack to the Blum-Micali Construction. Available in http://sites.google.com/site/elloaguedes.

Guedes, E. B., de Assis, F. M., and Lula Jr., B. (2010b). Quantum Permanent Compromise Attack to Blum-Micali Pseudorandom Generator. In *Proceedings of the International Telecommunications Symposium 2010*.

Jain, R. (1991). *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation and Modeling*. John Wiley.

Kaliski, B. S. (1988). *Elliptic Curves and Cryptography: A Pseudorandom Bit Generator and Other Tools*. PhD thesis, MIT, Cambridge, MA, USA.

Kelsey, J., Schneider, B., Wagner, D., and Hall, C. (1998). Cryptanalytic Attacks on Pseudorandom Number Generators. *Lecture Notes in Computer Science*, 1372/1998:168–188.

Krawczyk, H. (1992). How to Predict Congruential Generators. *Journal of Algorithms*, 13:527–545.

Marsaglia, G. (1995). The Marsaglia Random Number CDROM, including the DIEHARD Battery of Tests of Randomness.

Rukhin, A. (2008). A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. Technical report, National Institute of Standards and Technology.

Shor, P. (1997). Polynomial-time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Journal on Computing*, 26:1484–1509.

Sidorenko, A. and Schoenmakers, B. (2005). State Recovery Attacks on Pseudorandom Generators. In *Western European Workshop on Research in Cryptology*, pages 53–63.