






On the Possibility of a Backdoor in the Micali-Schnorr Generator

Hannah Davis¹, Matthew D. Green², Nadia Heninger³, Keegan Ryan³,
and Adam Suhl³

¹ Seagate Technology, Scotts Valley, USA
hannah.e.davis@seagate.com

² Johns Hopkins University, Baltimore, USA
mgreen@cs.jhu.edu

³ University of California, San Diego, La Jolla, USA
nadiah@cs.ucsd.edu, {kryan, asuhl}@ucsd.edu

Abstract. In this paper, we study both the implications and potential impact of backdoored parameters for two RSA-based pseudorandom number generators: the ISO-standardized Micali-Schnorr generator and a closely related design, the RSA PRG. We observe, contrary to common understanding, that the security of the Micali-Schnorr PRG is not tightly bound to the difficulty of inverting RSA. We show that the Micali-Schnorr construction remains secure even if one replaces RSA with a publicly evaluatable PRG, or a function modeled as an efficiently invertible random permutation. This implies that any cryptographic backdoor must somehow exploit the algebraic structure of RSA, rather than an attacker’s ability to invert RSA or the presence of secret keys. We exhibit two such backdoors in related constructions: a family of exploitable parameters for the RSA PRG, and a second vulnerable construction for a finite-field variant of Micali-Schnorr. We also observe that the parameters allowed by the ISO standard are incompletely specified, and allow insecure choices of exponent. Several of our backdoor constructions make use of lattice techniques, in particular multivariate versions of Coppersmith’s method for finding small solutions to polynomials modulo integers.

1 Introduction

In 2013, a collection of leaks due to Edward Snowden revealed the existence of a large-scale U.S. government effort called the SIGINT Enabling Project., intended to compromise the integrity of cryptographic systems. Equipped with a \$200M annual budget, the project sought to “insert vulnerabilities into commercial encryption systems” and to “influence policies, standards and specification for commercial public key technologies” [3]. These leaks also revealed that the U.S. National Security Agency authored and maintained sole editorial control of the 2005 ISO 18031 standard on random bit generation [55], a standard that was largely incorporated into the U.S. ANSI X9.82 standard. A draft of the ANSI

standard in turn forms the basis for the U.S. National Institute for Standards and Technology’s NIST Special Publication 800-90A, which defines requirements for random bit generation in government-approved cryptographic products.

Even before the Snowden leaks, the NIST/ANSI and ISO standards have drawn scrutiny for their inclusion of a number-theoretic PRG known as the Dual Elliptic Curve Deterministic Random Bit Generator (Dual EC DRBG), a construction that is exploitable by a party that generates the system public parameters and retains a secret trapdoor [62]. The Snowden leaks inspired renewed investigation of this standard and its deployment, revealing that Dual EC was more widely deployed than many academic researchers had realized. Moreover, later investigation revealed that TLS and IPsec implementations incorporating Dual EC [17, 18] also made specific implementation decisions that rendered them practically exploitable by an adversary who possesses the Dual EC trapdoor.

Even if the standardized parameters were not *deliberately* backdoored, the mere possibility of such parameters poses a threat to users of a standardized PRG. In one noteworthy case, an undocumented implementation of Dual EC in Juniper NetScreen’s firewalls appears to have been exploited in practice; in 2012 an outside group compromised the NetScreen codebase and replaced Juniper’s Dual EC constants with parameters of their own devising [17]. These parameters were in place for over three years, presumably enabling the outside group to decrypt the communications of Juniper NetScreen customers, which at the time included the U.S. Federal Government. Demonstrating the existence, or ruling out the possibility, of methods to backdoor the parameter generation process is the only way to mitigate the risk of parameter substitution attacks which otherwise undermine the security of commercial encryption technology.

The MS DRBG Generator. While Dual EC is the only number-theoretic generator adopted as a NIST standard, the current draft of ISO 18031 (and early drafts of the ANSI X9.82 standard) also include a second public-key generator that has received surprisingly little successful analysis. Based on a design by Micali and Schnorr [50, 51], the MS DRBG algorithm is a pseudorandom number generator whose security is purportedly related to the hardness of breaking RSA. In brief, the algorithm is instantiated using a state s_0 and an RSA public key (N, e) , and at each stage the algorithm applies the RSA function to the state to obtain an integer $z_{i+1} = s_i^e \bmod N$. The most significant bits of z_{i+1} become the new state s_{i+1} for the next iteration of the algorithm, and the least significant bits of z_{i+1} are the output b_{i+1} of the pseudorandom number generator for that iteration.

While RSA (and Rabin)-based pseudorandom generators have been studied in the academic literature for many years [13, 32, 33, 50, 51, 63], two aspects of the MS DRBG standard draw attention. First, in contrast with common practice for RSA-based generators, the generator outputs up to a $1 - 2/e$ fraction of the bits (or up to 864 bits for a 1024-bit modulus with a claimed 80-bit security level.)¹

¹ Previous RSA-based generators (including some early drafts of MS DRBG) recommend outputting $\lg \lg N$ bits at each iteration (where \lg denotes base-2 logarithm). MS DRBG’s larger output is justified by a novel pseudorandomness assumption introduced by Micali and Schnorr [51].

More critically, the MS DRBG standard includes a design choice that is reminiscent of the Dual EC generator: namely, it incorporates a series of recommended public parameters that are intended to be used in production as the modulus N . As with Dual EC, the provenance of these moduli is not documented in the standard. However, correspondence from the ANSI standards process (revealed under the Freedom of Information Act [52]) supports the conclusion that both Dual EC and MS DRBG were authored by the National Security Agency, which also generated the parameters for both specifications. Unlike the Dual EC parameters (which could conceivably have been generated such that the generating party would not learn a trapdoor) according to the standard the MS DRBG moduli are the product of primes p, q chosen by the standard author. The NSA’s knowledge of this secret factorization calls into question the security of the generator when used in a setting where the NSA is adverse to its user.

MS DRBG has not, to our knowledge, been used in any real-world systems. This is perhaps unsurprising; there is virtually no reason ever to use a number theoretic DRBG instead of one based on symmetric cryptography: symmetric DRBGs have much better performance and still derive their security from well-studied cryptographic assumptions. However, several surprisingly widespread implementations of Dual EC DRBG (e.g., Juniper’s NetScreen implementation) were unknown to researchers and were only discovered by chance long after the deprecation of Dual EC. MS DRBG, though still in the ISO standard, has received far less attention than Dual EC from the research community. It is possible there were (or are) MS DRBG implementations used in production that have not received public scrutiny.

Given the known vulnerabilities in Dual EC DRBG and the (allegedly) identical provenance of MS DRBG, it is therefore reasonable to ask whether MS DRBG is vulnerable to an analogous attack. Concretely:

Does knowledge of the factors of (or malicious construction of) the recommended moduli imply a practical attack on the MS DRBG generator?

This question is surprisingly difficult to answer. While the literature is replete with studies of RSA-based generators, the majority of this work naturally assumes that the factorization of N is kept secret. In that setting, standard results on RSA hardcore bits can be used to argue the indistinguishability of generator output. Clearly such arguments no longer apply in settings where the factorization is *known* to the attacker. And yet in contrast to many other RSA-based constructions, knowledge of the factorization does not point to an obvious attack strategy against MS DRBG or similar RSA-based generators. Such gaps between “best reduction” and “best attack” are hardly unknown in the literature. We argue, however, that the history and provenance of the MS DRBG standard make it worthy of a closer look.

Our Results. We study both the implications and potential impact of backdoored parameters for the Micali-Schnorr generator and for the RSA PRG, a closely related design that was never standardized. To our knowledge, we are the first to identify vulnerabilities in these algorithms from this perspective in the literature.

First, we observe that the security of the Micali-Schnorr PRG is not tightly bound to the difficulty of inverting RSA. We show that the Micali-Schnorr construction remains secure even if one replaces RSA with a publicly evaluatable PRG or an ideal (and efficiently invertible) random permutation. While these results appear obvious in hindsight, they do not appear to have been articulated in the literature on RSA-based generators nor to have been known to the standards bodies. These observations imply that any cryptographic backdoor must somehow exploit the algebraic structure of RSA, rather than an attacker's ability to merely invert RSA or the presence of secret keys. We exhibit two such backdoors in related constructions: a family of exploitable parameters for the RSA PRG, and a second vulnerable construction for a finite-field variant of Micali-Schnorr. We also observe that the parameters allowed by the ISO standard are incompletely specified, and allow insecure choices of exponent. Several of our backdoor constructions make use of lattice techniques, in particular multivariate versions of Coppersmith's method for finding small solutions to polynomials modulo integers. We evaluate the impact of our attacks in the context of network protocols and find that the ISO weak exponent vulnerability would be exploitable in the context of IPsec.

Ultimately, although we were unsuccessful in fully solving the question we set out to answer, that of either finding an efficiently exploitable backdoor for the Micali-Schnorr generator or ruling out the possibility, we hope that this work will bring more attention to this unsolved problem and point the way to potentially fruitful cryptanalytic advances.

1.1 Technical Overview

Our goal in this work is to evaluate the hypothesis that the standardization of ISO/ANSI MS DRBG may represent an intentional attempt to subvert cryptographic systems. This possibility is intriguing for two different reasons: first, a better understanding would offer new historical context on the development of public-key standards and the intentions of nation-state cryptologic agencies. From a technical perspective, detailed investigation of this question might uncover the existence of heretofore non-public attacks on public-key cryptosystems. While factoring-based PRG constructions are some of the earliest work in this area, they appear to have received surprisingly little attention from a modern perspective, and this question shines new light on the tightness of the connection between the hardness assumptions and the security of the output.

Our approach is to assume the worst case: that the authors of the standard retained the factorization of each recommended modulus (or maliciously generated these moduli), and additionally possessed techniques that enabled them to practically exploit this knowledge. From this starting point we then attempt to "re-derive" the necessary techniques and to evaluate whether they can be used in practical settings. Our primary focus in this work is on techniques that enable state recovery given some quantity of generator output, since knowledge of the internal generator state would enable a passive attacker to obtain future generator output and possibly compromise the security of encryption protocols.

Indistinguishability, State Recovery and Backtracking Resistance. Like most standardized generators, MS DRBG uses an iterated construction. The generator is initially seeded with a state s_0 that is updated through application of a purported one-way function. The same function is also used to produce output bits. As with any PRG, knowledge of previous outputs must not provide an attacker with a meaningful advantage in predicting future output bits. This can only be achieved if all internal states remain secret until the generator is reseeded, since knowledge of any state permits the prediction of all future states and outputs. Both ISO and ANSI require an even stronger security property: the compromise of any intermediate state s_i must not enable prediction or distinguishing of generator outputs from *previous* cycles. We show that neither MS DRBG nor RSA PRG achieve this property when the factorization of N is known.

Eliminating the Obvious. A natural approach for a subversion attacker² to break these PRGs is to simply invert the RSA function to recover some internal generator state after observing generator output. However, in MS DRBG (as well as more traditional RSA and Rabin-based generators) this direct approach fails because the construction does not output all bits of RSA function output.

While the security assumptions that underlie the security proof for MS DRBG [51] are clearly false if the factorization of the modulus N is known, straightforward attempts to reverse the security reduction are also futile, for a similar reason. Micali and Schnorr’s reduction relies on two elements: a novel indistinguishability assumption for partial RSA outputs (repeated herein as Assumption 1), which is combined with rejection sampling and the ACGS algorithm of Alexi, Chor, Goldreich, and Schnorr [5] (in Theorem 1) to reduce this to the hardness of the RSA problem. An adversary who can falsify Assumption 1 still does not obtain all bits of the ciphertext from the generator’s output. Even when the attacker has access to an inversion oracle, both sides of the reduction are efficient to solve, so even reversing this portion of the reduction is unlikely to lead to a practical subversion attack. Moreover, the running time of this reduction algorithm for parameters of practical interest is much more expensive than simply brute forcing the unknown state would be, which makes Theorem 1 vacuous in the case of MS DRBG.³

Eliminating Black Box Attacks. Given the above, we turn our attention to whether being able to falsify Assumption 1 in a black box way suffices for a distinguishing attack on MS DRBG. Stated broadly, this assumption implies that for some length-expanding function $F : \{0, 1\}^k \rightarrow \{0, 1\}^n$ with $k \ll n$, the ensemble $\{b_1, b_2, \dots, b_h\}$ is indistinguishable from random, when each $b_i \leftarrow F(s_{i-1}) \bmod 2^{n-k}$ and $s_i \leftarrow \lfloor F(s_{i-1})/2^{n-k} \rfloor$ (with s_0 a random k -bit string, and h an arbitrary number of outputs). It is relatively easy to argue this assumption holds when F is itself a PRG (or is modeled as a random function or permu-

² An attacker with backdoor information; see Sect. 5.1.

³ Naturally ISO has specified parameters so that brute force and collision attacks against the state are infeasible.

tation on $\{0, 1\}^n$) as we prove in Sect. 4. We observe that the MS PRG construction instantiated with such a random permutation remains secure even when the inverse oracle is available to an adversary, or when the function contains no secrets or keys. Indeed, this construction is analogous to several common PRG constructions based on hash functions. The theorems we prove are straightforward, and their implications are obvious in retrospect, but they encode observations about the security of the MS DRBG construction that do not appear to have been formalized in the literature.

The main question then is whether such an assumption can hold when F is realized via the RSA function in the unusual setting where the factorization is known. These results suggest that any attacks on the MS or RSA PRGs must exploit *algebraic* properties of RSA and modular exponentiation, rather than being able to take advantage of factorization in a black-box way.

Algebraic Attacks. After eliminating the possibility of an “obvious” factoring-based backdoor, we give several candidate backdoor constructions and algorithmic weaknesses for RSA PRG and MS PRG that exploit algebraic properties of modular exponentiation. Most of our state recovery attacks make use of lattice-based techniques, in particular variants of multivariate Coppersmith’s method. A straightforward application of Coppersmith-type methods to break MS and RSA PRG is ruled out by the ISO parameters; such attacks seem unlikely to allow recovery of a state larger than n/e bits for generic parameters (for RSA exponent e), while the ISO parameters set the state size at $2n/e$ bits.

We give backdoor constructions that introduce additional structure that results in feasible attacks that work beyond these known bounds. For RSA PRG, we show how to efficiently generate RSA moduli that embed cyclic and linear recurrence relations in the PRG output and how to hide these recurrences in the factorization of N ; we also give an algorithm exploiting these recurrences for an efficient full state recovery attack from parameter ranges that were not known to be previously exploitable. Unfortunately, extending this idea to MS PRG does not seem to result in an efficiently exploitable backdoor without some further structure that allows us to simplify the exponentially many polynomial terms generated by the recurrence relations. We illustrate such an algebraic structure by showing that a variant of MS PRG defined over small-characteristic finite fields is trivially broken by using the linearity of the Frobenius endomorphism.

Finally, we show that the existence of RSA decryption exponents allows the efficient generation of apparently unnoticed weak exponent choices for MS PRG that are not ruled out by the parameters in the ISO standard. These weak exponents allow an efficient state recovery attack for the output lengths and state sizes in the ISO standard from a single PRG output block. We further observe that if this PRG were used to generate nonces and secret keys in the IPsec protocol (as Dual EC was in real-world implementations), this attack would allow an efficient state recovery attack from a single handshake nonce generated from raw PRG output.

Future Directions. The problem of designing an efficient backdoor for the Micali-Schnorr scheme has floated around the cryptographic community as an open problem since at least 2013 [37], with little success.

In this work, we rule out some obvious-seeming approaches that are dead ends, and illuminate some potentially fruitful directions for future exploration. In the end, we leave the question of identifying or ruling out an efficiently exploitable general backdoor in the Micali Schnorr algorithm unsolved in this paper. Ultimately, a solution to this problem may involve new ideas in the cryptanalysis of RSA.

2 Background

The ISO, NIST, and ANSI standards refer to the (pseudo)random number generation algorithms they describe as *random bit generators* (RBGs) and to deterministic pseudorandom number generation algorithms as *deterministic random bit generators* (DRBGs).

ISO-18031 lists a number of (informal) security requirements for RBGs. We list the most relevant of these below, and mark the exact text from the standard in quotes. We have given names to these properties for future convenience.

Indistinguishability “Under reasonable assumptions, it shall not be feasible to distinguish the output of the RBG from true random bits that are uniformly distributed.” Indistinguishability has been extensively studied in the academic literature [14, 65].

State Compromise Resistance “The RBG shall not leak relevant secret information (e.g., internal state of a DRBG) through the output of the RBG.” While indistinguishability is the main requirement for a PRG in the academic literature, the focus of many practical attacks is a full state compromise [17, 18, 20].

The following properties⁴ are listed as optional requirements:

Backtracking Resistance “Given all accessible information about the RBG (comprising some subset of inputs, algorithms, and outputs), it shall be computationally infeasible (up to the specified security strength) to compute or predict any previous output bit.”

Prediction Resistance “Given all accessible information about the RBG (comprising some subset of inputs, algorithms, and outputs), it shall be infeasible (up to the specified security strength) to compute or predict any future output bit at the time that [prediction resistance] was requested.”

⁴ The standard calls these properties “backward secrecy” and “forward secrecy”, but in the opposite way from the academic literature; we rename these properties to avoid confusion.

We will use the term “PRG” to refer to the algorithms in this paper, and the term DRBG when we specifically reference the ISO standard. There have also been extensive academic efforts to formalize requirements for pseudorandom number generation under various attack models including recovery from state compromise [28].

2.1 The RSA PRG

The ISO standard introduces MS DRBG as a variant of the “so-called RSA generator”, which iteratively applies RSA encryption to a starting seed to generate a sequence of states and outputs some least significant bits of each state as output. Micali and Schnorr [50] name this algorithm “the ‘incestuous’ generator”.

We summarize this algorithm in Algorithm 1 below. Let N be an integer RSA modulus of length $\lg N = n$. Let k be the length of the PRG output on a single iteration, so we have $k < n$. The length of the state is $r = n$ bits. e will be a positive integer that is the RSA exponent; in order for (N, e) to be valid RSA parameters e should be relatively prime to $\varphi(N)$. (The choice $e = 2$ is the Rabin generator used in Blum-Blum-Shub [13].)

Algorithm 1: RSA PRG

Input : A number of iterations h

Output: hk pseudorandom bits

- 1 Sample initial state $s_0 \xleftarrow{\$} [1, N]$ using truly random coins.
 - 2 **for** $i \leftarrow 1$ **to** h **do**
 - 3 $s_i \leftarrow s_{i-1}^e \bmod N$
 - 4 $b_i \leftarrow s_i \bmod 2^k$
 - 5 **end**
 - 6 Output the concatenation $b_1 || b_2 || \dots || b_h$.
-

There is a simple formula for the i th state in terms of the initial state:

$$s_i \equiv s_0^{e^i} \bmod N$$

Output Length. There are several choices of output lengths discussed in the literature on this PRG.

- $k = \lg n$. Micali and Schnorr [51] show that the RSA PRG is secure for $k = \lg n$ bits of the RSA function, based on the Alexi-Chor-Goldreich-Schnorr theorem [5] that these bits are hardcore.
- $k = 1$. Fischlin and Schnorr [33] improve the running time of the ACGS reduction and use this running time to propose concrete parameters of $k = 1$ bit of output with a 1000-bit modulus N .
- $k = (1/2 - 1/e - \epsilon - o(1))n$. Steinfeld, Pieperzyk, and Wang [63] prove the security of outputting more bits under the hardness of improving on the Coppsmith bound for solving polynomials modulo RSA moduli [21].

2.2 The Micali-Schnorr PRG (MS PRG)

Like the RSA PRG, the Micali-Schnorr PRG iteratively applies the RSA function, but it separates the bits used to generate the next state from the bits that are output. It splits each RSA output into its most and least significant bits. The least significant bits become PRG output, while the most significant bits become the RSA input for the PRG’s next iteration.

Let N be an integer RSA modulus of length $\lg N = n$. Let k be the length of the PRG output on a single iteration, so we have $k < n$. r will be the internal state length; for the Micali-Schnorr algorithm we have $r = n - k$. e will be a positive integer that is the RSA exponent; this is specified so that e is relatively prime to $\varphi(N)$.

Micali-Schnorr as Published in Their Papers. There are two published versions of the Micali-Schnorr paper. The first version, “Efficient, Perfect Random Number Generators,” appeared in CRYPTO ’88 [50]. The journal version of the paper, “Efficient, Perfect Polynomial Random Number Generators,” was published in the Journal of Cryptology in 1991 [51].

The algorithm for the Micali-Schnorr PRG as published in the original paper takes as input a requested number of iterations h and outputs hk pseudorandom bits. Micali and Schnorr refer to this construction as the “sequential polynomial generator of the weaning type.”

Algorithm 2: The Micali-Schnorr algorithm

Input : A number of iterations h

Output: hk pseudorandom bits

1 Sample initial state $s_0 \xleftarrow{\$} [1, N2^{-k}]$ using truly random coins.

2 **for** $i \leftarrow 1$ **to** h **do**

3 $z_i \leftarrow s_{i-1}^e \pmod N$

4 $b_i \leftarrow z_i \pmod{2^k}$

5 **if** *version from [50]* **then**

6 $s_i \leftarrow \lfloor z_i 2^{-k} \rfloor + 1;$

7 **else if** *ISO-18031 version* **then**

8 $s_i \leftarrow \lfloor z_i 2^{-k} \rfloor;$

9 **end**

10 Output $b_1 || b_2 || \dots || b_h$.

Output Length. In the original paper, Micali and Schnorr discuss the following choices for k and n .

- $k = O(\lg n) = O(\lg \lg N)$ They list this as a suitable choice for the PRG. The algorithm in Micali and Schnorr’s reduction runs in time polynomial in $2^k n \epsilon^{-1}$ so if the PRG is insecure for $k = O(\lg n)$ then this gives a polynomial time RSA decryption algorithm.

- $k = O(n^{1/3})$ They argue that this choice is suitable by comparing the resulting reduction time to the running time of the number field sieve for factoring, and note that if the algorithm is insecure for this parameter, it would beat the number field sieve.
- $k = n(1 - 1/e)$ is clearly insecure. For this choice, the security proof does not apply because $s_0^e < N$, so no mod operation occurs and “RSA” decryption is easy in this case.
- $k = n(1 - 2/e)$ is left as an open question, but Micali and Schnorr promote this choice as one that would produce an efficient generator if indeed it is secure. The security of this choice is based on the indistinguishability of RSA encryptions of short ($n - k$ -bit) plaintexts from random integers modulo N . This is the value chosen by the ISO standard.

D.2 Default moduli for the MS_DRBG ()

D.2.1 Introduction to MS_DRBG default moduli

Each modulus is of the form $n = pq$ with $p = 2p_1 + 1$, $q = 2q_1 + 1$, where p_1 and q_1 are $(\lg(n)/2 - 1)$ -bit primes.

D.2.2 Default modulus n of size 1024 bits

The hexadecimal value of the modulus n is:

```
b66fbfda fbac2fd8 2eb13dc4 4fa170ff c9f7c7b5 1d55b214 4cc2257b 29df3f62
b421b158 0753f304 a671ff8b 55dd8abf b53d31ab a0ad742f 21857acf 814af3f1
e126d771 a61eca54 e62bfd5 85c311b0 58e9cd3f aab758a5 e2896849 6ec1dd51
d0355aa1 55d4d912 6140dcfa b9b03f62 a5032d06 536d8574 0988f384 27f35885
```

D.2.3 Default modulus n of size 2048 bits

Fig. 1. A portion of ISO 18031 Appendix D.2 with the default 1024-bit modulus [43].

ISO/IEC 18031 Micali-Schnorr. The Micali-Schnorr algorithm was standardized in ISO/IEC 18031 as a deterministic random bit generator, named MS DRBG, alongside the Dual EC DRBG design. Dual EC was removed from ISO 18031 in a 2014 Technical Corrigendum.

The version of the Micali-Schnorr algorithm that appears in ISO/IEC 18031 [43] differs in one minor respect from the academic publication. Specifically, it alters line 6 so that $s_i \leftarrow \lfloor z_i 2^{-k} \rfloor$. (That is, it does not increment the result.) It isn’t clear what effect, if any, this change has on the security of the scheme, and it is not documented in either publication. In personal communication, Micali told us he does not recall the reason for the original choice or the change.

Output Length. The ISO standard requires that the output length satisfies $8 \leq k \leq \min(n - 2\gamma, n(1 - 2/e))$, where γ is the target security level. The default k is the largest value this inequality allows, rounded down to a multiple of 8.

The introduction to MS DRBG (section C.4.3.1) applies very different output security bounds to RSA PRG than MS DRBG: it states that the $k = \lg \lg N$ least significant bits the RSA generator outputs “are (asymptotically in N) known to

be as secure as the RSA function f . The Micali-Schnorr generator `MS_DRBG()` uses the same e and N to produce many more random bits per iteration, while eliminating the reuse of bits as both output and seed” [43].

Modulus and Exponent Generation. The ISO standard states that implementations “shall” permit either an implementation-generated “private modulus” or the use of one of the default moduli in the standard (see Fig. 1). The standard requires the length of the modulus to conform to the requested security strength: a 1024-bit N for $\gamma = 80$; 2048 bits for $\gamma = 112$; 3072 for $\gamma = 128$; 7680 for $\gamma = 192$; and 15360 for $\gamma = 256$.

The ISO standard also specifies that custom RSA moduli should be generated so that $p - 1$, $p + 1$, $q - 1$, and $q + 1$ have a prime factor of at least γ bits. It also states that the default moduli have been generated so that $p = 2p_1 + 1$, $q = 2q_1 + 1$ for p_1, q_1 primes, and that $p + 1$ and $q + 1$ have “the required large prime factor”.

It is also required that e be relatively prime to $(p - 1)(q - 1)$. While this is necessary for these to be well-defined RSA parameters, the decryption exponent $d = e^{-1} \bmod (p - 1)(q - 1)$ is never used in the normal course of random number generation, so it is not clear why this requirement needs to be present.

The default moduli are stated to have “strong” primes as factors, which “essentially guarantees” that $\varphi(N)$ will be relatively prime to odd e , but the factorization of these default moduli is not given, so users are unable to verify this for themselves when using the default moduli with user-generated e . If a user-generated exponent is not supplied, the default $e = 3$ is used.

Backtracking and Prediction Resistance. The standard states: “[backtracking resistance] is inherent in the algorithm, even if the internal state is compromised.” This is not true against an adversary who knows the factors of the modulus: if the state and all but a few previous output bits are compromised, the adversary can learn the remaining bits by decrypting the candidate z_i values and checking whether the result is less than $N/2^k$. The standard ensures prediction resistance by requiring the implementation to reseed every 50,000 outputs.

ANSI X9.82. The Dual EC and MS DRBG algorithms were the two number-theoretic (public-key cryptographic) PRG designs promoted by the NSA for inclusion in ANSI X9.82 [45]. A version of MS DRBG was present in early drafts of the ANSI X9.82 specification from 2004 until it was removed in August 2005.

The text of the entire X9.82 DRBG specification is largely identical to the standard ultimately published by ISO in 2005 [43]. A number of draft versions of the X9.82 standard as well as internal discussions and documentation have been made available as part of a FOIA request from NIST in 2014 and 2015 [52], which provide interesting insights into the development of the standard. There is evidence that this text was written by the US government [11].

There appear to have been differing views among committee members on the number of bits that should be output from MS PRG. Early proposals suggested outputting far fewer bits than the ISO version ultimately standardized. A set of

2004 slides from the NSA at a NIST workshop suggested outputting only the “hardcore” bits for each modulus size, which are provably as hard to predict as the entire state under the RSA assumption. For a 1024-bit RSA modulus the suggestion was 10 bits, and for 2048 and 3072-bit moduli the suggestion was 11 bits [45].

An undated (but apparently early) draft of X9.82 includes the comments “The MS generator allows a much larger percentage of N bits to be used on each iteration, and has an additional advantage that no output bits are used to propagate the sequence. (It does, however, rely on a stronger assumption for its security than the intractability of integer factorization.) As the X9.82 standard evolved, committee members argued for restricting the number of bits generated on each exponentiation to $O(\lg \lg N)$ *hard* bits, as is done in Blum-Blum-Shub. The result is that the efficiency argument for choosing MS over BBS doesn’t apply. Nonetheless, a user does have more options in the choice of parameters” [2]. Later drafts of the text mention only the larger output lengths ultimately adopted by ISO.

The sole comments we have located that justify the decision to drop Micali-Schnorr from the standard come from a document titled “DRBG recommendations from the X9.82 Editing Group” which states “We recommend keeping DUAL_EC_DRBG. Despite the fact that it is much slower than the other DRBGs, it offers a third distinct technology that can serve as a hedge against breakthroughs in cryptanalysis of hashes and block ciphers. . . . We suggest dropping HASH_DRBG and the MS_DRBG, as well as support for the other NIST curves in the DUAL_EC_DRBG” [1].

2.3 Related Work

Backdoored Random Number Generation. In addition to works cited in the introduction, a long line of literature considers the possibility of *algorithm substitution attacks* (previously referred to as “subversion” and “kleptographic”) attacks [8, 10, 19, 31, 56, 60, 66, 67]. To formalize this work into the setting of PRGs, Dodis et al. [27] give a formal treatment and prove that such schemes are equivalent to public-key encryption schemes with pseudorandom ciphertexts. Degabriele et al. [25] extend these results to consider backdoored PRNGs (which unlike PRGs may take additional inputs for prediction resistance).

Backdoored RSA Parameters. There is a surprisingly long line of work on generating “backdoored” RSA parameters [7, 16, 23, 46, 54, 64, 66–68]. These works focus on trapdoors that admit efficient factorization or recovery of private keys given only a public key (N, e) . In contrast, our work begins from the assumption that the adversary possesses the factorization of N and addresses the problem of compromising an algorithm using this knowledge.

MS DRBG. Some previous works have considered MS DRBG. Fouque, Vergnaud, and Zapalowicz give a time/memory tradeoff for recovering the state

faster than brute force [34], and Fouque and Zapolowicz study the statistical distance of short RSA [35]. In a 2013 blog post Matthew Green posed the problem of finding a practical attack against MS DRBG when the factors are known [37]. Antonio Sanso suggested in a 2017 blog post that Mersenne or other special-form primes might lead to a backdoor in Micali-Schnorr [61]. Lynn Engelberts extended Sanso’s analysis of special form primes in a 2020 masters thesis. [30]

Security Proofs for Number-Theoretic PRGs. The limited applicability of asymptotic security proofs on concrete parameters for number-theoretic PRGs has been studied by Koblitz and Menezes [48].

3 Security Reductions for the MS and RSA PRGs

The security reduction that Micali and Schnorr give to their “sequential” construction has two steps. First, they define a security question, which they label **Q1**. We rephrase this as an assumption below:

Assumption 1 [51] (**Q1**). *The following distributions are polynomially indistinguishable (for public e and N)*

- $(N, s^e \bmod N)$ for $s \xleftarrow{\$} [1, N2^{-k}]$
- (N, r) for $r \xleftarrow{\$} [1, N]$.

The authors next provide a polynomial-time reduction that transforms a distinguisher algorithm for the Micali-Schnorr PRG into a distinguisher for Assumption 1. The proof uses a hybrid argument. It is this assumption that is used to justify the large output sizes used in the ISO version of MS PRG. The second half of the reduction completes the reduction to the hardness of RSA inversion, and gives a security reduction for both MS PRG and RSA PRG.

Theorem 1 [51]. *Let N be an RSA modulus. Every probabilistic algorithm that ϵ -rejects ciphertexts of random messages $s \xleftarrow{\$} [1, N2^{-k}]$ can be transformed into a probabilistic algorithm for decrypting arbitrary RSA ciphertexts; this algorithm terminates after at most $(2^k \epsilon^{-1} n)^{O(1)}$ steps.*

This reduction contains several steps that are fundamentally exponential time in k , the number of bits of output. In particular, the algorithm samples 2^k messages until it expects to find one with the required number of zeros. Thus, this reduction is only polynomial time for $\lg n$ bits of output. Fischlin and Schnorr [33] have improved the running time of the ACGS algorithm [5] used in the decryption step, but the reduction remains exponential in k .

In addition to being exponential time in the output length k , running the reduction will be more expensive than simply brute forcing the unknown bits of the plaintext when $k > n/2$, that is, when the output is larger than the state. In fact, the constants hidden in the $O(1)$ result in significantly worse parameters.

The exponential cost in the reduction in the proof of Theorem 1 appears to be the reason for only outputting $\lg n$ bits of output for the RSA PRG. It is

interesting that the ISO standard accepts much more generous parameters for Micali-Schnorr output than for the RSA PRG without having attempted to find an analogously relaxed assumption that might permit more generous outputs as Steinfeld, Pieprzyk, and Wang [63] ultimately did.

Statistical Indistinguishability Results and Mod p Variants. Micali and Schnorr also consider a variant of their PRG defined modulo a prime p , and hypothesize that this variant is still secure despite the fact that their factoring-based assumptions no longer hold. The journal version of their paper [51] contains theorems proving the statistical randomness of the $(n/2 - k - (\lg n)^2)$ least significant bits of $s^e \bmod N$ for $s \leftarrow_{\mathbb{S}} [1, N^{2^{-k}}]$ when N is prime or an RSA modulus.

Fouque and Zapalowicz [35] give a more general version of this theorem for RSA moduli that applies to size bounds above \sqrt{N} and prove that the $\lg N$ least significant bits of $s^e \bmod N$ for $s < M$ for a chosen bound $M < N$ are statistically indistinguishable from uniform. They apply their bounds to Micali-Schnorr and find that asymptotically, this bound dictates that the output is not statistically indistinguishable when more than $n/3$ bits are output.

These statistical indistinguishability results provide evidence that least significant bits of modular exponentiation (modulo primes or RSA moduli) are indistinguishable from uniform at much less aggressive parameters than ISO chose. However, such statistical indistinguishability results cannot apply when the output exceeds the length of the seed. Thus they do not rule out the possibility of attacks on the PRG with long or multiple outputs.

4 Ruling Out Black-Box Attacks

In this section, we will try to make more precise the intuition that the security of MS and RSA PRG is more closely related to the assumption of pseudorandomness of RSA ciphertexts than the hardness of inverting RSA. This offers a more formal explanation for why there does not appear to be a black-box way to use an RSA decryption oracle to break the security of MS PRG.

We begin by defining a generic Micali-Schnorr-type construction, which we call MS- f -PRG. In this variant the RSA operation is replaced by some function f . That is, Step 3 of Algorithm 2 in Sect. 2.2 becomes $z_i \leftarrow f(s_{i-1})$, for f that we will instantiate below.

4.1 Micali-Schnorr Is Secure with a PRG

The Micali-Schnorr construction is still secure when instantiated with a PRG.

Theorem 2. *If $f : [1, 2^{n-k}] \rightarrow [1, 2^n]$ is a secure pseudorandom generator, then the output of MS- f -PRG is pseudorandom.*

The proof is the same as the proof of Theorem 5.1 in [51], substituting the pseudorandomness of f for Assumption 1.

While this finding is not surprising, it illustrates that the security of MS- f -PRG need not depend on any secret information. This informs how a provably secure variant of MS PRG could be instantiated, but unfortunately it does not enable a proof for the variants with RSA modulus N or prime modulus p (as discussed in the original work of Micali and Schnorr [51]). In the context of an adversary who knows the factorization of N , $f(s) = s^e \bmod N$ with short seed $s < 2^{n-k}$ is distinguishable from random: simply decrypt $f(s)$ and check if the seed is short. This function is clearly not a PRG, and so we gain no information about the security of this construction. A similar argument applies in the prime modulus case $f(x) = x^e \bmod p$.

4.2 MS PRG Is Still Secure When Implemented with a Random Permutation

We next show that the Micali-Schnorr construction is secure when the one-way RSA function is replaced with a public (invertible) random permutation. While this analysis is clearly quite artificial, it offers a useful bound on the efficiency of *generic attacks* (i.e., attacks that do not exploit special properties of the RSA function) when the factorization of N is known.⁵ This suggests that if it is indeed possible to backdoor the Micali-Schnorr construction, the backdoor must take advantage of some nontrivial algebraic property of RSA.

We begin by defining our variant of the MS PRG in which RSA encryption is replaced with a publicly accessible random permutation. Concretely, at line 3 of Algorithm 2 we replace the RSA evaluation $z_i \leftarrow s_{i-1}^e \bmod N$ with $z_i \leftarrow f(s_{i-1})$ where $f : [1, N-1] \rightarrow [1, N-1]$ is a publicly accessible random permutation. We give the attacker the ability to decrypt by making the inverse permutation f^{-1} publicly accessible. With this modification, we obtain the following theorem.

Theorem 3. *No adversary \mathcal{A} that makes q total queries to black-box oracles for random permutations f and f^{-1} can distinguish the hk -bit output string of our modified variant of Algorithm 2 from a random hk -bit string with advantage greater than $\frac{(h+1)^2}{2N} + \frac{2hq}{N2^k} + \frac{hq}{N} + \frac{2hq}{N-q} + \varepsilon$ (for some negligible ε).*

A proof of Theorem 3 can be found in the full version [24].

4.3 RSA-PRG as a Sponge

The iterative construction of RSA-PRG—apply a transformation to the state, then output a fraction of the bits—is widely used in symmetric cryptography and is known as the sponge construction. If we replace RSA encryption with a random function f in the RSA PRG construction, we can use theorems developed for cryptographic sponge constructions to obtain strong bounds on the security of the resulting construction. These theorems hold for functions that are fixed public and efficiently invertible permutations.

⁵ More critically, this construction does not have any implications for the security of MS PRG instantiated with the RSA function, since RSA encryption quite clearly behaves differently than a random function.

Theorem 4. *Let f -PRG be the RSA PRG construction except replacing the $x \mapsto x^e \bmod N$ operation with a fixed, public, efficiently invertible random permutation $f : \mathbb{Z}_N \rightarrow \mathbb{Z}_N$. Then the output of f -PRG is indistinguishable from random to any adversary that runs in time polynomial in $\lg N - k$ and has black-box access to f and f^{-1} .*

Proof. f -PRG follows the sponge construction, with state size $\lg N$, rate k , and capacity $c = \lg N - k$. As Bertoni, Daemen, Peeters, and Van Assche show in Eq. 6 of [12], the Random Oracle (\mathcal{RO}) differentiating advantage against the sponge construction when used with a random permutation is upper bounded by $((1 - 2^{-k})m^2 + (1 + 2^{-k})m)2^{-(c+1)}$, where m is the number of calls to the underlying transformation.⁶ The \mathcal{RO} -differentiation game models an adversary with query access to f and f^{-1} . Thus no $\text{poly}(\lg N - k)$ -time adversary can distinguish f -PRG from a random oracle with more than negligible probability, even with query access to f and f^{-1} .

We remark that SHAKE-128 (from the SHA-3 standard [4]) is an XOF (and thus also a PRG) constructed as a sponge with state size 1600, outputting 5/6 of the state (1344 bits) per iteration, and the function f it uses to transform its state is a fixed, public, efficiently invertible permutation. This is a smaller state size than ISO MS DRBG uses ($\lg N = 3072$ for 128-bit security), and a larger fraction of bits that are output (5/6, compared to 1/3 for ISO using the default exponent $e = 3$).

5 Algebraic Attacks

Here we present several attacks against RSA PRG and variants of MS PRG. These results are not as compelling as the attack against Dual EC, but they illustrate different properties of the algebraic structure of RSA-based PRGs that may ultimately lead to either the development or ruling out of such a backdoor in the MS PRG.

5.1 Notions of Cryptographic Subversion

A growing body of work considers *algorithm substitution attacks* (ASAs) against cryptographic implementations. In this setting, a known cryptographic algorithm is replaced with a subverted algorithm designed by an attacker, who retains secret knowledge that allows for exploitation [8, 10, 19, 31, 56, 60, 66, 67]. This effectively models our assumptions for a subversion attack on MS DRBG. We do not present formal definitions here, and refer the reader to e.g., [8, 60] for details.

The MALICIOUS Framework. Peyrin and Wang [56] describe the MALICIOUS framework that includes several informal properties required by a subverted symmetric construction such as an RNG. Inspired by their definitions, we provide the following shorthand characterization for our asymmetric backdoor constructions.

⁶ [12] requires $m \ll 2^c$, as is the case here.

Undiscoverability: An outside observer should be unable to find the hidden backdoor, even if the general form of the backdoor is known [56].

Practical Construction: The backdoor designer should be able to efficiently construct parameters that allow for backdoor exploitation.

Practical Exploitation: An attacker should be able to efficiently violate the security properties of the scheme if they know the secret information required to exploit the backdoor.

Plausible Deniability: To an external observer, the public parameters, keys, and structure of the cryptosystem should appear to be “properly” generated.

Relationship to Formal Definitions of ASAs. Bellare, Paterson and Rogaway formalized a framework for *algorithm substitution attacks* (ASAs) [8]. Informally, the framework captures the properties described above using two independent security games.⁷ In the first game, a *detection adversary* (\mathcal{D}) represents the defender, and is used to define the undiscoverability of a cryptographic backdoor. In this framework, the detection adversary is asked to distinguish between the correct algorithm (e.g., an implementation of MS DRBG in which the modulus N is generated honestly at random and no secret factorization is retained) and a second *subverted* algorithm (such as the ISO standard with attacker-known or chosen factorization). The subverted algorithm passes this test if \forall probabilistic polynomial time (PPT) algorithms \mathcal{D} , \mathcal{D} distinguishes the input/output behavior of the subverted implementation from the correct implementation with at most negligible advantage.⁸

In the second game a *subversion adversary* (\mathcal{S}) is given access to secret knowledge about the subverted algorithm (for example, knowledge of the secret factorization of the modulus N .) The minimal criteria for a subversion attack is that there must exist some PPT \mathcal{S} that distinguishes the input/output behavior of the subverted and non-subverted implementations with non-negligible advantage. If the non-subverted implementation is itself a secure PRG, then the ability to distinguish between subverted and unsubverted implementations naturally implies an attack that distinguishes the output of the subverted algorithm from random bits. In practice, subversion attackers may also be able to carry out more powerful attacks, such as state recovery and future output prediction.

5.2 Algorithmic Background: Multivariate Coppersmith’s Method

Several of our attacks make use of the following version of multivariate Coppersmith’s method. For a basic review of lattices and terminology, see [49].

⁷ The “Practical Construction” requirement is captured formally by requiring that both honest and subverted algorithm have a polynomial-time *setup* algorithm that (in the subverted case) produces the subverted implementation and secret trapdoors.

⁸ Subsequent definitions by Russell *et al.* extend this notion to one in which the detection adversary (in this work called an “online watchdog”) is also allowed to observe interactions between the implementation and an attacker. We do not consider this scenario in our work, since we primarily focus on passive eavesdropping attacks.

Review of Coppersmith’s Method. Coppersmith’s method uses lattice reduction (typically LLL [49]⁹) to find small solutions to polynomials modulo integers. For univariate polynomials, this method is fully rigorous, and has a clean bound: for a degree- d polynomial $f(x) \in \mathbb{Z}[x]$ and $N \in \mathbb{Z}$, all roots $r \in \mathbb{Z}$ satisfying $f(r) \equiv 0 \pmod N$ can be found for $|r| < N^{1/d}$ in polynomial time in d and $\lg N$ [21].

The multivariate generalization of this method that we need for our attacks does not have a clean theorem statement, and in fact a fully rigorous generalization cannot exist [22]. Nevertheless, a heuristic generalization of this method often works in practice ([47], see also [44]), and it is this heuristic version that we will use. We will derive the relevant bounds using ad hoc, problem-specific constructions.

The following lemma tells us the condition under which we expect to succeed.

Lemma 1. *Let $\{f_i(\mathbf{x})\}_{i=1}^w$ be integer polynomials in m variables $\mathbf{x} = (x_1, \dots, x_m)$ and let $N \in \mathbb{Z}$. We wish to find one or more solutions $\mathbf{r} = (r_1, \dots, r_m)$ simultaneously satisfying $\{f_i(\mathbf{r}) \equiv 0 \pmod N\}_{i=1}^w$.*

If we can find m auxiliary polynomials Q_1, \dots, Q_m such that

$$Q_j(r_1, \dots, r_m) \equiv 0 \pmod{N^t} \quad \text{and} \quad |Q_j(r_1, \dots, r_m)| < N^t$$

for some integer $t \geq 1$ then each Q_j satisfies $Q_j(r_1, \dots, r_m) = 0$ over the integers. If in addition the Q_j are algebraically independent, then we can solve for a bounded number of possible solutions.

We sketch a general method to solve this problem in Algorithm 3 below. The details of Step 3 are application dependent; we elaborate on this in the full version [24].

The value t and choice of polynomial shifts $x_1^{a_1} \dots x_m^{a_m}$ in Step 1 of Algorithm 3 are chosen as part of the optimization process. We will refer to t as the *multiplicity* of the roots. The lattice dimension is determined by the number of distinct monomials in the set of polynomials $\{g\}$ used to generate the lattice. In general, the dimension is exponential in the number of variables.

To apply Lemma 1 we bound $|g(\mathbf{r})| < |\sigma(g)|_1$, so we want to find m vectors in the lattice whose ℓ_1 norms are less than N^t . For a random lattice L , the successive minima $\lambda_i(L)$ often have close to the same length, and in practice LLL [49] typically finds vectors of length $1.02^{\dim L} (\det L)^{1/\dim L}$ or $1.02^{\dim L} \lambda$ [53].

These vectors are guaranteed to be linearly independent as coefficient vectors, but the corresponding polynomials are not guaranteed to be algebraically independent. Nevertheless, the polynomials found by this algorithm for random problem instances with optimal parameters are often algebraically independent.

We expect the algorithm to succeed when Condition 1 is satisfied.

⁹ Given a lattice, the LLL algorithm computes in polynomial time a basis whose vectors satisfy heuristic and provable length bounds.

Algorithm 3: Multivariate Coppersmith Method (Sketch)

- Input** : $\{f_i(\mathbf{x})\}_{i=1}^w \in \mathbb{Z}[x_1, \dots, x_m]^w, N \in \mathbb{Z}, \{R_j\}_{j=1}^m$
Output: $\{r_j\}_{j=1}^m$ satisfying $|r_j| < R_j$ and $f_i(\mathbf{r}) \equiv 0 \pmod N$
- 1 Generate a basis of auxiliary polynomials of the form

$$g_{a,b}(\mathbf{x}) = \left(\prod_j x_j^{a_j}\right) \left(\prod_i f_i^{b_i}\right) N^{t-\sum_i b_i}.$$
 - 2 Map each polynomial to a scaled coefficient vector embedding:

$$\sigma : g(\mathbf{x}) = \sum_i g_i x_1^{c_{i,1}} x_2^{c_{i,2}} \dots x_m^{c_{i,m}} \mapsto (g_1 R_1^{c_{1,1}} \dots R_m^{c_{1,m}}, \dots)$$
 - 3 Construct a lattice basis B of coefficient vector embeddings $\sigma(g)$ for a carefully chosen subset of the g s generated in step 1.
 - 4 LLL-reduce the lattice basis.
 - 5 Construct a Gröbner basis of the polynomials $\sigma^{-1}(v)$ of all vectors v in the reduced basis whose ℓ_1 norms $|v|_1$ are shorter than N^t .
 - 6 Enumerate the candidate solutions given by the Gröbner basis and verify whether each is a valid solution for the r_i .

Condition 1 (Heuristic Multivariate Coppersmith) *Algorithm 3 will heuristically find all suitable roots if the basis for lattice L constructed in Step 3 of Algorithm 3 satisfies*

$$1.02^{\dim L} (\det L)^{1/\dim L} < N^t.$$

Applying Coppersmith’s Method to MS and RSA PRG. It is tempting to try to apply a multivariate Coppersmith approach directly to MS or RSA PRG to carry out a state recovery attack. In particular, such an attack involves finding a small solution of a degree- e polynomial modulo N , which is precisely the problem that Coppersmith-type methods solve.

In this section, we will sketch this attack and observe that it is ruled out by the parameter choices made by ISO for MS.

MS PRG. An attempted state recovery attack from two outputs would start from the polynomial relations between the unknown states s_i :

$$\begin{aligned} s_0^e - 2^k s_1 - b_1 &\equiv 0 \pmod N \\ s_1^e - 2^k s_2 - b_2 &\equiv 0 \pmod N \end{aligned}$$

Let $|s_i| < R$. Construct the lattice basis

$$B = \begin{bmatrix} R^e & 0 & -2^k R & 0 & -b_1 \\ 0 & R^e & 0 & -2^k R & -b_2 \\ 0 & 0 & NR & 0 & 0 \\ 0 & 0 & 0 & NR & 0 \\ 0 & 0 & 0 & 0 & N \end{bmatrix}$$

We have $\det L(B) = R^{2e+2} N^3$ and $\dim L(B) = 5$. Omitting approximation factors in such small dimension, and setting $t = 1$, Condition 1 tells us we expect to succeed if

$$(\det L(B))^{1/\dim L(b)} = (R^{2e+2}N^3)^{1/5} < N$$

which applies when $R < N^{1/(e+1)}$. In other words, the bit length of the state size $r = n - k$ should satisfy $r < n/(e + 1)$. Attempted improvements from higher degree polynomials and root multiplicities seem to give the same bound, even if more than two outputs are available.

This attack is ruled out by the choice of ISO parameters $r = 2n/e$. This makes sense because this attack does not even require the factorization of N .

RSA PRG. Steinfeld, Pieprzyk, and Wang [63] do a similar analysis of RSA PRG and obtain a heuristic bound of $r < n/(e + 1)$ for the unknown portion of the state. Herrmann and May [39] improve this to n/e when the PRG outputs the most significant bits of the state.

Simpler attacks when $r < n/e$. A state size bound of $r < n/e$ is a degenerate case for both RSA and MS PRGs, since there is no modular reduction performed when computing $s_i^e \bmod N$. Fouque, Vergnaud, and Zapalowicz point out that one can recover the state via Hensel lifting [34].

5.3 Attacks on RSA PRG

In this section, we show how to construct backdoor parameters for the RSA PRG. The states (and thus the output) generated by the RSA PRG have an iterative structure that cycles modulo a divisor of $\varphi(\varphi(N))$. This means that an attacker who can control the generation of N and e can embed a chosen relationship among outputs that enables efficient distinguishing and state recovery attacks.

e has Short Period (eSP) attack mod $\varphi(\varphi(N))$ In our first backdoor construction, we show that it is possible to efficiently construct RSA parameters for which the output of RSA PRG produces extremely short cycles. This violates indistinguishability, but would be observable by any attacker. We then show that it is possible to somewhat obscure the most obvious cyclic behavior in the output, which leads to an efficiently generatable and exploitable backdoor. While this behavior alone doesn't lead to a fully undiscoverable backdoor, it provides intuition for a more sophisticated backdoor we construct later in the section.

Recall that the multiplicative order of an integer modulo N is a divisor of $\varphi(N)$. In the RSA-PRG generator with modulus N and exponent e , we have state s_i satisfying $s_i = s_0^{e^i} \bmod N$. Micali and Schnorr (as well as Blum, Blum, and Shub [13]) note that the period of the sequence of outputs generated by s_0 will thus be a divisor of $\varphi(\varphi(N))$ [51]. They say that “in general” the period “will be a large factor of $\varphi(\varphi(N))$ and will be much larger than \sqrt{N} which is the average period of a random recursion in \mathbb{Z}_N . It is conceivable that the number $\varphi(\varphi(N))$ somewhat affects the output distribution of the generator and not only its period.” They do not appear to have considered the possibility of malicious parameter generation.

In particular, the period can be made a *small* factor of $\varphi(\varphi(N))$. Suppose e generates a small subgroup of $\mathbb{Z}_{\varphi(N)}^*$. That is, suppose $e^j \equiv 1 \bmod \varphi(N)$ for some small j . Then $s_j = s_0^{e^j \bmod \varphi(N)} = s_0$, and PRG cycles with period j . A

similar technique is possible when e shares a factor with $\varphi(N)$, we describe it in the full version [24]. One algorithm, given a specific e , to find primes p such that e has small order modulo $\varphi(p)$ is shown in Algorithm 4.

Algorithm 4: Constructing prime p s.t. e has small order mod $\varphi(p)$.

Input : An integer e

Output: A prime p such that e has small order modulo $\varphi(p)$.

- 1 Choose a cycle length ℓ .
 - 2 Compute small prime factors p_i of $e^\ell - 1$ using the elliptic curve method or other factoring methods that are efficient for small factors.
 - 3 Choose a subset of the p_i (and optionally also the composite cofactor) computed in the previous step, and check if $1 + \prod_i p_i$ is prime. (In order to generate odd primes, we will need one of the p_i to be 2.)
-

This algorithm is reasonably efficient in practice for parameters of interest. For example, for $e = 5$ and $\ell = 504$, it took 10s on a laptop with a dual-core Intel i7-6500 CPU to find an 880-bit prime with these properties using Sage with ECM for factorization, aborting factorization when it started to get slow. We tried a few candidates for e and ℓ within this range. To generate a hard-to-factor modulus N , one could generate two primes using this algorithm.

Partially Hidden Cycling Behavior. Cycles in the output would be easy to notice for any user who generates enough outputs, so the previous construction would be easily discoverable. But if e generates cycles of length $\ell \bmod p$, but not mod q , the outputs would not have as obvious cycling behavior to the end user. By choosing q such that $e^\ell \equiv c_q \pmod{\varphi(q)}$ for some c_q small enough that finding roots mod q of degree c_q polynomials is feasible, an adversary who knows the factorization and observes the sequence of outputs can efficiently recover the full state, as we will show in Theorem 5. Such a q can be generated similarly to Algorithm 4, but factoring $e^\ell - c_q$ instead of $e^\ell - 1$.

With such parameters, an attacker can recover the full state using only the first and $(\ell + 1)$ th PRG outputs, assuming each output has length $k \geq n/2$. (If RSA PRG parameters were set following the ISO parameters for MS DRBG, this will be the case for all $e \geq 5$.) The attack is given in Algorithm 5.

As a proof of concept, we generate (for public exponent $e = 5$) a 2048-bit backdoored $N = pq$ such that $5^{504} \equiv 1 \pmod{p-1}$ and $5^{504} \equiv 187 \pmod{q-1}$. We include it in the full version [24]. The state recovery attack (implemented in Sage) using this modulus took 31 s.

However, while practical to construct and practical to exploit, this “backdoor” is not undiscoverable (regardless of how q is generated) because a user could exploit the relationships modulo p to efficiently factor N as follows. They choose an initial state s_0 , and compute the sequence of states $s_i = s_{i-1}^e \pmod{N}$. If they discover a state s_ℓ where $\gcd(s_\ell - s_0, N)$ is nontrivial, then they can use this to factor N . (Note that this algorithm is similar to the Pollard rho algorithm but with a different “pseudorandom” walk.)

Algorithm 5: eSP attack in the “partially hidden cycle” case.

- 1 Let b_1 and $b_{\ell+1}$ be two outputs. Without loss of generality assume $s_{\ell+1} \geq s_1$.
 - 2 Observe that $b_{\ell+1} - b_1 \equiv (s_{\ell+1} - s_1) \pmod{2^k}$. Observe further that $s_{\ell+1} - s_1 \equiv 0 \pmod{p}$, and $0 \leq (s_{\ell+1} - s_1)/p < q < 2^k$.
 - 3 Let $m = (b_{\ell+1} - b_1)p^{-1} \pmod{2^k}$, reduced such that $0 \leq m < 2^k$. Now $m = (s_{\ell+1} - s_1)/p$ as integers.
 - 4 Solve the polynomial congruence $y^{c_q} - y - pm \equiv 0 \pmod{q}$. (Recall $c_q \equiv e^\ell \pmod{\varphi(q)}$.) (The degree c_q is small and q is prime, so this is feasible.)
 - 5 One of the roots will be $s_1 \pmod{q}$, because \pmod{q} we have $s_1^{c_q} \equiv s_1^{e^\ell} \equiv s_{\ell+1} \equiv pm + s_1$.
 - 6 For each root α , use CRT to recover $\tilde{s}_1 \in [0, 2^k q)$ such that $\tilde{s}_1 \equiv b_1 \pmod{2^k}$ and $\tilde{s}_1 \equiv \alpha \pmod{q}$, and check whether \tilde{s}_1 would produce correct outputs b_i .
 - 7 Since $s_1 < pq < 2^k q$, when $\alpha = s_1 \pmod{q}$, we will get $\tilde{s}_1 = s_1$ over the integers.
-

We summarize these results in the following informal theorem:

Theorem 5. *An attacker can efficiently generate RSA parameters $(N = pq, e)$ such that e has a chosen (small) order $\ell > \log_e \varphi(p)$ modulo $\varphi(p)$. This attacker can then carry out an efficient state recovery attack after observing at least $\ell + 1$ RSA PRG outputs of length $k \geq n/2$ bits generated using these parameters. This backdoor has efficient parameter generation and exploitation but is discoverable.*

These constructions can be extended in a straightforward manner in the case where N is a multi-prime or unbalanced RSA modulus.

While these ideas do not generate a fully satisfactory backdoor, they provide intuition for the construction in the next section, where we will replace the relationship $e^\ell \equiv 1 \pmod{\varphi(N)}$ (or $\pmod{\varphi(p)}$) with a more complex polynomial.

The SUS Backdoor for RSA-PRG. In order to conceal the discoverable cyclic behavior of the eSP backdoor, we augment this idea to generate moduli that embed a small, sparse polynomial relationship satisfied by the exponent e modulo $\varphi(N)$ or $\varphi(p)$. We will call this “Small Unknown Solution” or SUS.

Parameter Generation. Let $f(x) = \sum_{i \in S} c_i x^i$ be a sparse polynomial that will remain secret, where the c_i are all ± 1 and are roughly balanced. A correspondingly backdoored prime p will satisfy the relation $f(e) \equiv 0 \pmod{p-1}$. This implies a relation between PRG states $\prod_{i \in S} s_i^{c_i} \equiv 1 \pmod{p}$, where $s_i = s_0^{e^i}$. To backdoor an RSA modulus N , we either ensure $f(e) \equiv 0 \pmod{\varphi(N)}$, or (like the “partially hidden cycle” eSP variant) $f(e) \equiv 0 \pmod{\varphi(p)}$ but not $\pmod{\varphi(q)}$.

Theorem 6. *SUS prime and RSA modulus generation is polynomial time in the length of the modulus.*

Proof. We apply Algorithm 4 except that we replace the desired relation $e^\ell - 1 \pmod{p-1}$ with a sparse polynomial $f(e) = \sum_i c_i e^i$. To generate an RSA

modulus, we can either generate two primes from different subsets of the factors in Step 3 (so that $f(e) \equiv 0 \pmod{\varphi(N)}$), or we can backdoor p and choose q normally (so that $f(e) \equiv 0 \pmod{\varphi(p)}$). The latter allows arbitrarily large N with fixed-size p , because the backdoor does not depend on the choice of q .

SUS State Recovery Attack. The state recovery algorithm uses multivariate Coppersmith. We write $s_0^{e^i} = s_i = b_i + 2^k r_i$, using the outputs b_i and unknown state MSBs r_i , with $0 \leq r_i < R = N/2^k$. For ease of exposition let us assume for now that $f(e) \equiv 0 \pmod{\varphi(N)}$; applying our backdoor polynomial f we obtain

$$\prod_{i \in S} (2^k r_i + b_i)^{c_i} \equiv 1 \pmod{N}$$

This is a low-degree multivariate polynomial modulo N whose roots are the unknown portions of each state. Recall all c_i are ± 1 , with roughly balanced sets S^+ of positive c_i and S^- of negative c_i . This gives

$$\prod_{i \in S^+} (2^k r_i + b_i) - \prod_{i \in S^-} (2^k r_i + b_i) \equiv 0 \pmod{N}$$

Our polynomial degree is $\max(|S^+|, |S^-|)$, which is independent of e . We can then recover the r_i using multivariate Coppersmith.

If instead we had $f(e) \equiv 0 \pmod{\varphi(p)}$ but not $\pmod{\varphi(q)}$, we would instead recover the $r_i \pmod{p}$. But as long as $p > R$ this is the same as recovering r_i over the integers.

Example. Suppose $f(e) = e^{200} + e^{20} - e^{180} - e^0 \equiv 0 \pmod{\varphi(N)}$; we have $|S^+| = |S^-| = 2$. The Coppersmith polynomial in unknowns $r_{200}, r_{20}, r_{180}, r_0$ is

$$f(\mathbf{s}) = (r_{200} + 2^{-k} b_{200})(r_{20} + 2^{-k} b_{20}) - (r_{180} + 2^{-k} b_{180})(r_0 + 2^{-k} b_0).$$

We apply Algorithm 3 with $t = 1$ and no extra shifts to generate a lattice with $\dim L = 7$ and $\det L = R^8 N^6$ (where R is our bound on the r_i). Applying Condition 1 (and omitting the approximation factor in dimension 7), we expect to succeed when $(R^8 N^6)^{1/7} < N$, or when $R < N^{1/8}$. Had we instead had $f(e) \equiv 0 \pmod{\varphi(p)}$, our success condition would instead be that $R < p^{1/8}$. In either case, this bound is independent of e .

As a demonstration, we generated a 1024-bit RSA modulus N satisfying $e^{200} - e^{180} + e^{20} - 1 \equiv 0 \pmod{\varphi(N)}$ for $e = 17$. We include it in the full version [24]. Parameter generation took 19s using Sage on a single core of an Intel E5-2699 processor. Using these parameters with $k = 896$ -bit outputs, our attack successfully recovered the state in 213 milliseconds from 200 PRG outputs. For these parameters, the fraction of bits output is below the $(1 - 1/e)$ fraction required by Herrmann and May [39]; that is, our attack requires less output to succeed than theirs. In fact, the fraction of bits output is smaller than $(1 - 2/e)$, the maximum fraction of output bits recommended in the ISO parameters for Micali-Schnorr — although when N is 1024 bits (at the 80-bit security level)

the ISO standard recommends only 864 bit outputs, to ensure at least 160 bits remain unknown. In practice, however, LLL reduction of this Coppersmith lattice yields shorter vectors than predicted, and our attack empirically succeeds for these example parameters with ISO-sized $k = 864$ outputs, and even with outputs as small as $k = 856$ bits.

Using a higher multiplicity (and thus a larger-dimension lattice) allows the attack to succeed with even smaller outputs:

Theorem 7. *A SUS-backdoored modulus N of length n with backdoor polynomial f of degree ℓ and $|S|$ nonzero coefficients allows an efficient state recovery attack after observing ℓ outputs of length $k > n(1 - 1/c_S)$ for a constant c_S that depends only on $|S|$, and not on the exponent e .*

When f has $|S| = 2$ terms, $c_S = 2$, when f has $|S| = 4$ terms, $c_S < 6.55$, and when $|S| = 6$ terms, $c_S < 16.96$.

The attack requires only $|S|$ outputs within this range at specified positions.

Proof. The recovery algorithm works as follows. Let $f(x) = \sum_i c_i x^i$. Let $S^+ = \{i \mid c_i > 0\}$ and $S^- = \{i \mid c_i < 0\}$. Apply multivariate Coppersmith’s method to solve for the unknown r_i in $\prod_{i \in S^+} (2^k r_i + b_i)^{c_i} - \prod_{i \in S^-} (2^k r_i + b_i)^{|c_i|} \equiv 0 \pmod p$.

For $|S| = 2$, we can construct a full-rank 3-dimensional lattice with multiplicity $t = 1$ to obtain the above bound. This case is degenerate: a polynomial with coefficients $+1, -1$ will generate output that cycles. We obtain the stated bound for $|S| = 4$ from a full-rank 1365-dimensional lattice with $t = 8$; for $|S| = 6$, a 1443-dimensional lattice with $t = 4$. For details see the full version [24].

We have chosen these values so that running LLL for these lattices is within feasible range today; one can get improved bounds for the c_S by choosing larger multiplicities and generating larger (but still polynomially sized) lattices.

Optimized lattice construction methods, like Herrmann and May’s technique of unravelled linearization [39], seem to not apply here. We give more analysis in the full version [24].

Undiscoverability vs. Practical Exploitation. We hypothesize that SUS-backdoored parameters could be undiscoverable, if the sparse backdoor polynomial f is properly chosen. However, making the backdoor harder to discover seems to make it harder to exploit.

The backdoor polynomial f in the SUS attack is a sparse polynomial with the property that $f(e) \equiv 0 \pmod{\varphi(N)}$ or $\pmod{\varphi(p)}$. More terms in f make it harder to guess, but it also significantly increases the required fraction of output bits or the dimension of the lattice reduced using Coppersmith’s method.

If f has too few terms, it becomes possible to guess f by brute force, and then verify a guess by checking (for some arbitrary a) whether $a^{f(e)} \equiv 1 \pmod N$ (if $f(e) = 0 \pmod{\varphi(N)}$) or if $\gcd(a^{f(e)} - 1, N)$ is nontrivial (if $f(e) = 0 \pmod{\varphi(p)}$).

As an example, suppose we want f to have eight nonzero terms. The multivariate polynomial to be solved using Coppersmith will have degree 4 ($|S^+| = |S^-| = 4$). If we assume the RSA PRG is reseeded every 50000 outputs (as the ISO standard recommends for MS DRBG), the degree of f must be

less than 50000, since any outputs after the first 50000 will not be related. The size of the search space for f would be roughly $\binom{50000}{7} \approx 2^{97}$. (It is 7 and not 8 because $f(e) \equiv 0 \Rightarrow e^j f(e) \equiv 0$ implies the existence of a backdoor polynomial with degree exactly 49999).

A meet-in-the-middle attack does better than brute force. If $f(e) \equiv 0 \pmod{\varphi(N)}$, split $f = f_0 + f_1$ into the first 3 and last 4 unknown terms. Since $a^{f_0(e)+f_1(e)} \equiv 1 \pmod N$, precompute possible $a^{f_0(e)}$ terms and check for collisions with $a^{-f_1(e)}$. This takes $\binom{50000}{4} \approx 2^{58}$ time and $\binom{50000}{3} \approx 2^{44}$ space. A similar meet-in-the-middle is possible when the backdoor relation is $\pmod{\varphi(p)}$ but not $\pmod{\varphi(N)}$, using fast multipoint evaluation; see the full version [24] for details. We conjecture that no faster attack is possible; we leave this question open.

To illustrate the tradeoffs, in addition to the earlier example, we generated two backdoored parameter sets: one requires as high as the 3500th output, took 4 core-hours to exploit, and we conjecture is 2^{25} -undiscoverable; the other requires only as high as the 150th output, took 3 core-minutes to exploit, but is conjectured only 2^9 -undiscoverable.¹⁰ We give these (and other) parameters and discuss the tradeoffs further in the full version [24].

Both the construction and the discoverability analysis extend to unbalanced or multi-prime RSA. Exploitation of the backdoor is most efficient if the output is as large as possible, so the attacker would want to work modulo $\varphi(N)$ or modulo $\varphi(p)$ for a large prime factor p .

Extending this Idea to Micali-Schnorr. Our attempts to extend this idea from RSA PRG to Micali-Schnorr have encountered some barriers.

First, the output of MS PRG does not follow the clean iterative structure of the RSA PRG. For RSA PRG, we can write the i th block of output b_i as a value that is close to a power of the initial state $s_i \equiv s_{i-1}^e \equiv s_0^{(e^i)} \pmod N$, or a single monomial like x^{e^i} in a polynomial equation. For MS PRG, writing the i th block of output in terms of the initial state by iteratively expanding the expression

$$s_i \equiv 2^{-k}(s_{i-1}^e - b_i) \pmod N,$$

yields a polynomial with exponentially many terms involving previous outputs.

The minimum degree of our backdoor polynomial $\sum_{i \in S} \pm e^i \equiv 0 \pmod{p-1}$ needs to be $\log_e \varphi(p)$ to embed information $\pmod p$, so our polynomial expression will have exponentially many terms in $\lg p$. Using larger coefficients in the polynomial to generate terms like ce^i increases the degree of the lattice polynomial.

Another way of viewing this obstacle is that the high-degree non-sparse relation between states is due to the simultaneous presence of addition, multiplication, and exponentiation modulo N (or p) in the state update function. If only exponentiation were involved, as is the case of RSA-PRG, we can simplify the expression as above. If only multiplication by a constant and addition were involved, all s_i are affine functions of s_0 . When all three operations are involved, however, the resulting expression is a polynomial with exponentially many terms.

¹⁰ Example code is available at https://github.com/ucsd-hacc/msdrbg_code.

One path forward would be to generate some algebraic structure that permits simplification or elimination of enough cross-terms that the polynomial no longer has exponentially many terms. (We give an example of such a structure in the next section.) Alternatively, we observe that these polynomials will have linear depth if evaluated as a circuit. Exploiting this idea would require new algorithmic ideas, since a lattice attack requires writing down the polynomial to be solved.

5.4 Attacks on MS PRG

Finite Field MS-PRG Is Insecure. In this section we define a variant of Micali-Schnorr over finite fields of small characteristic, and detail a straightforward state-recovery attack on this variant that involves no backdoors.

This attack does not imply anything about the existence of an attack (or a feasible backdoor) on standard MS DRBG, but it demonstrates that the pseudorandomness of modular exponentiation depends on the choice of field and illustrates algebraic structure that eliminates the exponential blow-up in terms that kept us from extending the ideas in the SUS backdoor to MS-PRG. This attack works for any choice of output length k , unlike the other attacks we detail.

Finite Field Micali-Schnorr. Our finite field variant of the Micali-Schnorr PRG is presented in Algorithm 6. Our eventual backdoor will rely on the characteristic of the field matching the exponent e , which we will take to be a small prime.

Let \mathbb{F}_{e^n} be the finite field of size e^n . We can represent elements of \mathbb{F}_{e^n} as polynomials in the quotient ring $\mathbb{F}_e[x]/N(x)$ (with N monic, irreducible, and $\deg(N) = n$) or as coefficient vectors in $(\mathbb{F}_e)^n$. Addition, multiplication, and exponentiation are defined in the standard ways.

Theorem 8 (Informal). *Finite-field Micali-Schnorr with state size n and output size k allows an efficient probabilistic state recovery attack when $\lceil (n-k)/k \rceil$ outputs are observed.*

Attacking FF-MS-PRG. Our attack relies on the linearity of the Frobenius endomorphism $x \mapsto x^e$ to limit the complexity introduced by exponentiation. This means the entire update step $s_i \leftarrow x^{-k}(s_{i-1}^e - b_i)$ is affine in terms of previous, and therefore also the initial, state. The bound $\deg(s_i) < n - k$ means several elements are 0 when interpreted as a coefficient vector, and this constraint allows the attacker to formulate a linear system of equations involving the known 0-elements of s_i , the unknown initial state s_0 , and the known outputs b_i .

Algorithm 6: Finite-Field Micali-Schnorr

Input : Parameters $e \in \mathbb{Z}$, $N \in \mathbb{F}_e[x]$, a number of iterations h
Output: hk output values in $[0, e - 1]$

- 1 Sample initial state $s_0 \in \mathbb{F}_e[x]/N$ of degree $< n - k$ using truly random coins.
- 2 **for** $i \leftarrow 1$ **to** h **do**
- 3 $z_i \leftarrow s_{i-1}^e \bmod N(x)$
- 4 Write $z_i = x^k s_i + b_i$ for $\deg(s_i) < n - k$ and $\deg b_i < k$.
- 5 **end**
- 6 Output $b_1 || b_2 || \dots || b_h$.

The attacker observes output until this linear system is overdetermined, and then solves it. A solution giving s_0 is guaranteed to exist. Although the solution is not always unique, in practice this method appears to recover a solution close to the initial state after $\lceil (n - k)/k \rceil$ outputs.

This attack is efficient. With $n = 1024$ and $k = 341$, recovering the FF-MS-PRG state from 9 outputs took 7 min implemented in Sage; the unoptimized construction of the linear system was the bottleneck. Further details of this attack are included in the full version [24].

The Bad- e (Be) Attack. In this section, we describe choices for the exponent e that lead to efficient state-recovery attacks for the Micali-Schnorr generator. The particular choices of e we make are unusual, but allowed by the ISO standard, and are efficient to exploit with the output sizes recommended by ISO.

As observed in Sect. 5.2, a straightforward application of multivariate Copersmith’s method for a state recovery attack against MS PRG is ruled out by the parameters specified by ISO. We can circumvent these restrictions by choosing a large e such that $e^{-1} \bmod \varphi(N)$ is small.

Flexible Choice of e . ISO specifies that “The implementation should allow the application to request any odd integer e in the range $1 < e < 2^{\lg(N)-1} - 2 \cdot 2^{\lg N/2}$.”

Our attack instantiates the public exponent e with a value other than the default exponent $e = 3$. Using larger e results in a larger output length k under the recommended parameters. Interestingly, while MS DRBG can be instantiated with almost any non-default e , there are more requirements on the public modulus : N may either be one of the default moduli or randomly generated.

Theorem 9 ($e = d^{-1}$ for Small d Is Insecure). *Instantiating Micali-Schnorr with RSA using exponent $e = d^{-1} \bmod \varphi(N)$ for d small allows an efficient state recovery attack from a single output when the state has length $r < n / \binom{d}{2} + 1$, in time polynomial in d and $\lg N$.*

Proof. One output b_1 yields a degree- d polynomial relating states s_0 and s_1 .

$$s_0^e = (2^k s_1 + b_1) \bmod N$$

$$s_0 - (2^k s_1 + b_1)^d = 0 \bmod N$$

A straightforward application of multivariate Coppersmith results in a lattice of dimension $d + 2$ and determinant $R^{\binom{d}{2}+1}N^{d+1}$ for R the bound on the size of the state. Applying Condition 1 and omitting the approximation factor if we expect d to be a small constant, we expect to succeed when $r(\binom{d}{2} + 1) < n$.

We can verify that this attack is allowed by the ISO parameters. When e is large, which is what we expect for $d^{-1} \bmod \varphi(N)$ for d small, the ISO parameters set $r = 2\gamma$ where γ is the security parameter. Thus we expect this attack to work when $\gamma(d^2 - d + 2) < n$. To be concrete, for the ISO security parameters (listed in Sect. 2), this inequality is satisfied for $d = 3$ for all parameter sizes, for $d = 5$ at $\gamma = 128$ and above, and $d = 7$ for $\gamma = 256$.

This exponent can be efficiently computed from knowledge of the factorization of N , and we expect it to be large since $\varphi(N) \mid de - 1$. This choice of e is arguably not a plausibly deniable “backdoor” since in practice e is almost always chosen to be small. In addition, it is efficiently discoverable for any small d via the Boneh and Durfee attack on small private RSA exponents [15]. However, in cryptographic protocols in which parameters are negotiated by machines making basic validity checks rather than actively looking for suspicious parameters, even such a discoverable backdoor could easily go undiscovered.

The attack can be generalized to $e = e_0e_1^{-1}$ for small e_0, e_1 , which enables efficient attacks on ISO security levels $\gamma = 112$ through 256, although the generalized attack is still detectable. We refer to the full version [24] for details.

6 Impact on Cryptographic Protocols

Deployed cryptographic systems typically use random numbers as input to a cryptographic protocol. The precise interaction between protocol, implementation and a subverted RNG impact the exploitability of a system. We now briefly consider how our attacks on MS and RSA PRG may affect common protocols.

Case Study: Using MS DRBG State Recovery to Subvert IPsec. IPsec [29] is an encryption protocol often used for VPNs. We focus on the key agreement protocol, typically IKE. During the period of standardization (approximately 2004–2007), the current version of the protocol was IKEv1 [38].

As noted in previous analyses [17, 18, 20] many IKE implementations use a single PRG to generate both unencrypted nonces, encryption padding and ephemeral secret keys for Diffie-Hellman key agreement. An attacker wishing to passively exploit the “Bad- e ” state recovery attack in Sect. 5.4 would observe protocol handshakes, use the nonces to recover the state, and then iterate the state forward to recover the secret Diffie-Hellman exponent, recover the shared secret, and derive the symmetric session keys to decrypt the session data.

For the simplest attack described in Theorem 9, state recovery is feasible for all security parameters with exponent $e = 3^{-1} \bmod \varphi(N)$ and requires observing at least $3n/4$ bits of output. For $n = 1024$ this is 96 bytes and for $n = 2048$ it is 192 bytes, both within the 256-byte upper limit on a variable-length nonce.

Extracting Generator Output from Public Nonces. The most likely source for public output is the random nonces in each key agreement: these range from 8–256 bytes in IKE. Thus in IKE a single nonce is conceptually sufficient to recover a single generator output block using a 1024-bit or 2048-bit modulus.

Extracting Generator Output from RSA Padding. Some configurations of the IKEv1 protocol employ RSA-PKCS#1v1.5 encryption to authenticate endpoints. In this configuration, one party encrypts a nonce to the other party's encryption key. Assuming an attacker can interact with the server once, it may therefore obtain raw PRG output in the padding of the RSA ciphertext.

With an RSA public key of length \bar{n} bytes and a nonce of length m bytes, each ciphertext contains $\bar{n} - 3 - m$ bytes of *non-zero* RSA padding bytes, in addition to the m -byte nonce.¹¹ Assuming a 32-byte nonce, this provides 93 bytes of padding (or 125 bytes for padding and nonce combined) for a 1024-bit RSA encryption key and 221 bytes (or 253 bytes for both) for a 2048-bit encryption key. The 93-byte padding is less than the 96 output bytes required for a 1024-bit key, but the remaining bytes could be recovered via brute force.

Case Study: TLS. SSL/TLS [6, 36] are the most common secure communications protocols used on the Internet. SSL and TLS each combine the use of long-term keys or secrets, as well as a key agreement protocol and symmetric encryption scheme for transmission of secure data into a single protocol. Common versions between 2004–2007 included SSL version 3 [36] and TLS 1.0–1.2 [6, 26, 57].

The random portion of an SSL/TLS nonce is 28 bytes long.¹² For SSL/TLS or IKE implementations with smaller nonces, an attacker would need to obtain several nonces over multiple key exchanges (≈ 4 at the 28 bytes length) in order to recover sufficient state to obtain one 108-byte MS DRBG output at the 1024-bit security level. Even this approach poses a challenge: for our basic attacks, the recovered output bytes must be *consecutive*. In a naive implementation of either protocol, the generation of nonces may be interspersed with other uses of the PRG: as a result, only fragments of each output block would be available. There are two potential engineering solutions that could mitigate this result:

1. During the standardization period, the NSA proposed and co-authored numerous IETF draft extensions to SSL/TLS [40–42, 58, 59] that cause servers and clients to output much longer nonces on request. At least one extension was ultimately deployed in the BSAFE commercial cryptography library [9]. The occasional use of such extensions by any client would provide eavesdroppers with an arbitrary amount of generator output that could be used to recover secret keys until the generator was reseeded.
2. Some commercial implementations of IKE *pre-generate* nonces in advance of a handshake, storing the results in a queue for later use [17]. Such implementations have been discovered in devices implementing the Dual EC DRBG

¹¹ The PKCS#1v1.5 standard requires that all padding bytes be non-zero, since the 0 byte is used as a delimiter. Recovering the raw byte stream would thus require some additional steps depending on how this string is generated.

¹² Each nonce is 28 bytes of random data concatenated with a 4-byte timestamp.

generator, a design choice that maximizes the practical impact of a subversion attack. A similar implementation decision could allow the exfiltration of multiple consecutive bytes of generator output over several handshakes.

There are also some algorithmic exploitation possibilities:

1. Multivariate Coppersmith methods can be used to solve for multiple nonconsecutive chunks of output; the exact bounds would depend on the details of the implementation.
2. Our SUS backdoor for RSA PRG exploits sequences of non-consecutive blocks of output, albeit selected to satisfy the linear backdoor recurrence embedded in the modulus. A more moderate improvement in the bounds might allow a recovery attack of this form.

7 Conclusion

In this paper, we studied the question of whether an adversary who controls the generation of the parameters used for the Micali-Schnorr PRG can break the security of the algorithm. To that end, we identified vulnerable parameters permitted by the ISO standard for Micali-Schnorr, and developed a novel backdoor algorithm for the closely-related RSA PRG that permits efficient state recovery attacks beyond previously known bounds. However, we encountered barriers in adapting our backdoor technique to MS PRG for realistic parameters, and thus the main question we set out to solve remains open.

A solution to this problem may involve the development of new ideas in the cryptanalysis of RSA. For example, the small characteristic finite field case has exploitable structure. Taking advantage of this structure leads to improvements in algorithms like the function field sieve for discrete logarithms over small-characteristic finite fields. An analogous improvement for the integers that allows simplifications of the recurrences might open doors (or be related to existing advances) in the study of factorization or RSA cryptanalysis algorithms.

Acknowledgments. We thank Emmanuel Thomé and Antonio Sanso as well as numerous attendees of CHES 2016 for enjoyable conversations about this problem. Bor de Kock contributed to an early version of this project. This work was supported by NSF under awards CNS-1653110, CNS-1801479, DMS-1913210, and CNS-2048563, and by DARPA under Contract No. HR001120C0084. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the United States Government or DARPA.

References

1. DRBG Recommendations from the x9.82 Editing Group. <https://github.com/matthewdgreen/nistfoia/blob/master/6.4.2014%20production/055%20%20DRBG%20Recomm%20from%20X9.82%20Editing%20Group.pdf>
2. DRBGs Based on Hard Problems. <https://github.com/matthewdgreen/nistfoia/blob/master/6.4.2014%20production/039%20-%20DRBGs%20Based%20on%20Hard%20Problems.pdf>
3. Excerpt from 2013 Intelligence Budget Request: SIGINT ENABLING. Media leak (2013). <https://archive.nytimes.com/www.nytimes.com/interactive/2013/09/05/us/documents-reveal-nsa-campaign-against-encryption.html>
4. SHA-3 Standard: permutation-based hash and extendable-output functions (2015-08-04 2015). <https://doi.org/10.6028/NIST.FIPS.202>
5. Alexi, W., Chor, B., Goldreich, O., Schnorr, C.P.: RSA/Rabin bits are $1/2 + 1/\text{poly}(\log N)$ secure. In: 25th FOCS, pp. 449–457. IEEE Computer Society Press (1984). <https://doi.org/10.1109/SFCS.1984.715947>
6. Allen, C., Dierks, T.: The TLS Protocol Version 1.0. RFC 2246 (1999). <https://doi.org/10.17487/RFC2246>, <https://www.rfc-editor.org/info/rfc2246>
7. Anderson, R.J.: Practical RSA Trapdoor. *Electron. Lett.* **29**, 995–995 (1993)
8. Bellare, M., Paterson, K.G., Rogaway, P.: Security of symmetric encryption against mass surveillance. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 1–19. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44371-2_1
9. Benjamin, D.: Additional TLS 1.3 Results from Chrome (December 2017). <https://mailarchive.ietf.org/arch/msg/tls/i9blmvG2BEPfls1OJkenHknRw9c/>
10. Berndt, S., Liskiewicz, M.: Algorithm substitution attacks from a steganographic perspective. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017, pp. 1649–1660. ACM Press (2017). <https://doi.org/10.1145/3133956.3133981>
11. Bernstein, D.J., Lange, T., Niederhagen, R.: Dual EC: A Standardized Back Door, pp. 256–281. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49301-4_17
12. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: On the indifferentiability of the sponge construction. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 181–197. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_11
13. Blum, L., Blum, M., Shub, M.: A simple unpredictable pseudo-random number generator. *SIAM J. Comput.* **15**(2), 364–383 (1986). <https://doi.org/10.1137/0215025>
14. Blum, M., Micali, S.: How to generate cryptographically strong sequences of pseudo random bits. In: 23rd FOCS, pp. 112–117. IEEE Computer Society Press (1982). <https://doi.org/10.1109/SFCS.1982.72>
15. Boneh, D., Durfee, G.: Cryptanalysis of RSA with private key d less than $N^{0.292}$. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 1–11. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48910-X_1
16. Cesati, M.: A new idea for RSA backdoors. arXiv preprint [arXiv:2201.13153](https://arxiv.org/abs/2201.13153) (2022).
17. Checkoway, S., et al.: A systematic analysis of the juniper dual EC incident. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016, pp. 468–479. ACM Press (2016). <https://doi.org/10.1145/2976749.2978395>

18. Checkoway, S., et al.: On the practical exploitability of dual EC in TLS implementations. In: Fu, K., Jung, J. (eds.) USENIX Security 2014, pp. 319–335. USENIX Association (2014)
19. Chow, S.S.M., Russell, A., Tang, Q., Yung, M., Zhao, Y., Zhou, H.S.: Let a non-barking watchdog bite: cryptographic signatures with an offline watchdog. In: Lin, D., Sako, K. (eds.) PKC 2019. LNCS, vol. 11442, pp. 221–251. Springer, Heidelberg (2019). https://doi.org/10.1007/978-3-030-17253-4_8
20. Cohnsey, S.N., Green, M.D., Heninger, N.: Practical state recovery attacks against legacy RNG implementations. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) ACM CCS 2018, pp. 265–280. ACM Press (2018). <https://doi.org/10.1145/3243734.3243756>
21. Coppersmith, D.: Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *J. Cryptol.* **10**(4), 233–260 (1997). <https://doi.org/10.1007/s001459900030>
22. Coppersmith, D.: Finding small solutions to small degree polynomials. In: Silverman, J.H. (ed.) *Cryptography and Lattices*, pp. 20–31. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44670-2_3
23. Crépeau, C., Slakmon, A.: Simple backdoors for RSA key generation. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 403–416. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36563-X_28
24. Davis, H., Green, M., Heninger, N., Ryan, K., Suhl, A.: On the possibility of a backdoor in the Micali-Schnorr generator. *Cryptology ePrint Archive*, Paper 2023/440 (2023). <https://eprint.iacr.org/2023/440>
25. Degabriele, J.P., Paterson, K.G., Schuldt, J.C.N., Woodage, J.: Backdoors in pseudorandom number generators: possibility and impossibility results. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, LNCS, vol. 9814, pp. 403–432. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53018-4_15
26. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346 (2006). <https://doi.org/10.17487/RFC4346>
27. Dodis, Y., Ganesh, C., Golovnev, A., Juels, A., Ristenpart, T.: A formal treatment of backdoored pseudorandom generators. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 101–126. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46800-5_5
28. Dodis, Y., Pointcheval, D., Ruhault, S., Vergnaud, D., Wichs, D.: Security analysis of pseudo-random number generators with input: /dev/random is not robust. In: Sadeghi, A.R., Gligor, V.D., Yung, M. (eds.) ACM CCS 2013, pp. 647–658. ACM Press (2013). <https://doi.org/10.1145/2508859.2516653>
29. Doraswamy, N., Glenn, K.R., Thayer, R.L.: IP Security Document Roadmap. RFC 2411 (1998). <https://doi.org/10.17487/RFC2411>, <https://www.rfc-editor.org/info/rfc2411>
30. Engelberts, L.: Analysis of the Micali-Schnorr PRNG with known factorisation of the modulus. Master’s thesis, University of Oxford (2020)
31. Fischlin, M., Mazaheri, S.: Self-guarding cryptographic protocols against algorithm substitution attacks. In: Chong, S., Delaune, S. (eds.) CSF 2018 Computer Security Foundations Symposium, pp. 76–90. IEEE Computer Society Press (2018). <https://doi.org/10.1109/CSF.2018.00013>
32. Fischlin, R., Schnorr, C.P.: Stronger security proofs for RSA and Rabin bits. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 267–279. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-69053-0_19
33. Fischlin, R., Schnorr, C.P.: Stronger security proofs for RSA and Rabin bits. *J. Cryptol.* **13**(2), 221–244 (2000). <https://doi.org/10.1007/s001459910008>

34. Fouque, P.A., Vergnaud, D., Zapalowicz, J.C.: Time/memory/data tradeoffs for variants of the RSA problem. In: Du, D.Z., Zhang, G. (eds.) *Computing and Combinatorics*, pp. 651–662. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38768-5_57
35. Fouque, P.A., Zapalowicz, J.C.: Statistical properties of short RSA distribution and their cryptographic applications. In: Cai, Z., Zelikovsky, A., Bourgeois, A. (eds.) *Computing and Combinatorics*, pp. 525–536. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08783-2_45
36. Freier, A.O., Karlton, P., Kocher, P.C.: The Secure Sockets Layer (SSL) Protocol Version 3.0. RFC 6101 (2011). <https://doi.org/10.17487/RFC6101>, <https://www.rfc-editor.org/info/rfc6101>
37. Green, M.: A few more notes on NSA random number generators (2013). <https://web.archive.org/web/20230109062504/https://blog.cryptographyengineering.com/2013/12/28/a-few-more-notes-on-nsa-random-number/>
38. Harkins, D., Carrel, D.: The Internet Key Exchange (IKE). IETF RFC 2409 (Proposed Standard) (1998)
39. Herrmann, M., May, A.: Attacking power generators using unravelled linearization: when do we output too much? In: Matsui, M. (ed.) *ASIACRYPT 2009*. LNCS, vol. 5912, pp. 487–504. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10366-7_29
40. Hoffman, P.E.: Additional Random Extension to TLS. Internet-Draft draft-hoffman-tls-additional-random-ext-01, Internet Engineering Task Force (2010). Work in Progress. <https://datatracker.ietf.org/doc/draft-hoffman-tls-additional-random-ext/01/>
41. Hoffman, P.E.: Additional Master Secret Inputs for TLS. RFC 6358 (2012). <https://doi.org/10.17487/RFC6358>, <https://www.rfc-editor.org/info/rfc6358>
42. Hoffman, P.E., Solinas, J.: Additional PRF Inputs for TLS. Internet-Draft draft-solinas-tls-additional-prf-input-01, Internet Engineering Task Force (2009). Work in Progress. <https://datatracker.ietf.org/doc/draft-solinas-tls-additional-prf-input/01/>
43. International Organization for Standardization: ISO/IEC 18031:2011 Information Technology—Security Techniques—Random Bit Generation (2011). <https://www.iso.org/standard/54945.html>
44. Jochemsz, E., May, A.: A strategy for finding roots of multivariate polynomials with new applications in attacking RSA variants. In: Lai, X., Chen, K. (eds.) *ASIACRYPT 2006*. LNCS, vol. 4284, pp. 267–282. Springer, Heidelberg (2006). https://doi.org/10.1007/11935230_18
45. Johnson, D.B.: X9.82 part 3: number theoretic DRBGs. Presented at the NIST RNG Workshop (2004). <https://csrc.nist.gov/CSRC/media/Events/Random-Number-Generation-Workshop-2004/documents/NumberTheoreticDRBG.pdf>
46. Joye, M.: RSA moduli with a predetermined portion: Techniques and applications. In: Chen, L., Mu, Y., Susilo, W. (eds.) *Information Security Practice and Experience*, pp. 116–130. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-79104-1_9
47. Jutla, C.S.: On finding small solutions of modular multivariate polynomial equations. In: Nyberg, K. (ed.) *EUROCRYPT 1998*. LNCS, vol. 1403, pp. 158–170. Springer, Heidelberg (1998). <https://doi.org/10.1007/BFb0054124>
48. Kobitz, N., Menezes, A.: Another look at “provable security”. II. (invited talk). In: Barua, R., Lange, T. (eds.) *INDOCRYPT 2006*. LNCS, vol. 4329, pp. 148–175. Springer, Heidelberg (2006). https://doi.org/10.1007/11941378_12

49. Lenstra, A.K., Lenstra, H.W., Jr., Lovász, L.: Factoring polynomials with rational coefficients. *Math. Ann.* **261**(4), 515–534 (1982)
50. Micali, S., Schnorr, C.P.: Efficient, perfect random number generators. In: Goldwasser, S. (ed.) *CRYPTO 1988*. LNCS, vol. 403, pp. 173–198. Springer, Heidelberg (1990). https://doi.org/10.1007/0-387-34799-2_14
51. Micali, S., Schnorr, C.P.: Efficient, perfect polynomial random number generators. *J. Cryptol.* **3**(3), 157–172 (1991). <https://doi.org/10.1007/BF00196909>
52. National Institute of Standards and Technology. Results of a recent FOIA for NIST documents related to the design of Dual EC DRBG (2015). <https://github.com/matthewdgreen/nistfoia/>
53. Nguyen, P.Q., Stehlé, D.: Lll on the average. In: Hess, F., Pauli, S., Pohst, M. (eds.) *Algorithmic Number Theory*, pp. 238–256. Springer, Heidelberg (2006). https://doi.org/10.1007/11792086_18
54. Patsakis, C.: Number theoretic SETUPS for RSA like factoring based algorithms. *J. Inf. Hiding Multim. Signal Process.* **3**(2), 191–204 (2012)
55. Perlroth, N., Larson, J., Shane, S.: N.S.A. able to foil basic safeguards of privacy on web. *New York Times* (2013). <https://www.nytimes.com/2013/09/06/us/nsa-foils-much-internet-encryption.html>
56. Peyrin, T., Wang, H.: The MALICIOUS framework: embedding backdoors into tweakable block ciphers. In: Micciancio, D., Ristenpart, T. (eds.) *CRYPTO 2020*. LNCS, vol. 12172, pp. 249–278. Springer, Heidelberg (2020). https://doi.org/10.1007/978-3-030-56877-1_9
57. Rescorla, E., Dierks, T.: The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (2008). <https://doi.org/10.17487/RFC5246>, <https://www.rfc-editor.org/info/rfc5246>
58. Rescorla, E., Salter, M.: Opaque PRF Inputs for TLS. Internet-Draft draft-rescorla-tls-opaque-prf-input-00, Internet Engineering Task Force (2006). Work in Progress. <https://datatracker.ietf.org/doc/draft-rescorla-tls-opaque-prf-input/00/>
59. Rescorla, E., Salter, M.: Extended Random Values for TLS. Internet-Draft draft-rescorla-tls-extended-random-02, Internet Engineering Task Force (2009). Work in Progress. <https://datatracker.ietf.org/doc/draft-rescorla-tls-extended-random/02/>
60. Russell, A., Tang, Q., Yung, M., Zhou, H.S.: Cliptography: clipping the power of kleptographic attacks. In: Cheon, J.H., Takagi, T. (eds.) *ASIACRYPT 2016*. LNCS, vol. 10032, pp. 34–64. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53890-6_2
61. Sanso, A.: How to try to predict the output of Micali-Schnorr generator (MS-DRBG) knowing the factorization (2017). <http://blog.intothesyymetry.com/2017/12/how-to-try-to-predict-output-of-micali.html>
62. Shumow, D., Ferguson, N.: On the possibility of a back door in the NIST SP800-90 Dual Ec Prng. Presented at the Crypto 2007 rump session (2007). <http://rump2007.cr.yt.to/15-shumow.pdf>
63. Steinfeld, R., Pieprzyk, J., Wang, H.: On the provable security of an efficient RSA-based pseudorandom generator. In: Lai, X., Chen, K. (eds.) *ASIACRYPT 2006*. LNCS, vol. 4284, pp. 194–209. Springer, Heidelberg (2006). https://doi.org/10.1007/11935230_13
64. Wüller, S., Kühnel, M., Meyer, U.: Information hiding in the RSA modulus. In: *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security*, pp. 159–167 (2016)

65. Yao, A.C.C.: Theory and applications of trapdoor functions (extended abstract). In: 23rd FOCS, pp. 80–91. IEEE Computer Society Press (1982). <https://doi.org/10.1109/SFCS.1982.45>
66. Young, A., Yung, M.: The dark side of “black-box” cryptography, or: should we trust capstone? In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 89–103. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-68697-5_8
67. Young, A., Yung, M.: Kleptography: using cryptography against cryptography. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 62–74. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-69053-0_6
68. Young, A., Yung, M.: A space efficient backdoor in RSA and its applications. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 128–143. Springer, Heidelberg (2006). https://doi.org/10.1007/11693383_9