5.2.2. Cube Attack

Cube attack [9] is a generic key-recovery attack that can be applied to any cryptosystem, provided that the attacker can obtain a bit of information that can be represented by a low-degree decomposition multivariate polynomial in Algebraic Normal Form of the secret and public variables of the target cryptosystem. According to the cube attack, our PRNG can be regarded as a system of multivariate polynomials $p(k_1, \dots, k_{45}, v_1, \dots, v_{20})$ with public IV variables $v_1, v_2 \dots, v_{20}$ and secret key variables $k_1, k_2, \dots, k_{45}$. The polynomial $p(k_1, \dots, k_{45}, v_1, \dots, v_{20}) = t_I . p_{S(I)} + q(k_1, \dots, k_{45}, v_1, \dots, v_{20})$ is called a *master* polynomial, where $t_I = v_{i_1} v_{i_2} \dots v_{i_k}$ is a monomial with $\{i_1, i_2, \dots, i_k\} \subseteq \{1, 2, \dots, 20\}$ and $p_{S(I)}$ is called a *superpoly* of t_I in p. The term t_I is called a *maxterm* if $deg(p_{S(I)}) = 1$. We implemented the cube attack against our PRNG in CUDA and exploited the power of a GPU (i.e, a Tesla C2070 from NVIDIA) for accelerating the computation significantly. We took the first output bit after the 36-round initialization phase in order to find the maxterms in the master polynomial and performed an exhaustive search over all possible cube dimensions ranging from 1 to 20. However, our experiment did not find any linear and quadratic superpoly equations for different cube dimensions.

5.2.3. Time-Memory-Data Tradeoff Attack

Time-memory-data tradeoff attack is a generic cryptanalytic attack which can be applied to any cipher. In a stream cipher, the complexity of a time-memory-data tradeoff attack depends on the length of the internal state, which is given by $O(2^{\frac{n}{2}})$, where n is the length of the internal state [4]. We note that a stream cipher with low sampling resistance is vulnerable to a more flexible time-memory-data tradeoff attack. In our PRNG, the WG transformation is the filtering function as well as the internal state update function and the number of terms in the algebraic normal form representation of the WG transformation is 15, among which only two terms are linear and the remaining terms are either quadratic or cubic. Only by fixing four input variables in the WG transformation, one can obtain a linear function in one variable. Thus, the sampling resistance of the proposed PRNG is high. Since the length of the internal state is 65-bit in our PRNG, the expected complexity of the time-memory-data tradeoffs attack is $O(2^l)$, where l equals 32.5.

5.2.4. Other Attacks

In the fast correlation attacks [21], the internal state of an LFSR based stream cipher can be recovered by first determining a system of linear equations according to a statistical model and then solving the system of linear equations. In our PRNG, the internal state is updated in a nonlinear way. Thus it is hard for an attacker to decide such a system of (non-) linear equations according to some statistical models.

For an LFSR based stream cipher, the DFT attacks [13] can be applied when the exact linear complexity of the output sequence and enough consecutive output bits are known. In our PRNG, the exact linear complexity and period of the output sequence are not known for an initial state. Therefore, the DFT attacks cannot be applied to our PRNG. Moreover, in the EPC C1 Gen2 standard protocol, it is hard for an attacker to obtain enough consecutive bits.

A chosen IV attack on the original version of WG cipher was presented in [25], where one can distinguish several bits of the output sequence by building a distinguisher based on differential cryptanalysis. In our PRNG, two nonlinear terms (i.e., an output from the WG transformation as well as a 5-bit tuple generated by the first building block) are added to the recurrence relation. Thus the differentials after 36 rounds of the initialization phase will contain most internal state bits. As a result, it would be hard for an attacker to distinguish output bits generated by the proposed PRNG.

5.3. Comparisons with Sponge-based PRNGs

A sponge-based PRNG is constructed using a sponge function [3], which is composed of two phases: an *absorbing* phase and a *squeezing* phase. While truly random seeds are fed into the internal state of the sponge function in the absorbing phase, the squeezing phase outputs pseudorandom numbers [3]. Any sponge-based hash function such as U-QUARK [1], DM-PRESENT [6], PHOTON [14], and SPONGENT [5] can be used to construct a sponge-based PRNG. Since a sponge-based PRNG requires a multiple seeding mechanism, one needs additional random sources to provide multiple truly random seeds to the PRNG while generating pseudorandom numbers. Note that the multiple seeding mechanism provides the forward secrecy for the sponge-based PRNGs, which can also be achieved by our PRNG provided that additional random sources are available[1]. Moreover, our PRNG

[1] Assume that the length of a seed K is a multiple of the length of the secret key $K = k_1 || k_2 || \dots || k_r, r \geq 1$. If a multiple seeding mechanism is applied to our PRNG, the seed is updated by repeating the following step r times: k_i $(i = 1, 2, \dots, r)$ is XORed with the values in the key bit positions of the current internal state, followed by an 18-round of the initialization phase.