Designing a Side Channel Resistant Random Number Generator

Suresh N. Chari[1], Vincenzo V. Diluoffo[2], Paul A. Karger[1], Elaine R. Palmer[1], Tal Rabin[1], Josyula R. Rao[1], Pankaj Rohotgi[1,⋆], Helmut Scherzer[3,⋆⋆], Michael Steiner[1], and David C. Toll[1]

[1] IBM Corporation, Thomas J. Watson Research Center
P.O. Box 704, Yorktown Heights, NY 10598, USA
{schari|diluoffo|erpalmer|talr|jrrao|msteiner|toll|pkarger}@us.ibm.com
[2] IBM Corporation, Systems and Technology Group
150 Kettletown Rd., Southbury, CT 06488, USA
[3] IBM Deutschland GmbH, Secure Systems and Smart Cards
Schönaicher Str. 220, D-71032 Böblingen, Germany

Abstract. This paper describes the design of the random number generator (RNG) in the Caernarvon high assurance smart card operating system. Since it is used in the generation of cryptographic keys and other sensitive materials, the RNG has a number of stringent security requirements that the random bits must be of *good* quality i.e. the bits must not be predictable or biased. To this end, a number of standards such as the German AIS 31 mandate that true random bits be *continuously* tested before use in sensitive applications such as key generation. A key issue in implementing this standard is that such testing before use in key generation greatly increases the attack surface for *side-channel* attacks. For example, template attacks which can extract information about the random bits from even a *single* run provided we use the same bits at *many* different points in the computation. Because of these potential risks, the Caernarvon operating system uses pseudo random number generators which are initially seeded by *externally* generated high quality random bits, and then perturbed by bits from the true random number generator. We describe a PRNG design which yields high quality random bits while also ensuring that it is not susceptible to *side-channel* attacks and provide an informal argument about its effectiveness.

1 Introduction

This paper describes the design of a side-channel resistant random number generator for the Caernarvon [31] high-assurance smart card operating system project. The Caernarvon OS is intended to test if it is possible to build very high levels

[⋆] Now with Cryptography Research, 575 Market Street, 11th Floor, San Francisco, CA 94105, USA, **pankaj.rohatgi@cryptography.com**

[⋆⋆] Now with Giesecke & Devrient GmbH, Postfach 80 07 29, D-81607, München, Germany, **helmut.scherzer@gi-de.com** of assurance into smart card operating systems - it is to be evaluated at Common Criteria EAL7 under the German evaluation scheme. The choices we have evaluated and the features of our design would also be applicable to a number of other high security environments. In particular, practical considerations such as resistance to side-channel attacks and minimizing the wear on persistent memory which have guided our design will be relevant in many different applications.

High assurance systems such as Caernarvon require high quality random bits to support a multitude of uses. In Caernarvon the use cases include:

- **Key Generation.** Random sources are typically used to generate keys for symmetric ciphers such as DES and AES. Frequently, they are also used to generate keying material in algorithms such as Diffie-Hellman key exchange. Occasionally they are used to generate asymmetric cryptographic keys such as RSA keys.
- **Random nonce and other parameter generation.** Smart card systems use RNGs to generate nonces and randomness used in cryptographic protocols.
- **Blinding.** Commonly used countermeasures to defeat timing attacks use random numbers to blind data and keys.
- **Masking.** Increasingly, random numbers are used to mask operands to protect against side-channel attacks such as SPA/DPA[20], EM analysis[1], etc.

The security and effectiveness of the implementation of cryptographic algorithms and functions crucially rely on the random numbers used in the above operations: they should be of "good" quality and should be kept secret. Random numbers that are predictable or biased may open the crypto algorithms (or their keys) to attack. Equally, it is of little use to generate keys for cryptographic algorithms if random numbers used to generate these keys leak to the outside world.

Smart cards and other systems typically contain a *true* random number generator (TRNG), *i.e.* a physical source of entropy. These are implemented by circuits which generate random numbers and whose physical properties are chosen to produce high quality random bits which could pass all the standard tests for random sequences. However, the use of true RNGs is not without potential problems. For example, it has been observed that hardware random number generators may "age", *i.e.* the quality of the random numbers degrades over time. Also, small devices, such as smart cards, can sometimes be vulnerable to differential fault attacks [5, 4] against the the hardware RNG itself, such that it generates predictable numbers or even always generates the same number.

Given the potential for such problems, a number of standards place stringent requirements on the use and testing of random numbers from a hardware RNG. For Common Criteria evaluations under the German scheme, these requirements are specified in AIS 31 [15] which requires that the RNG be tested on system start up, (*i.e.* on every activation of the smart card), and also that the used random numbers are continuously tested. The tests specified in AIS 31 are those defined in FIPS 140-2 [27]. These testing requirements, of course, have significant impact on performance and usability. Performing FIPS 140-2 tests on card activation takes a significant amount of time, which would be noticeable by any user of the card. Further, performing such tests would be very difficult while still meeting the stringent timing constraints imposed on smart card startup by the ISO 7816-3 standard [16]. Continuous RNG tests, executed as the smart card performs crypto operations, also exacerbate performance problems.

Such testing of random bits can also adversely affect the security of the implementation: Side-channel attacks such as SPA/DPA [20], EMF [1], template attacks [7], and other "TEMPEST" attacks [30] are capable of extracting useful information from even a single operation of a device. While it is difficult to extract significant information from *just* reading the TRNG output, the leakage is amplified if the *same* sensitive value is used at many different places in the computation. Template attacks work by building a "database" of signatures of the side channel for each possible value of some sensitive byte or bits; the likelihood of building a good distinguishing signature increases direcly in proportion to the number of places the same value is used or manipulated. If we run a series of tests on the output of the TRNG before other uses, then we manipulate the output bits of the TRNG without modification at many different places. This increases the attack surface of side-channel attacks!

We even proposed testing the TRNG output with one set of bits, confirming that the TRNG is operating correctly, and then immediately generating a new set of random bits from the TRNG to use without testing. However, our evaluation laboratory concluded that this proposal would not comply with the AIS 31 requirements. Thus, we concluded that it was unrealistic to use a TRNG directly for a high-assurance system that must comply with AIS 31 and also resist side-channel attacks.

Instead, we adopted a compelling alternative: pseudo random number generators (PRNGs) which generate a sequence of random bits starting from an initial (random) seed. There are many secure methods to construct PRNGs: for example the algorithms specified in FIPS 186-2 [11]. Our design generates good quality random bits compliant to relevant standards while simultaneously ensuring that the PRNG cannot be attacked using side-channels. Our PRNG is based on a random seed generated *off-line* using a fully tested source of true random bits. This seed is stored in persistent memory and feeds a component PRNG whose output is the seed for a second component PRNG whose output is used for card operations. The first PRNG is run for each activation of the smart card and its state is fed back to memory. Thus, we minimize the number of updates to persistent memory which is crucial due to the limited number of write cycles for EEPROM/Flash. We use the TRNG to perturb the values of the seeds to protect against any possible compromise of the stored seeds. Since this perturbation of the seeds does not constitute direct use of TRNG output for cryptographic purposes, AIS 31 does not apply, but AIS 20 [14] does. We argue that this construction results in a good quality stream of random bits and justify the security of our construction and its resistance to side channel attacks. The construction of our PRNG is similar to that of Petit, et. al. [22] who also design a similar PRNG with the goal of resistance to side channel attacks. We note here that our construction [8] precedes theirs.

This paper is organized as follows: Section 2 describes Caernarvon and provides relevant background on hardware RNGs and standards which govern their testing. Sections 3 and 3.1 describe the construction of our PRNG, and Section 4 argues that the quality of random numbers produced by the PRNG is good. We discuss the security of our construction and specifically the resistance to side-channel attacks in Section 5, and Section 6 describes related work.

2 Background

2.1 Caernarvon Operating System

The Caernarvon operating system was designed to be evaluated under the Common Criteria [10] at EAL7, the highest defined level of assurance, under the German evaluation scheme. It demonstrates that high assurance for smart cards is technically feasible and commercially viable. Historically, smart card processors have not supported hardware protection features necessary to separate the OS from the applications, and one application from another [18]. The assurance in the Caernarvon OS is based on exploiting the first smart card processors to offer such hardware protection features.

The Caernarvon OS implements a formally specified, mandatory security policy [23] providing multi–level security, suitable for both government agencies and commercial users. The mandatory security policy requires effective authentication of its users independent of applications, for which the Caernarvon OS contains a privacy-preserving, two–way authentication protocol [24] integrated with the policy. The Caernarvon OS also includes a strong cryptographic library that has been separately certified under the Common Criteria at EAL5+ for use with other systems. While the initial platform for the operating system was smart cards, the design could also be used in other embedded devices, such as USB tokens, PDAs, cell phones, etc.

2.2 Hardware Random Number Generators

Smart cards and other similar systems typically feature true random number generators (TRNG) built from physical sources of noise. There has been considerable amount of work on harnessing such physical sources to produce good quality unbiased random output (see, for example, [2, 21, 13]).

Since they are built from physical sources, the output of TRNGs may include biases. Thus, before use in sensitive applications, the output needs to be tested to ensure quality. A good description of an potential evaluation methodology for TRNGs is described by Schindler, *et. al.*[25]. This and other standards offer testing guidelines but do not endorse or exclude any TRNG design principles.

It should be noted that, as of July 2009, as stated on page 1 of FIPS 140-2, Annex C [6], “There are no FIPS Approved non-deterministic random number generators.” FIPS-140-2 refers to hardware or TRNGs as *non-deterministic.*

2.3 Testing requirement standards

The following are some of the standards for testing of RNGs before using the output in cryptographic and other sensitive applications.

The AIS 31 RNG Testing Requirements Killman and Schindler [19] proposed tests for RNGs to ensure that evaluated systems do not suffer from failure models of hardware RNGs. This was later implemented in an Application Note and Interpretation of the Scheme(AIS 31) [15] and recommends that the output of hardware RNGs be testing carefully prior to use with start-up and continuous tests. While such testing will certainly result in better quality, we feel that the recommendations do not take into account the potential for side channel attacks on the implementation of such testing. Section 2.4 discusses some of these attacks and how they can be applied to RNG testing phases. It is our belief that the AIS 31 mandated testing can significantly increase the attack surface for side channel attacks. This issue was first discussed by Karger [17].

Because AIS 31 does not consider this increased potential for side-channel attacks due to testing, Common Criteria Guidance Documents for several evaluated smart card chips[4] require that cryptographic use of the random numbers generated by a true RNG be subject to the tests of FIPS 140-2 [27, Section 4.9.1], thereby requiring the increased attack surface.

We note that the latest protection profile for smart card chips [26] does discuss the risks of the inherent leakage. However, an update to AIS 31 is still needed, because the potential risks are not limited only to smart cards.

FIPS Tests FIPS-140, the definitive US standard for cryptographic devices, did include a number of test on the output of RNGs which have been dropped since 2002. The draft of the upcoming FIPS 140-3 [28] includes a number of tests for *pseudo* random number generators (PRNGs). FIPS 140-3 mandates the following tests for PRNGs (Random bit generator(RBG) in their terminology):

- Deterministic components of a Random Bit Generator (RBG) shall be subject to the Cryptographic Algorithm Test in Section 4.9.1 (of FIPS 140-3).
- Data output from the RBG shall pass the Continuous RBG Test as specified in Section 4.9.2 (of FIPS 140-3).

The cryptographic algorithm test requires that the algorithms used in the PRNG be tested before they are used. The continuous test is that each generated random number be saved so that the next one generated compared with the previous one. The standard further specifies that if an entropy source (a hardware RNG) is used, then the minimum entropy test must be performed on each output of the source. We note that the same performance and security issues are applicable to these tests. For instance, it will be difficult to perform the cryptographic

[4] Guidance documents are defined in the Common Criteria [9] assurance component AGD_USR.1. Citations can not be provided due to non-disclosure requirements.

algorithm test at start-up while still meeting the maximum latency requirements of ISO 7816-3 [16]. As argued earlier, any testing performed on random bits can increase the attack surface for side channel attacks.

2.4 Side Channel Attacks

High assurance systems must be built to resist attacks which exploit information such as power consumption [20], EM emanations [1], template attacks [7], TEMPEST attacks [30] and other such by-products of the implementation of sensitive operations that are capable of extracting useful information during the computation. Here we only highlight how these attacks affect our design choices for the RNG.

Simple Power Attacks (SPA) and its EM equivalent (SEMA) target leakages that occur in a single execution of the device, *e.g.* through conditional execution of code depending on a sensitive value. While these are very powerful they are easy to protect against, and most implementations guard against such obvious leakage. These attacks also target other leakages that can occur in a single step, *e.g.* in some hardware reading a byte from EEPROM leaks the Hamming weight of the byte that is read.

Differential Power and EM attacks (DPA/DEMA) exploit statistical biases that occur in side channels due to manipulation of sensitive values. The attacks work by first amplifying these biases by running the device with multiple different inputs. For these attacks to be successful, the same sensitive values must be manipulated in all different runs of the device.

Template attacks extract useful information from a single sample of the side channel from the device. These work by building signatures of the side channel for each possible value of some sensitive byte (or a few bits) using a test device. Given a single run of the device under test they attempt to identify using statistical techniques the most likely value of the sensitive byte. Key to the success of these attacks is building the right set of signatures. The likelihood of building better signatures increases with the number of places in the computation that the *same* byte is manipulated. For instance, if random bits sampled from an RNG are subject to a number of tests where the same bits are being manipulated, then the likelihood of building good signatures increases.

2.5 Constraints of Persistent Memory

Smart cards use EEPROM and/or Flash as persistent memory since they have no access to off-chip memory. The PRNG in the Caernarvon system is designed to use persistent memory across runs of the card.

Write operations to EEPROM or Flash memory are slow, usually in the 1 to 6 millisecond range each, depending on the technology generation. Further, the write block size for EEPROM is limited, for example, to 128 bytes. Thus writing any significant amount of data to a file is likely to take multiple write operations, plus additional writes to update the control block information. Furthermore, EEPROM and Flash memories have a limited number of write cycles before they start to fail, for example between 100,000 and 500,000 for EEPROM, and only 10,000 for Flash.

Our PRNG described below will store PRNG state in persistent storage but do this once per run of the card. The Caernarvon operating system includes extensive techniques to mitigate these problems for more general applications.

3 RNG Design Overview

The design criteria for the PRNG are the following:

- **Quality** The random bits produced in each run of the smart card should be unpredictable. Further, the random bits in any run should be unpredictable, even knowing the bits in any other run.
- **Security** The random number generator should be resistant to side-channel attacks such as SPA/DPA [20], EM Analysis [1] and Template Attacks [7].
- **Effectiveness** The RNG should make effective use of persistent storage minimizing updates to such storage.

Given these requirements the design follows quite naturally: First, quality necessitates a seed sampled from a high entropy source, but obtaining this from an on-chip source would require testing the bits. As noted, this can increase the surface for side channel attacks, thus lowering the effective entropy of the on-chip source. Thus our PRNG is seeded by random bits which are stored in EEPROM which can, of course, be generated offline securely and comprehensively tested. The FIPS 140-2 standard [27] does not impose requirements or tests on this external source of entropy. However, the draft for FIPS 140-3 requires that the claimed minimum entropy of the source be provided to the cryptographic module which is then required to verify that the claimed value is sufficient for intended applications. This is currently not part of our design, but we note that the device can't directly test the source entropy. The best one could do is check for plausible error conditions, such as strings of all constants, etc.

The PRNG in the Caernarvon operating system is shown in Fig. 1. It consists of two component PRNGs cascaded: the first is called the Lifetime PRNG or the LPRNG and the second is called the Activation PRNG or APRNG. LPRNG is seeded by random bits which are stored in persistent memory. Each invocation of either component can be *optionally* seeded with additional random bits from an on-chip source of randomness. We stress that the strength of the random bits generated by the composite PRNG *primarily* depends on the quality of the external seed LSEED. For instance, if this seed is revealed due to a compromise of the off-chip process then this can compromise the PRNG. Adding on-chip randomness can ensure that the output of the PRNG is not a deterministic function of LSEED. While the PRNG doesn't depend on this on-chip source for its strength, a pedantic reading of the standards may require us to test this input. Template and other side channel attacks can significantly reduce the effective entropy of the on-chip source. Adding this to the seed obtained from the external source can not *decrease* the strength of the PRNG. Further, while complying with standards, one could argue that since the claimed strength is only dependent on LSEED, we may not need to test this additional optional input. AIS 31 [19] recognizes this type of use as functionality class P1 which does not require such extensive testing.

Each invocation changes the internal state of the PRNG which is used in the next invocation. For the LPRNG, this is stored back in EEPROM as the seed for the next run as shown. The output of the first invocation of the LRPNG is the seed for the APRNG whose output is used whenever random bits are required in this run of the smart card. In the Caernarvon OS, we choose as PRNG implementations schemes recommended by FIPS 186-2 [11]. The analysis of our random number generator does not depend on the choice of the PRNG block chosen from amongst those recommended in Annex C of the FIPS 140-2 standard [6]. We have chosen the algorithm for generating random values from the Appendix 3.2 of the Digital Signature Algorithm standard [11] with the method described in Appendix 3.4 using the DES algorithm to implement the $G()$ function. We note here that the recommendations given by NIST [3] are more recent and should be the choice for an updated design.

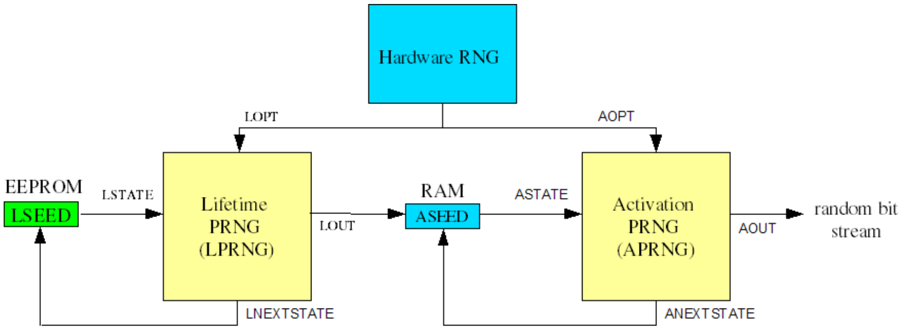

Fig. 1. Functional Outline of Caernarvon PRNG

3.1 Detailed description of the PRNG

This section briefly describes the construction of the two PRNGs and documents the choices made from relevant standards.

LPRNG is seeded by a 160 bit value LSEED from EEPROM. The value t chosen as `67452301 EFCDAB89 98BADCFE 10325476 C3D2E1F0` is used in the "compress" function at each invocation. The pseudo-code for the update function of LPRNG is

```
Inputs:
   - LSTATE: content of LSEED, stored in persistent storage; initially
     generated from an external source of entropy and installed in the
     chip at initialization.
   - LOPT: optional input from Hardware RNG (corresponding to input
     labelled ‘‘optional user-input’’ in FIPS 186-2).

Update:
a. LSTATE = (LSTATE + LOPT)
b. LOUT = G(t, LSTATE)
c. LNEXTSTATE = (LSTATE + 1 + LOUT)

Output:
   - LOUT: used as new seed ASEED by APRNG.
   - LSTATENEXT: new state used to replace previous LSEED
```

This sequence is executed exactly once during an activation of the card. The initial value of `LSTATE` is obtained from persistent storage `LSEED`. To potentially add more entropy, we chose to add randomness from an on-chip source via `LOPT`. Testing the bits `LOPT` will, of course, reduce the effective entropy of the source due to side channel attacks. As we have noted before, the strength of our PRNG rests solely on the strength of `LSEED` and hence we argue that this may be enough to address the requirements of the standards even without testing `LOPT`.

The output of LPRNG, i.e. `LOUT = G(t, LSTATE)`, is used to seed the APRNG. The updated state `LNEXTSTATE` is written back to persistent storage. Our implementation ensures that until this state is successfully updated in persistent storage, APRNG will not be activated. This ensures that if random bits are used anywhere in this activation of the card, then the value of `LSEED` will indeed be different in future activations. This prevents the attack where the card is disabled before the write back to EEPROM is completed resulting in the same sequence of random bits across different activations.

The "compress" function $G()$ will be based on the DES algorithm, specified in Section 3.4 of FIPS 186-2, chosen because of on-chip DES hardware. The properties of our RNG would be the same for compress functions built from other algorithms. For new designs, a better choice would be the HMAC-SHA1 based construction of deterministic RBGs given by [3].

The APRNG uses `LOUT`, the output from the LRPNG, as its seed. It is constructed similar to LPRNG with optional additional input from on-chip random sources. Its pseudo-code is:

```
Inputs:
   - ASTATE: content of ASEED, on activation the output of the LPRNG.
   - AOPT: optional input chosen from source of randomness This is done
     at most once. Further invocations will NOT have any additional input.

Update:
a. ASTATE = (ASTATE + AOPT)
b. AOUT = G(t, ASTATE)
c. ANEXTSTATE = (ASTATE + 1 + AOUT)

Output:
   - AOUT: randomness returned to caller of APRNG.
```

- `ASTATENEXT`: new state used to replace previous `ASEED`.

This generates 160 bits of randomness for each invocation and updates the internal state of APRNG.

The Caernarvon random number generation process described above realizes the requirements we had earlier listed. First, cryptographic keys and other sensitive values are generated from the output of a PRNG. Thus, we do not need to test the quality before use which may result in exposure through side channel attacks. However, the quality of the random bits is still high since the seed used by the PRNG is sampled from an high entropy source off-line. The strength of our PRNG rests (almost) exclusively on the strength of the off-line process for generating LSEED. We note that even if testing of optional input from on-chip random sources results in reduced levels of entropy due to side channel attacks, it is still sufficient when combined with LSEED. The analysis will show that the quality is maintained across different activations of the card. Further we note that the update to the seed in LPRNG is done only once per activation of the card, and thus we make effective use of persistent storage.

4 Cryptographic Analysis

This section justifies the cryptographic strength of the PRNG construction. First, we argue generically that the chaining construction of the PRNG is secure assuming we start with a secure individual PRNG construction. This and the assumption that the PRNG schemes recommended by the FIPS 186-2 standard are secure yield a proof of security of our chaining construction. We also argue that our PRNG is a class $K4$ DRNG according to the AIS 20 [14] standard.

In following lemma we state that our construction as shown in Figure 1 is a special instance (with $n = 2$) of a general class of secure PRNGs composed by chaining a primitive PRNG n times:

Lemma 1. *Let $r_{\text{chain}(n)}$ be a PRNG formed by chaining n primitive PRNGs r_{prim}. If there is a distinguisher which can distinguish the output of $r_{\text{chain}(n)}$ from a uniformly distributed random string of equal length then there is also a distinguisher distinguishing the output of r_{prim} from a random string.*

Proof Without loss of generality, assume that r_{prim} has a seed length of l, each inner PRNG r^i_{prim} reseeds its child m times before getting reseeded itself and we output externally lm^n bytes, i.e., each r^i_{prim} expands a seed to length lm. When reseeding a particular PRNG r^i_{prim} we talk of a new *instance* of that PRNG.

For this case, you can visualize $r_{\text{chain}(n)}$ as a m-ary tree of depth n where the ith level corresponds to the instances of the ith PRNG r^i_{prim} and the jth child of a node corresponds to the jth chunk of l bytes returned by the corresponding instance of the PRNG. The concatenation of the leaf nodes is the output produced by $r_{\text{chain}(n)}$.

The lemma follows from an inductive hybrid argument. The hybrids at depth n are $r_{\text{chain}(n)}$, $r_{\text{chain}(n)}$ with each of the m sub-trees of depth $n-1$ recomputed from a fresh random seed (instead of the seed derived from the parent), and for progressive hybrids we replace the sub-trees in increasing order of index with a sub-tree where all nodes are freshly sampled random elements.

The distribution induced by the extreme hybrids correspond to the distribution of $r_{\text{chain}(n)}$ and of a random distribution, respectively. It is also easy to see that neighboring hybrids either differ by (a) a single value which is either a random lm string or a single expansion of r_{prim} from a random and independent seed or (b) a sub-tree which correspond to an (independent) $n-1$ PRNG $r_{\text{chain}(n-1)}$ or a random tree. Hence, we can reduce a distinguisher of any pair of hybrids in a distinguisher of either r_{prim} or $r_{\text{chain}(n-1)}$ from random data. The latter in turn can be reduced recursively into hybrids until we arrive in hybrids differing all only in a r_{prim} distinguishing problem. As there are only a polynomial number of hybrids, any non-negligible advantage in distinguishing $r_{\text{chain}(n)}$ from a random string can be converted into a non-negligible distinguisher of r_{prim} from a random string. $\square$

Thus if we are given a secure PRNG primitive and the seed to the PRNG chain is chosen uniformly at random then our construction is cryptographically secure. Thus under the assumption that the FIPS 186-2 PRNG is secure and assuming proper secret and random seed generation at smart card personalization time, our overall realization is cryptographically secure as well. (Note that to our knowledge there is no published security proof (in a strong cryptographic sense) for the FIPS-186-2 PRNG. However, sound design principles, a number of easy to made security arguments and empirical evidence give us a reasonable assurance of its security.)

Informally, note that in the construction of the PRNG, on each invocation we add the output of the PRNG back to its internal state. Thus the PRNG is *forward secure i.e.* given the internal state after k invocations we can not infer the random outputs from prior invocations of the PRNG. Thus, the PRNG design fulfills the requirements of a $K4$ DRNG according to the AIS-20 standard. More precisely, the fulfillment of the individual criteria is as follows:

- $K1$ **DRNG:** This is a simple requirement which requires that we identify an integer value c such that every sequence of c outputs of the PRNG is distinct. By our assumption, the output of the core PRNG primitive *i.e.* the FIPS 186-2 generator is indistinguishable from random so it trivially satisfies this requirement ignoring the eventual cycling of the 160 bit output.
- $K2$ **DRNG:** Specifically, we are asked to characterize the statistical properties of the RNG such as the monobit test, poker test and tests on runs. We note that the FIPS 186-2 generators and hence our construction will satisfy all these criteria.
- $K3$ **DRNG:** This level requires us to assert that the entropy of the PRNG is at least 80. We note here that our PRNG operates on 160 bit seeds which are chosen off-line from a high entropy source which is carefully tested. Thus our PRNG achieves this level.
- $K4$ **DRNG:** For this level, we are required to argue that the PRNG is *forward-secure* as describe above. As argued, our PRNG meets this criterion.

FIPS 186-2 does not impose any requirements on the user input, i.e. it does not need to be random or secret to guarantee pseudo-randomness of the output stream or the security of the algorithm. Thus, replacing the user input with the output of the HW/RNG does not affect the security of the implementation, even if the HW/RNG malfunctions. On the other hand, if the HW/RNG is functioning properly then the output of LPRNG is truly random, and the output of APRNG is pseudo-random. This shows that there is no requirement to test the statistical properties of the bits of the HW/RNG, as they do not affect the security of the cryptographic aspects of the system. Yet, when it is functioning properly we gain entropy.

5 Attacks and Defenses

A key criterion for our design was resistance to side-channel attacks against the functioning of the PRNG. We have considered the following attacks.

- Simple Power Analysis/Simple EM Analysis (SPA/SEMA).
- Differential Power Analysis/Differential EM Analysis (DPA/DEMA)
- Template attacks using Power or EM or both.

Our design and implementation has been guided by techniques which are effective in the mitigation of these attacks. In particular, the following techniques which minimize and practically eliminate the threat posed by these attacks.

Technique 1 The strength of our PRNG relies on the entropy of the external seed supplied to the card. Using off-chip sources greatly reduces the attack surface for template attacks as discussed below. Our PRNG is *only* claimed to be as strong as this external seed. The addition of the input from an on-chip source does not affect this claim.

Technique 2 The execution sequence of all PRNG code is independent of its internal state.

Technique 3 As defined, both the component PRNGs can be optionally provided additional seeding material from on-chip sources. Due to the testing requirements this can only be counted on to provide a marginal additional source of entropy. However, even this amount can add to the strength of the PRNG. While it is certainly not feasible to base the entire PRNG on sampling from the on-chip source due to the low effective entropy, adding this optional input can be beneficial.

Technique 4 The implementation utilizes the hardware RNG to implement random masking/share based computation of the PRNG specification. While side channel attacks will only result in lowered entropy we argue below that this is sufficient to protect against statistical side-channel attacks.

As discussed below these techniques provide adequate countermeasures against side-channel attacks. The Caernarvon Persistent Storage Management code provides a CRC check on bits being stored. This will be disabled due to potential leakage of information.

5.1 Simple Power Analysis (SPA)

Simple power/EM analysis attacks target leakage that occurs in a single activation of the card such as through conditional execution depending on sensitive state OR high leakage on any given step of the computation. Our implementation is careful to ensure that the PRNG code execution sequence is independent of state (Technique 2). Thus SPA/SEMA attacks are limited to leakage at individual steps such as reading of bytes from EEPROM in the case of LPRNG. Masking steps in the computation, even when sampled from the hardware RNG, can also reduce the information that is revealed during the computation.

5.2 Differential Power Analysis (DPA)

Classical DPA is not a problem with LPRNG/APRNG since an attacker cannot invoke the update function of any PRNG many times with the same secret state. The state gets modified at each invocation of the update function and this constantly evolving secret state is a good defense against DPA style attacks; the attacker gets only one sample to attack any secret state. Note that in our implementation we actually add random masks with the bits sampled from an on-chip source. As we have discussed before, we can only count on this being a low entropy source. However, even with this source of bits, masking can further prevent DPA. We reiterate that the main defense is that DPA is not easy to mount since the keys change at every invocation.

5.3 Template attacks

Template attacks can be used to classify the single signal received during the operation of the LPRNG/APRNG on a given unknown state. The efficacy of this attack relies on having a large enough ”signature” or “template” of computation manipulating the *same* sensitive value. The attacker can build offline a series of templates for this signature corresponding to different values of this sensitive input and use them to identify the specific value on a given activation of the card.

The main defense against template attacks is the use of an off-chip source to generate LSEED. This can be directly used in the computation of the PRNG and thus we can be certain that building templates this will be difficult. This is because of the properties of the PRNG there is rapid diffusion and LSEED will not be manipulated unmodified at too many points. We do allow for the optional input sampled from the on-chip RNG. With full conformance to the standards, there is a good chance that template attacks can substantially reduce the effective entropy. Note, however, that the PRNG is still secure since LSEED is not tested. The masking of the computation using Technique 4 above will also make it difficult to build effective templates even though it is sampled from a source with low effective entropy.

Resistance to side channel attacks can be best argued with a description of the implementation, since there are many implementation details relevant to the argument. Further, to convincingly argue against side channel attacks we have to formalize the precise attack models along the lines of Petit *et. al.*[22]. We have not built such a model for our system.

6 Additional Related Work

Several recent designs for hardware random number generators have appeared, including Dole's [12] that describes networked computers generating and sharing entropy in proportion to the need for random numbers, Walsh and Beisterfeldt's [33] that generates high-quality random numbers by sampling the output of a Voltage Controlled Oscillator (VCO) at a frequency much lower than the frequency of the oscillator output, and Sprunk's [29] which uses a TRNG to drive a PRNG. Tsoi, Leung and Leong [32] show compact FPGA implementations of both a TRNG and a PRNG, but they do not connect them together. Furthermore, none of these designs address the possibility of side channel attacks or the issues of EEPROM memory wear.

7 Conclusions

We have seen that generation of cryptographically strong random numbers is actually quite difficult. While many of the standards concerning the testing of random numbers generated by hardware or true random number generators (TRNG) quite properly worry about hardware failures, they do not adequately cover the possibility of side channel attacks during RNG testing. We have shown a novel combination of a TRNG with a PRNG that alleviates both the testing concerns and the side-channel concerns that also limits the possibility of EEPROM or Flash memories being worn out from re-writing seed values too frequently. While our examples have all focused on cryptographic algorithms, such as DES, SHA-1, DSA, and RSA, the principles equally well apply to newer algorithms, such as elliptic curves, AES, and SHA-256, etc.

We must recommend that both the NIST FIPS 140 standard and the German AIS 31 guideline be updated to reflect these kinds of issues, so that future developers can more easily construct random number-based systems that are truly secure against both hardware failures and side-channel attacks.

References

[1] Agrawal, D., Archambeault, B., Rao, J.R., Rohatgi, P.: The EM side-channel(s). In: Proc. CHES. pp. 29–45. LNCS, Vol. 2523, Springer (13-15 Aug 2002)

[2] Bagini, V., Bucci, M.: A design of reliable true random number generator for cryptographic applications. In: Proc. CHES. pp. 204–218. LNCS, Vol. 1717, Springer (12-13 Aug 1999)

[3] Barker, E., Kelsey, J.: Recommendation for random number generation using deterministic random bit generators (revised). NIST SP800-90, National Institute of Standards and Technology, Gaithersburg, MD (Mar 2007), `http://csrc.nist.gov/publications/nistpubs/800-90/SP800-90revised_March2007.pdf`

[4] Biham, E., Shamir, A.: Differential fault analysis of secret key cryptosystems. In: Proc. Crypto. pp. 513–525. LNCS, Vol. 1294, Springer (17–21 Aug 1997)

[5] Boneh, D., DeMillo, R.A., Lipton, R.J.: On the importance of checking cryptographic protocols for faults. In: Proc. EUROCRYPT. pp. 37–51. LNCS, Vol. 1233, Springer (11–15 May 1997)

[6] Campbell, J., Easter, R.J.: Annex c: Approved random number generators for FIPS PUB 140-2, security requirements for cryptographic modules. FIPS PUB 140-2, Annex C, National Institute of Standards and Technology, Gaithersburg, MD (Draft of 31 July 2009), `http://csrc.nist.gov/publications/fips/fips140-2/fips1402annexc.pdf`

[7] Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: Proc. CHES. pp. 13–28. LNCS, Vol. 2523, Springer (13–15 Aug 2002)

[8] Chari, S.N., Diluoffo, V.V., Karger, P.A., Palmer, E.R., Rabin, T., Rao, J.R., Rohatgi, P., Scherzer, H., Steiner, M., Toll, D.C.: Method, apparatus and system for resistence to side channel attacks on random number generators. United States Patent No. 7496616 (Filed 12 Nov 2004, Issued 24 Feb 2009)

[9] Common Criteria for Information Technology Security Evaluation, Part 3: Security assurance requirements. Version 2.3 CCMB2005-08-003 (August 2005), `http://www.commoncriteriaportal.org/public/files/ccpart3v2.3.pdf`

[10] Common Criteria for Information Technology Security Evaluation, Parts 1, 2, and 3. Version 2.3 CCMB2005-08-001, CCMB2005-08-002, and CCMB2005-08-003 (August 2005), `http://www.commoncriteriaportal.org/thecc.html`

[11] Digital signature standard. FIPS PUB 186-2, with Change Notice 1, 5 October 2001, National Institute of Standards and Technology, Gaithersburg, MD (Jan 2000), `http://csrc.nist.gov/publications/fips/archive/fips186-2/fips186-2.pdf`

[12] Dole, B.: Distributed state random number generator and method for utilizing same. United States Patent No. US6628786B1 (30 September 2003)

[13] Epstein, M., Hars, L., Krasinski, R., Rosner, M., Zheng, H.: Design and implementation of a true random number generator based on digital circuit artifacts. In: Proc. CHES. pp. 152–165. LNCS, Vol. 2779, Springer (8–10 September 2003)

[14] Functionality classes and evaluation methodology for deterministic random number generators. AIS 20, Version 1, Bundesamt für Sicherheit in der Informationstechnik (BSI), Bonn, Germany (2 December 1999), `http://www.bsi.bund.de/zertifiz/zert/interpr/ais20e.pdf`

[15] Functionality classes and evaluation methodology for physical random number generators. AIS 31, Version 1, Bundesamt für Sicherheit in der Informationstechnik (BSI), Bonn, Germany (25 Sept 2001), `http://www.bsi.bund.de/zertifiz/zert/interpr/ais31e.pdf`

[16] ISO 7816-3, Identification cards - Integrated circuit(s) with contacts - Part 3: Electronic signals and transmission protocols, Second edition. ISO Standard 7816-3, International Standards Organization (Dec 1997)

[17] Karger, P.A.: The importance of high-assurance security in pervasive computing. In: Security in Pervasive Computing: First International Conference. p. 9. LNCS, Vol. 2802, Springer, Boppard, Germany (12–14 Mar 2003), `http://web.archive.org/web/20040524183841/http://www.dfki.de/spc2003/karger.pdf`

[18] Karger, P.A., Toll, D.C., McIntosh, S.K.: Processor requirements for a high security smart card operating system. In: Proc. 8th e-Smart Conference. Eurosmart, Sophia Antipolis, France (19–21 Sep 2007), available as IBM Research Division Report RC 24219 (W0703-091), `http://domino.watson.ibm.com/library/CyberDig.nsf/Home`

[19] Killman, W., Schindler, W.: A proposal for: Functionality classes and evaluation methodology for true (physical) random number generators. Tech. rep., T-Systems debis Systemhaus Information Security Services and Bundesamt fr Sicherheit in der Informationstechnik (BSI), Bonn, Germany (25 Sep 2001), http://www.bsi.bund.de/zertifiz/zert/interpr/trngk31e.pdf

[20] Kocher, P., Jaffe, J., Jun, B.: Differential Power Analysis: Leaking Secrets. In: Proc. Crypto. pp. 143–161. LNCS, Vol. 1666, Springer (15–19 August 1999)

[21] Maher, D.P., Rance, R.J.: Random number generators founded on signal and information theory. In: Proc. CHES. pp. 219–230. LNCS, Vol. 1717, Springer (12–13 Aug 1999)

[22] Petit, C., Standaert, F.X., Pereira, O., Malkin, T., Yung, M.: A block cipher based pseudo random number generator secure against side-channel key recovery. In: ASIACCS'08. pp. 56–65. Tokyo, Japan (18–20 March 2008)

[23] Schellhorn, G., Reif, W., Schairer, A., Karger, P., Austel, V., Toll, D.: Verification of a formal security model for multiapplicative smart cards. In: Proc. 6th (ESORICS). pp. 17–36. LNCS, Vol. 1895, Springer (2000)

[24] Scherzer, H., Canetti, R., Karger, P.A., Krawczyk, H., Rabin, T., Toll, D.C.: Authenticating Mandatory Access Controls and Preserving Privacy for a High-Assurance Smart Card. In: Proc. 8th (ESORICS). pp. 181–200. LNCS, Vol. 2808, Springer (13–15 Oct 2003)

[25] Schindler, W., Killmann, W.: Evaluation criteria for true (physical) random number generators used in cryptographic applications. In: Proc. CHES. pp. 431–449 (13–15 Aug 2002)

[26] Security IC platform protection profile. Tech. Rep. BSI-PP-0035, developed by Atmel, Infineon Technologies AG, NXP Semiconductors, Renesas Technology Europe, and STMicroelectronics, registered and certified by Bundesamt für Sicherheit in der Informationstechnik (BSI), Bonn, Germany (15 June 2007), http://www.commoncriteriaportal.org/files/ppfiles/pp0035b.pdf

[27] Security requirements for cryptographic modules. FIPS PUB 140-2, Change Notice 2, National Institute of Standards and Technology, Gaithersburg, MD (3 December 2002), http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf

[28] Draft - security requirements for cryptographic modules. FIPS PUB 140-3, National Institute of Standards and Technology, Gaithersburg, MD (6 Apr 2007), http://csrc.nist.gov/publications/fips/fips140-3/fips1403Draft.pdf

[29] Sprunk, E.J.: Robust random number generator. United States Patent No. US6253223B1 (26 Jun 2001)

[30] Tempest fundamentals (u). Declassified in 2000 under Freedom of Information Act NACSIM 5000, National Security Agency, Ft. George G. Meade, MD (1 Feb 1982), http://cryptome.org/nacsim-5000.zip

[31] Toll, D.C., Karger, P.A., Palmer, E.R., McIntosh, S.K., Weber, S.: The caernarvon secure embedded operating system. Operating Systems Review 42(1), 32–39 (2008)

[32] Tsoi, K.H., Leung, K.H., Leong, P.H.W.: Compact FPGA-based true and pseudo random number generators. In: 11th Annual IEEE Symp. on Field-Programmable Custom Computing Machines. Napa, CA (9–11 Apr 2003)

[33] Walsh, J.J., Biesterfeldt, R.P.: Method and apparatus for generating random numbers. United States Patent No. US6480072B1 (12 Nov 2002)