TRNG (순수 난수 발생기)의 테스트 기법 연구

Test Methods of a TRNG (True Random Number Generator)

문상국

목원대학교 정보전자영상공학부

Sangook Moon

Mokwon University, School of Information-Electronics-Image Engineering

E-mail : smoon@mokwon.ac.kr

요 약

TRNG (True Random Number Generator)를 테스트 하는 방법은 PRNG (Pseudo Random Number Generator)나 산술연산기를 비롯한 결정적 (deterministic) 소자에 대한 테스트와는 많이 틀려서, 새로운 개념과 방법론이 제시되어야 한다. 하드웨어적으로 결정적인 소자들은 패턴을 사용한 테스트 (ATPG; automatic test pattern generation)에 의해 커버가 될 수 있지만, 순수 난수는 발생 결과의 아날로그적인 특성에 의하여 자동 패턴 생성 방식에 의해 소자를 테스트하기가 불가능하다. 본 논문에서는 하드웨어와 소프트웨어를 결합한 테스트 방식으로 테스트 패턴에 연속적인 패턴의 변화를 주면서 통계적으로 관찰하는 방식인 Diehard test라는 테스트 방식을 연구, 분석하고, 순수 난수의 테스트 시 고려해야 할 주안점을 제안한다.

ABSTRACT

Since the different characteristics from the PRNG (Pseudo Random Number Generator) or various deterministic devices such as arithmetic processing units, new concepts and test methods should be suggested in order to test TRNG (Ture Random Number Generator). Deterministic devices can be covered by ATPG (Automatic Test Pattern Generation), which uses patterns generated by cyclic shift registers due to its hardware oriented characteristics, pure random numbers are not possibly tested by automatic test pattern generation due to its analog-oriented characteristics. In this paper, we studied and analyzed a hardware/software combined test method named Diehard test, in which we apply continuous pattern variation to check the statistics. We also point out the considerations when making random number tests.

키워드

TRNG, PRNG, test method, ATPG

Ⅰ. 서 론

정보보호기술은 암호학적인 알고리즘에 의해 복잡한 연산을 거쳐 이루어진다. 데이터를 암호화하는 기술은 암호에 사용되는 많은 수학연산을 처리하기 위해 높은 컴퓨팅 파워를 요구한다. 이러한 이유로 인해 지난수년 동안 웹사이트에서의 신용카드 구매와 같은 경우를 제외하고는 대부분의 비즈니스에서 사용되지 않았으나, 최근에는 자신들이 가지고 있는 서버에 간단하게 add-on 보드 혹은 appliance 고속암호연산 프로세서 제품들을 장착하여 암호화의 수행을 빠르게 연산 가능

하도록 할 수 있게 되었다.

TRNG (True Random Number Generator)는 전기적 장치의 일종으로 컴퓨터 장치와 연동되어 순수한 난수를 발생시킨다. 이는 컴퓨터 프로그램에서 난수와 비슷하나 한계가 존재하는 결정적(deterministic) 주기를 가지는 의사 난수를 발생시키는 PRNG (Pseudo Random Number Generator)와 상대되는 개념을 가지고 있다. TRNG의 사용 용도로는 암호화에 사용되는 키의 생성, 복권추첨, 혹은 통계적인 시뮬레이션의 응용 방면으로 사용되며, 저항 소자, 혹은 반도체 다이오드, 혹은 방사능물질로부터의 전기적인 잡음을 난수 발생의 근거로 삼는다.

이러한 TRNG를 테스트하는 방법은 PRNG나 산술연산기를 비롯한 결정적 소자에 대한 테스트와는 많이 틀려서, 새로운 개념과 방법론이 제시되어야 한다. 이를테면, TRNG가 어떤 테스트를 통과했다고 하더라도 이 TRNG가 좋은 TRNG라는 보장이 없으며, 그 TRNG가 어떤 테스트를 실패했다 하더라도 그 TRNG가 반드시 나쁜 TRNG라는 보장이 없다는 것이다. 본 논문에서는 하드웨어와 소프트웨어를 결합한 테스트 방식으로 테스트 패턴에 연속적인 패턴의 변화를 주면서 통계적으로 관찰하는 방식인 Diehard test라는 테스트 방식을 연구, 분석하고, 순수 난수의 테스트 시 고려해야 할 주안점을 제안한다.

II. TRNG 테스트 기술

George Marsaglia는 PRNG의 테스트를 위하여 Diehard tests 라는 일련의 통계적인 테스트 방법을 제안하였는데, 이를 TRNG에 적용하기 위해서는 테스트 패턴에의 상당한 변화를 주면서 연속적인 패턴의 변화를 주면서 통계적으로 관찰하는 방법을 사용하여야 한다. 이상적인 하드웨어와 소프트웨어가 결합된 TRNG의 블록 다이어그램은 다음 그림 1과 같다. 소스로는 순수한 난수를 얻기 위해 아날로그 잡음을 사용한다. 순수한 아날로그 잡음은 앰프와 디지타이저에서 증폭되고 디지털 신호로 바뀌고 교정된 후 PRNG에서 제어되는 출력값과 XOR 연산 되어 출력된다.

아날로그 잡음 소스는 저항이나 반도체를 사용하고, 디지털 변환기(digitizer) 는 잡음 소스에서 발생된 잡음을 디지털 신호로 변환시켜 0 또는 1의 연속된 문자열을 생성하고, 이는 난수 발생기의 표준화되지 않은 출력이 된다. 생성된 난수들이 난수성을 가지기 위해서는 그 난수들에 대한 통계적인 평균이 1과 0이 적절히 배열된 통계를 가지는, 즉 0.5의 평균값을 보여야 하는데, 이를 위해서 교정기 (corrector)가 필요하다.

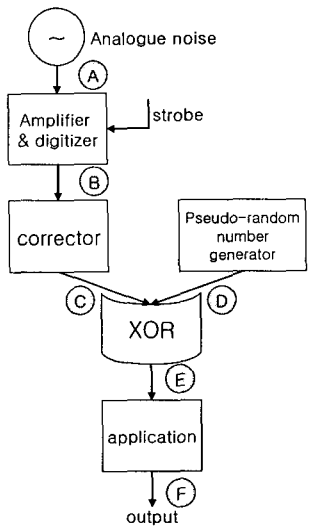

그림 1. TRNG 블록 다이어그램
Figure 1. TRNG block diagram

난수 문자열의 바이어스 (bias)를 교정하기 위한 교정기는 두 가지 종류가 있는데, 하나는 XOR 논리연산을 이용한 교정기와 Von Neumann 교정기가 있다. XOR 교정기는 연속된 비트의 쌍을 입력으로 받아 XOR 논리출력을 결과 값으로 취하는 것으로, 본래의 난수열이 독립적인 성질을 가지고 있다면 간단히 바이어스를 최소화 시킬 수 있는 특성을 가지고 있다. Von Neumann 교정기 역시 마찬가지로 연속된 비트의 쌍을 입력으로 받는데, 여기서 두 비트가 다르면 첫 번째 비트를 결과 값으로 취하고, 두 비트가 같으면 결과 값을 버림으로써 통계적으로 바이어스 되지 않은 난수열을 취할 수 있다. 표 1에서 교정되지 않은 난수열이 주어졌을 때 XOR 교정기와 Von Neumann 교정기의 출력 값을 예를 들어 보인다.

교정기를 거쳐나온 바이어스를 최소화한 비트열은 다시 PRNG에 의해 생성된 의사난수 비트열과 XOR 논리연산이 요구되는데, 이는 아날로그 난수발생기의 일시적인 전기적 glitch 혹은 회로 오동작과 같은 순간적인 오류 (bad spots)가 발생했을 때를 대비한 안전장치라고 할 수 있다. 그림 1에서 A에서 F까지의 레이블은 각 단계에서의 테스트 포인트를 지정해 놓은 것이다. D, E, F 포인트에서의 테스트는 난수성 측면에서는 특별한 의미가 없으며, 최소한 교정되지 않은 출력인 B 포인트에서의 테스트가 가장 큰 의미를 가진다. 난수 테스트 시 고려해야 할 사항들은 다음 표 2와 같다.

표 1. XOR 교정기와 Von Neumann 교정기에 의한 바이어스 처리 예
Table 1. An example of comparison of bias handling between XOR corrector and Von Neumann Corrector

Raw
10110010100100011010010111001000 1010010111
XOR
1 0 0 1 1 1 0 1 1 1 1 1 0 0 1 0 1 1 1 1 0
Von Neumann
1 1 1 0 0 1 1 0 0 1 1 1 0 0

표 2. 난수 특성 확인 시 고려사항들
Table 2. Things to consider when checking randomness of numbers

• Bias
• Drift
• Short term auto-correlation
• Other short term dependencies
• Discrete frequencies
• 1/*f* noise
• Other non-whiteness
• Bad spots
• Back door

III. Fake 연산

TRNG 처리부의 연산 흐름은 연산 시간을 줄이기 위하여 키의 비트를 확인하여 0일 경우 해당하는 연산을 수행하지 않고 다음 연산으로 넘어가는 방법을 사용함으로써 연산 시간을 줄이는 장점을 취하였다. 그러나 이런 경우를 사용하게 된다면 단순히 걸리는 시간만을 연속 체크함으로써 난수 키에 존재하게 되는 1의 개수 또는 0의 개수를 알아내기 쉽다는 단점이 나타날 수 있다. 이러한 단점을 막기 위해 실제 암호 연산을 하지 않는 경우 시간 체크 공격에 대비할 수 있는 fake 연산을 수행하는 방법을 고려하게 되었다. 본 연구에서 제안한 방법은 상당히 간략한 방법이라, 하드웨어적인 부담이 거의 없으면서 위와 같은 공격방법에 대응할 수 있는 방안을 제시한다.

Fake 연산이란, 말 그대로 외부적으로 보기에는 곱셈 동작을 취하는 듯 보이지만, 실제로는 해당 연산이 적용되지 않는 것을 말한다. 이를 정확히 구현하기 위해서는 해당 동작 수행 시에 수행 cycle과 전력의 소모를 파악한 후, 이와 대등한 동작을 수행하는 회로를 추가하는 방법이 가장 안전하다. 하지만, 이 경우에는 추가적인 하드웨어에 들어가는 비용이 기존 하드웨어의 50%를 넘어갈 가능성이 높기 때문에 좀 더 유효한 방법이 필요하다.

이러한 방안의 가장 간결한 해결책으로서, 기존의 하드웨어를 그대로 사용할 것을 제안하였다. 이렇게 될 경우 추가적인 하드웨어의 부담이 줄고 기존 연산과 동일한 전력소모를 가져올 수 있다. 그러나 실제 연산 결과를 레지스터 저장소에 저장하게 될 경우, 원래 수행해야 할 부분에 덮어쓰기가 되어 정상적인 동작을 못할 수도 있다. 따라서 fake 명령어가 들어올 경우에는 기록만 방지하도록 하고, 명령어 자체에서 받아들이는 operand의 특정 값의 상태에 따라 fake인지 아닌지를 구분하여 수행하도록 하면 된다.

IV. 결론 및 토의

TRNG를 테스트하는 방법은 PRNG나 산술연산기를 비롯한 결정적 소자에 대한 테스트와는 많이 틀려서, 새로운 개념과 방법론이 제시되어야 한다. 이를테면, TRNG가 어떤 테스트를 통과했다고 하더라도 이 TRNG가 좋은 TRNG라는 보장이 없으며, 그 TRNG가 어떤 테스트를 실패했다 하더라도 그 TRNG가 반드시 나쁜 TRNG라는 보장이 없다는 것이다. 본 논문에서는 하드웨어와 소프트웨어를 결합한 테스트 방식으로 테스트 패턴에 연속적인 패턴의 변화를 주면서 통계적으로 관찰하는 방식인 Diehard test라는 테스트 방식을 연구, 분석하고, 순수 난수의 테스트 시 고려해야 할 주안점을 제안하였다. 이러한 난수 발생 구조는 적용 범위에 상관 없이 다양한 암호화 어플리케이션에서 사용할 수 있다.

IV. 참고문헌

[1] William Stallings, Cryptography and network security principles and practie, 3rd Edition, (c) 003 by Pearson Education, Inc

[2] Trusted Open Group, "TOG Architecture Overview" , http://www.trustedcomputinggroup.org

[3] Trusted Open Group, "TOG TPM Specification v1.2 part 1" , http://www.trustedcomputinggroup.org

[4] Robert Davies, "Hardware Random Number Generations" , http://robernz.net/hwrng.thm