Introduction

Probability and statistics play very important roles in hardware security and trust (in fact, they do in MANY other fields as well)

Our focus is on their application to Trojan Detection and Physical Unclonable Functions

We will discuss only a few (of many) statistical techniques for these problems, in particular

- NIST statistics for evaluating the *randomness* of bit streams generated by PUFs
- Hamming distance statistics for evaluating PUF *uniqueness* and *stability*
- Regression models and outlier analysis for hardware Trojans detection

Randomness (this material derived from NIST "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications")

A **random bit sequence** can be interpreted as the result of a sequence of 'flips' of an unbiased (fair) coin

Randomness

Randomness

With sides labeled ’0’ and ’1’, each flip has probability of exactly 1/2 of producing a ’0’ or ’1’

Also, the ’flip’ experiments are **independent** of one another

The fair coin toss experiment is an example of a *perfect* random bit generator because the ’0’s and ’1’s are randomly and uniformly distributed

The result of the next trial is IMPOSSIBLE to predict!

Random Number Generators (RNGs)

An RNG uses a *non-deterministic* source (the **entropy source**, e.g., noise in an electrical circuit), plus a processing function (the **entropy distillation** process) to produce randomness

The *distillation* process is used to overcome any weaknesses in the entropy source that results in production of non-random numbers

Randomness

There are an *infinite number* of possible statistical tests that can be applied to a sequence to determine whether ’patterns’ exist

Therefore, no *finite* set of tests is deemed complete

Statistical tests are formulated to test a specific **null hypothesis** (H_0)

Here the null hypothesis-under-test is that the sequence being tested is **random**

The antonym to H_0 is the alternative hypothesis (H_a), that the sequence is NOT random

Each test has an underlying ***reference distribution*** which is used to develop a **critical value**, e.g., a value out on the tail of the distribution, say at 99%

The **test statistic** computed for the sequence is compared against the critical value, and if larger, the sequence is deemed NOT random (H_0 is rejected)

The premise is that the tested sequence, if random, has a very low probability, e.g., 0.01%, of exceeding the critical value

Randomness

The probability of a **Type I** error (data is actual random but test statistic exceeds critical value) is often called the **level of significance**, α

- Common values used in crypto are 0.01

Analogously, the probability of a **Type II** error (data is not random but passes the test) is denoted by β

- Beta (unlike alpha) is NOT a fixed value because there are an infinite number of ways a sequence can be non-random

The NIST tests attempt to minimize the probability of a Type II error

Note that the probabilities α and β are related to each other and to the size n of the tested sequence

- The third parameter is dependent on the other two

- Usually sample size n and an α are choosen, and a critical value is computed that minimizes the probability of a Type II error

Randomness

A **test statistic**, e.g. *S* is computed from the data, and is compared to the critical value *t* to determine whether H_0 is accepted

S is also used to compute a ***P-value***, a measure of the *strength* of the evidence **against** H_0

Technically, the *P-value* is the probability that a perfect RNG would have produced a sequence **less random** than the sequence-under-test

If the *P-value* is 1, then the sequence appears to have *perfect* randomness, if 0, then completely non-random, i.e., **larger** *P-values* support randomness

A significance level, α, is chosen and indicates the probability of a Type I error

- If the *P-value* $>= \alpha$, then H_0 is accepted, otherwise it is rejected

If α is 0.01, then one would expect 1 truely random sequence in 100 to be rejected

- A *P-value* < 0.01 indicates that the sequence is non-random with a **confidence** of 99%

Randomness

Two major assumptions:

- Uniformity: At **any** point in the generation of a random bit sequence, the number of ’0’s and ’1’s is equally likely and is 1/2, i.e., expected number of ’1’s is *n/2*
- Scalability: Any test applicable to a sequence is also applicable to a **subsequence** extracted at random, i.e. all subsequences are also random

Entropy

A measure of the *disorder* or *randomness* in a closed system

The entropy of uncertainty of a random variable X with probabilities p_i, ..., p_n is

$$H(X) = -\sum_{i=1}^{n} p_i \log p_i$$

$$H(X) = \frac{1}{\ln(2)}\log_2\left(\frac{1}{p}\right)$$ When $p_i = 1/n$ (equal probabilities)

$$H_\infty(X) = \min_{i=1}^{n}(-\log_2 p_i) = -\log_2(\max_i(p_i))$$ (**min-entropy**)

A distribution has a min-entropy of at least *b* bits if no possible state has prob. > 2^{-b}

Probability Distributions

Entropy

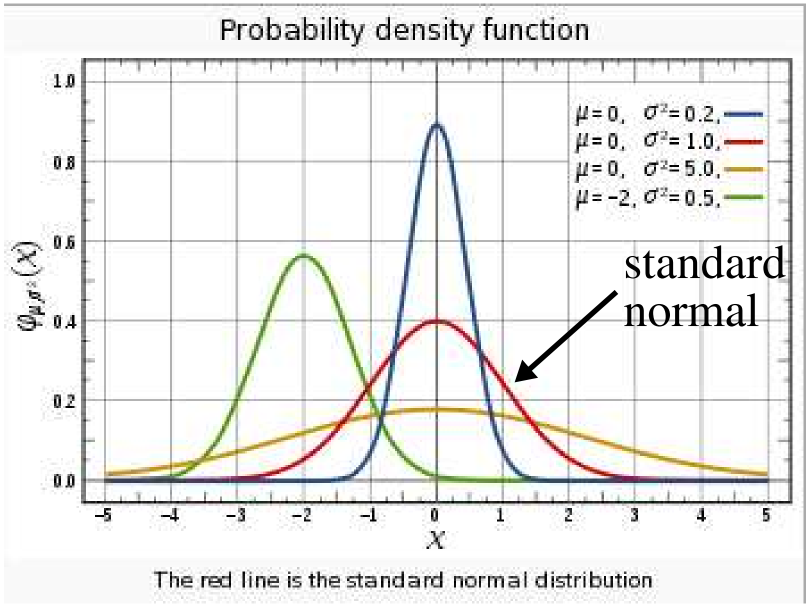
So a string of *10* binary values, one with worst case probability of occurrence of 1/*500* = 0.002, yields $-\log_2(0.002) = 8.966$ bits (the best you can achieve)

NIST reference distributions: **standard normal**

$$f(x;\mu,\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma}}$$

Normal (Gaussian) probability density function (wikipedia reference)

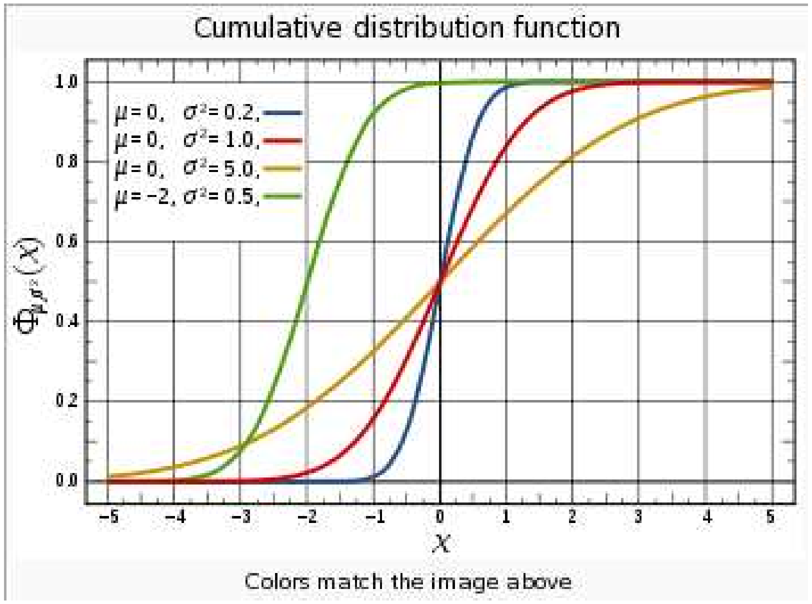
Probability density function



standard normal

The red line is the standard normal distribution

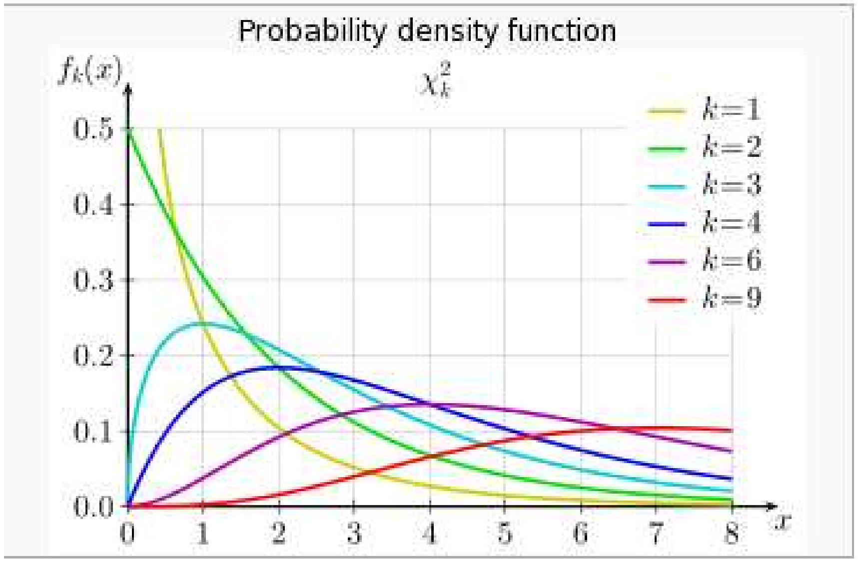
Cumulative distribution function

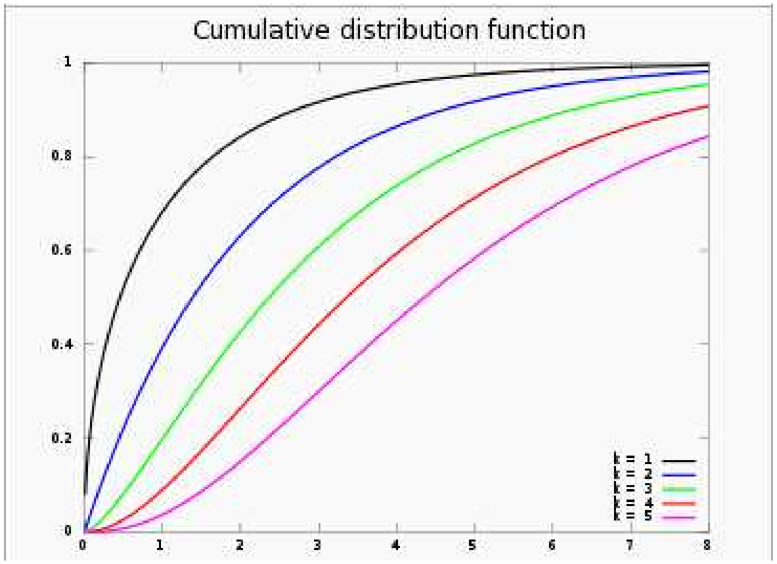


Colors match the image above

Probability Distributions

And **chi-square**(χ^2)





(wikipedia reference)

The chi-squared distribution with k *degrees of freedom* is the distribution associated with the sum of the squares of *k* **independent standard normal** random variables

Degrees of freedom: number of values in the final calculation of a statistic that are free to vary

NIST uses *chi-squared* tests to measure the 'goodness of fit' of an observed distribution and a theoretical one

NIST Test Suite

For NIST tests, if the bit sequence-under-test is non-random, then the calculated test statistic will fall in the *extreme regions* of the reference distribution

The NIST Test Suite has 15 tests -- for many of them, it is assumed the bit sequence is large, on order of 10^3 to 10^7

1) Frequency (Monobit) Test ($n > 100$)

Analyzes the proportion of ’0’s and ’1’s in the entire sequence, i.e., assesses the closeness of the fraction of ’1’s to 0.5

ALL SUBSEQUENT tests depend on the passing of this test!

Bit sequence is converted to ’1’s and ’-1’s using $X_i = 2\varepsilon - 1$ (ε_i represent the individual bits in the sample)

Test statistic is s_{obs} : absolute value of the sum of the X_i divided by the sqrt(n) (n is the sequence length)

NIST Test Suite

The reference distribution for the test statistic is *half normal, i.e.,* if z is distributed as normal, $|z|$ is distributed as half normal

For example, if ε = *1011010101* then $n = 10$ and $S_n = 1 + (-1) + 1 + 1 + (-1) + 1 + (-1) + 1 + (-1) + 1 = 2$

Test statistic:

$$s_{\text{obs}} = \frac{|S_n|}{\sqrt{n}} = \frac{|2|}{\sqrt{10}} = 0.63245532$$

Compute the *P-value* = **erfc**(s_{obs}/sqrt(2)), where *erfc* is the complementary error function

$$\text{erfc}\left(\frac{0.63245532}{\sqrt{2}}\right) = 0.527089$$

If the *P-value* is < 0.01, then conclude the sequence is non-random

Large values of s_{obs}, which are caused by large numbers of '0's or '1's, yield small *P-values*

NIST Test Suite

2) Frequency Test within a Block ($n > 100$, M is block size, N is # bits/block (Select $M >= 20$, $M > 0.01n$ and $N < 100$)

Analyzes the proportion of '1's within *M-bit* blocks to determine if they are ~$M/2$

Chi-squared is used as the reference distribution

Small *P-values* (computed from incomplete gamma function) indicate that at least one of the blocks has a large deviation in '1's from the expected of 0.5

3) Runs Test ($n > 100$)

Analyzes the total number of *runs* (uninterrupted sequences of identical bits), and determines whether the *oscillation* between '0's and '1's is **too fast** or **too slow**

Runs the **Frequency** test first, if the sequence fails, *P-value* set to 0

Computes test statistic by looking at each bit and its successor, if different add 1 to sum, else 0, e.g.,

ε = 1 00 11 0 1 0 11 generates test statistic (1+0+1+0+1+1+1+1+0) + 1 = 7

NIST Test Suite

4) Longest Run of Ones in a Block ($n > 128$)

(M set according to n, min $n = 128$, M -> 8, $n = 6272$, M -> 128, ...)

Analyzes the longest run of ’1’s within M-bit blocks, and determine if it is consistent with the length of the longest run expected in a random sequence

The larger the block size M (and corresponding n), the more partitions that exist in the ’binning’ table (see doc)

This makes sense since the probability that a longer sequence of ’1’s occurs increases as the bit sequence increases

5) Binary Matrix Rank Test (n > 38,912)

(M = rows of each matrix, Q = columns, code set with both = 32, also need at least 38 matricies)

Analyzes the linear dependence among ’fixed length’ substrings in the sequence, and determines if the *number of ranks* of size M, $M\text{-}1$ and those less than this match the expected number

NIST Test Suite

5) Binary Matrix Rank Test (*cont.*)

Rank indicates the number of rows that are **linearly independent**

The larger the rank for a given matrix, e.g., as it approach full rank or M, the more 'information' it posseses

6) Discrete Fourier Transform Test ($n > 1000$)

Analyzes the peak heights in the frequency spectrum of the sequence, and determines if there are *periodic* features (repeating patterns close to each other)

The test statistic is the **normalized difference** between the observed and expected number of frequency components that are beyond the 95% threshold

Test fails if number of peaks exceeding the 95% threshold is significantly different than 5%

The threshold is computed as a function of n

NIST Test Suite

7) Non-overlapping Template Matching Test (n > 100 with N = 8)

($M > 0.01 * n$, N can be changed but should be <= 100)

Analyzes the bit sequence for the number of times **pre-specified** target strings occur, to determine if too many occurrences of a *non-periodic* patterns occur

An m-bit window is used to search of a specific m-bit pattern

- If not found, the window slides by 1 bit position (overlapping)
- If found, window is reset to bit after the found pattern, and search continues

m is the length of each template B defined in the template library

- Sequences for $m = 2, 3 \ldots 10$ have been provided, sequences with $m = 9$ or 10 should be specified

Bit sequence is partitioned into N sequences of length M (N is fixed at 8)

- For example, if $n = 1000$, and $N = 8$, then $M = 125$

8) Overlapping Template Matching Test ($n > 10^6$ with N = 8)

Similar to the above but ’hits’ slide the window 1 position -- M = 1032 and N =968

NIST Test Suite

9) Maurer’s "Universal Statistical" Test ($n > 387{,}840$)

Analyzes the bit sequence to determine the **level of compression** that can be achieved (without loss of information)

The bit sequence is partitioned into two segments

- An **initialization segment** consisting of Q M-bit non-overlapping blocks
  The unique sequences in the M-bit blocks define a table where the last occurrence of each M-bit block is noted in the table
- A **test segment** consisting of K M-bit non-overlapping blocks
  The tests segment is scanned to determine the number of blocks since the last occurrence of the same M-bit block

Sums of distances between identical blocks is computed for the test statistic

NIST Test Suite

10) Linear Complexity Test ($n > 10^6$ + other restrictions on N and M)

Analyzes the bit sequence to determine the length of the smallest set of LFSRs needed to reproduce the sequence

Partition the bit sequence into *N* independent blocks of *M* bits

Use Berlekamp-Massey algorithm to determine the linear complexity (L_i) of each block

Here, L_i is the length of the shortest LFSR that can generate all bits in block *i*

11) Serial Test (choose m and n such that m < floor($\log_2 n$) -2)

Analyzes the bit sequence to determine the frequency of all possible (2^m) overlapping *m*-bit patterns, to determine if the number is uniform for all possible patterns

Every *m*-bit sequence has the same chance, as any other, of appearing in the bit sequence, so the distribution is expected to be uniform

NIST Test Suite

12) Approximate Entropy Test (choose m and n such that $m < \text{floor}(\log_2 n) - 5$)

Similar to Serial Test, this test analyzes the frequency of all possible overlapping *m*-bit patterns

The difference is that here we compare the frequency of overlapping blocks of **two consecutive/adjacent lengths** (m and $m + 1$) against the expected result

For example, assume $\varepsilon = 0100110101$,

Then the overlapping blocks become 010, 100, 001, 011, 110, 101, 010, 101, 010, 101

The number of occurrences for each possible (2^3) *m*-bit string are tallied, #000 = 0, #001 = 1, #010 = 3, etc and converted to **frequencies**

Entropy formula is applied using these counts

The process is repeated for m+1-bit strings and a test statistic computed

NIST Test Suite

13) Cummulative Sums Test ($n > 100$)

Analyzes the bit sequence to determine if the cummulative sum of incrementally increasing (decreasing) partial sequences is too large or too small

The bit sequence is converted to (1, -1) and sums are computed by simply adding one additional bit in the sequence at each step

- The largest excusion from zero is used as the test statistic

Large values of the test statistic indicate that there are too many ’1’s or ’0’s early in the sequence (for forward) or late in the sequence (for reverse)

Small values indicate that the ’1’s and ’0’s are intermixed TOO evenly

14 & 15) Random Excursions (Variant) Test ($n > 10^6$)

See documentation

NIST Test Suite

Interpretation of results (written to experiments/AlgorithmTesting/finalAnalysisReport.txt)

- Examine the number of bit sequences that pass each test

  For each sequence, a *P-value* is produced and is used to determine if the sequence passes the test, i.e., if *P-value* $> \alpha$

  At a significant level of $\alpha = 0.01$, about 1% of the sequences are expected to fail

- Examine the distribution of *P-values* for uniformity (using at least **55 sequences**)

  The result file contains categories C1 through C10, which partition the unit interval (0 to 1) into 10 equal sized segments

  The *P-values* computed from the sequences are 'binned' into these categories

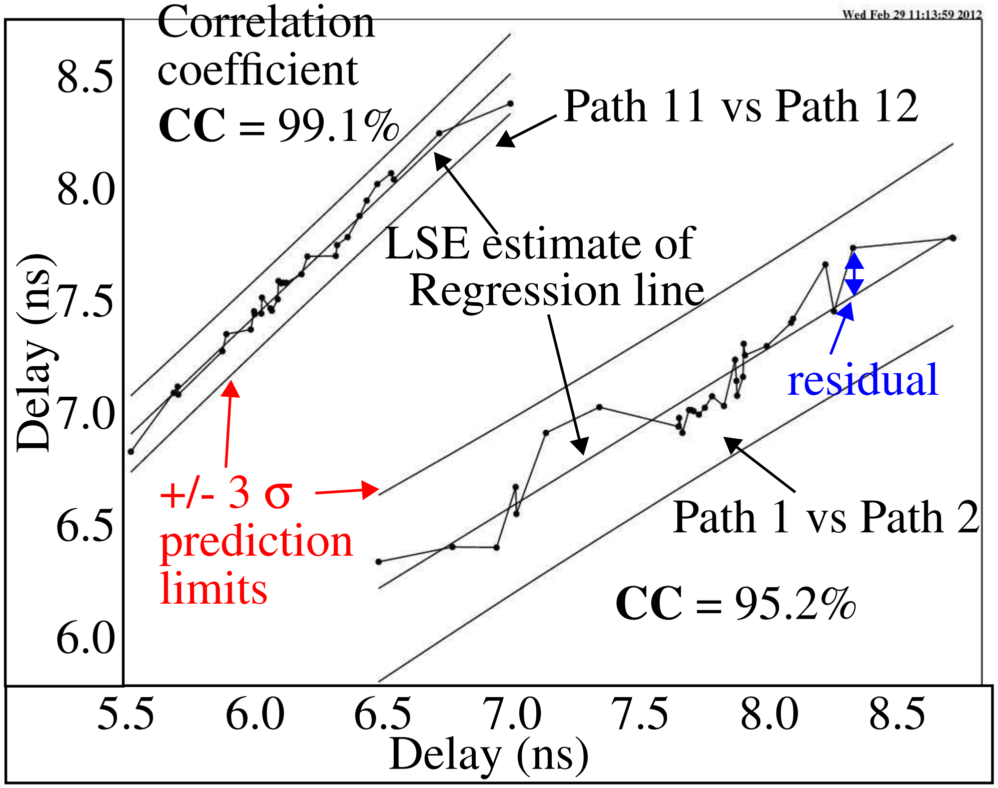
  The expected distribution is **uniform**

  A *P-value* of the *P-values* can also be computed using a chi-squared statistic

Regression Analysis

A statistical technique that examines the relationship between one dependent variable and one or more independent variables

For 2 variables, a regression plot can be created easily



30 Chips

90 nm IBM technology

Design-macro-under-test
Floating Point Unit

Regression line 'tracks' **chip-to-chip** process variations

Dispersion around reg. line represents measurement error and **within-die variation**

Correlation Coefficient (CC)

Given a sample $[(X_1, Y_1), (X_2, Y_2),\ldots,(X_n, Y_n)]$ is a random sample of *n* paired values of the random variables *X* and *Y*.

The **sample correlation coefficient** is:

$$CC = \frac{\sum(X_i - \bar{X})(Y_i - \bar{Y})}{\left[\sum(X_i - \bar{X})^2 \sum(Y_i - \bar{Y})^2\right]^{1/2}} \qquad -1 \le CC \le 1$$

A **test statistic** for testing the H_0 hypothesis of **zero** correlation between *X* and *Y*:

$$t^* = \frac{CC\sqrt{n-2}}{\sqrt{1 - CC^2}} \qquad \text{p-value} = 2P\{t(n-2) > t^*\}$$

Regression Analysis

Given a sample $\{(X_1, Y_1), (X_2, Y_2),...,(X_n, Y_n)\}$ of paired values from random variables X and Y, **least squares estimate** of the **regression line** is:

$$\widehat{Y} = b_0 + b_1 X$$

where

$$b_1 = \frac{\sum(X_i - \bar{X})(Y_i - \bar{Y})}{\sum(X_i - \bar{X})^2}$$

$$b_0 = \bar{Y} - b_1 \bar{X}$$

Sample variance in regression, MSE, is defined as:

$$MSE = \frac{\sum_{i=1}^{n} (Y_i - \hat{Y})^2}{n-2}$$

(Two degrees of freedom are lost b/c of b_0 and b_1)

Regression Analysis

Limits can be defined in several ways, depending on your needs:

- Confidence Interval for mean response of Y_h at X_h

$$Y = b_0 + b_1 x_i \pm t\sqrt{MSE}\sqrt{\frac{1}{n} + \frac{(X_h - \bar{X})^2}{\sum_{i=1}^{n}(X_i - \bar{X})^2}}$$

- Prediction Interval for a new observation

$$Y = b_0 + b_1 x_i \pm t\sqrt{MSE}\sqrt{1 + \frac{1}{n} + \frac{(X_h - \bar{X})^2}{\sum_{i=1}^{n}(X_i - \bar{X})^2}}$$