PASSsign statistical analysis

Author: Philipp Gühring

Date: 23.04.2014

Analysis Concept

Public key operations have certain requirements, which I am trying to check in this study.

I am taking a look at the following operations:

- Private key generation (transforming randomness/entropy into a private key)
- Public key generation (transforming a private key into a public key)
- Signature (transforming a private key into a signature)

Those operations have certain wanted and unwanted characteristics:

- Private key generation SHOULD use the retrieved randomness in the private key
    - But it might take them only as a starting point and do some deterministic changes to the random bits to get to a useable key
- All of the randomness that ends up in the private keys SHOULD pass randomness tests
- A majority of the randomness that ends up in the private key MUST pass randomness tests
- The public key generation MUST NOT leak private key bits into the public key
- The Signature MUST NOT leak private key bits

The following methods are used to analyze the given algorithms for the expected (absence of) characteristics:

- The Sourcecode of the algorithm implementation is extended to dump the following data to files:
    - The randomness retrieved
    - The private key generated
    - The public key generated
    - The signatures generated
- The extended application has being executed 100 times
    - Every execution generated 1 private/public key and 100 signatures
- Then the recorded files are being analyzed
    - The entropy of the bits in the private key
    - The entropy of the bits in the public key

- The entropy of the bits in the signature
- For correlations between randomness and private key
- For correlations between private key and public key bits
- For correlations between private key and signature bits

Technical details

The analyzed version of the software was the Github repository from April 2014
https://github.com/NTRUOpenSourceProject/ntru-crypto/

The computer that was used for the analysis is a Intel Xeon based system with a 32 Bit build of PASSsign, running on Debian.

Results

I did not find any correlations between private key and public key bits.

I did not find any correlations between private key and signature bits.

But other interesting characteristics were found:

Private Key Entropy

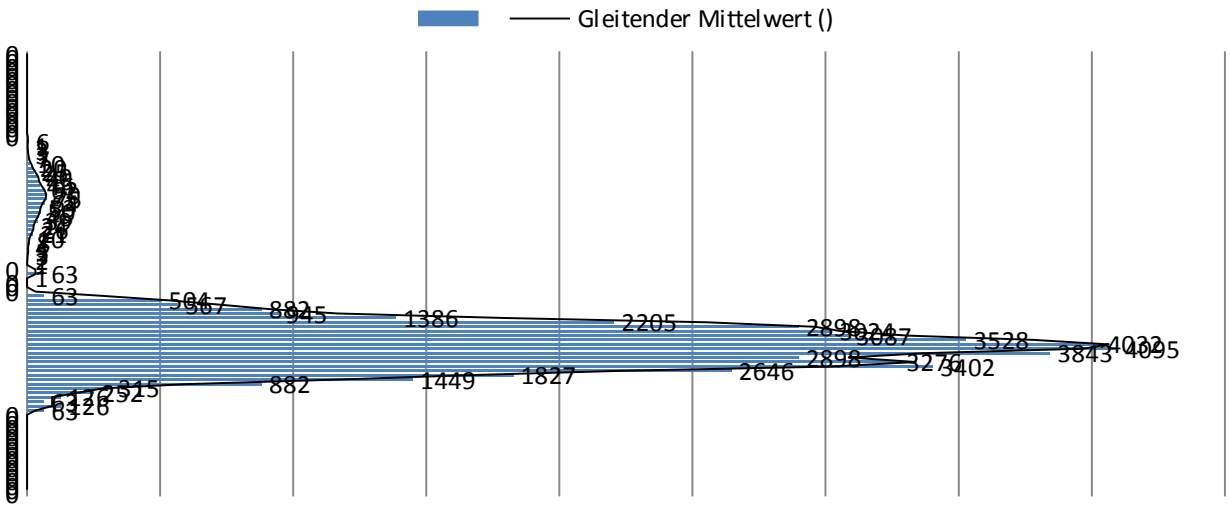
The PASSsign private key consists of 769 values, each of the values is either -1,0,1 encoded as 64 Bit signed integers. Theoretically, the private key could be saved with 2 bits per value, the other 62 Bits are not needed, and are always either 0 (if the value is 0 or 1) or 1 if the value is -1.

Value	63 Bits	1 Bit
-1	1	1
0	0	0
1	0	1

Given a random distribution of those 3 values, I would expect 98,4% (63 of 64) of the bits to be 1 in 33.3% of the cases and 1.56 % (1 of 64) of the bits to be 1 in 66.6% of the cases.

The entropy was analyzed for every bit (6152 Bytes => 49216 Bits), since we had 100 samples, we calculated for every bit the percentage that a certain bit is 1. 0% means that this bit was always 0, 50% means that half of the bits are 0 and half of the bits are 1, and 100% means that the bit was 1 all the time.

Then I analyzed the distribution of those 49216 bits:

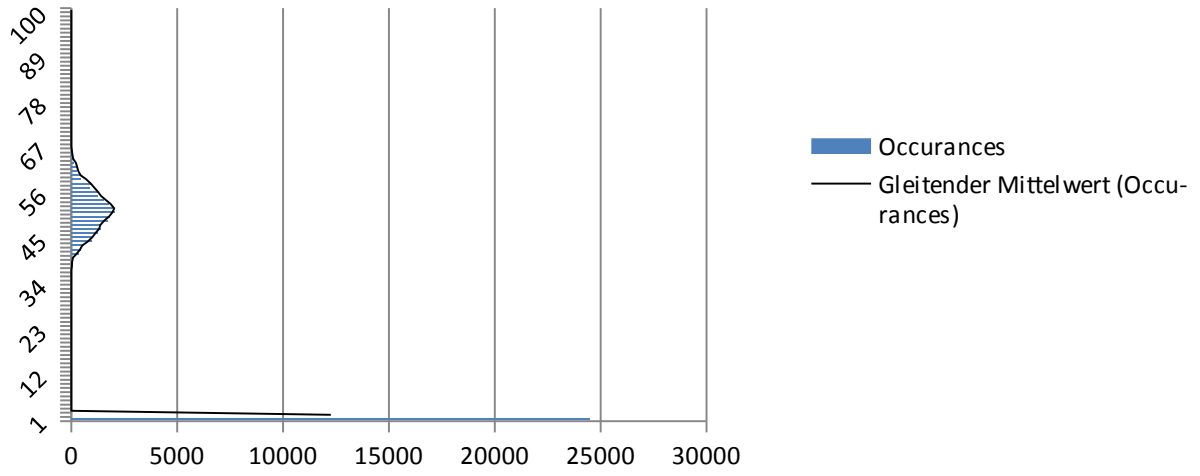


There is a huge bucket around 33.3%, which contains 98.31% of all values, which I expected to be there, since there is an equal distribution of the value -1,0,1, and 0 and 1 will result in in a 0 in 63 of the bits, and -1 will result in a 1 in 63 of those bits, so in average a bit is in 33% of the cases a 1.
The large distribution from 19 to 45 (delta: 26) around the peak is a bit surprising, and might be a sign for bad entropy, but it could be ok.
The second largest bucket around 66.6% which contains 1.56% of all values, which is also expected due to last of the 64 bits.

I do not have an explanation for the peak around 50% (with 64 instances which is 0.13% of all values). We should research, why it is here.
It is also interesting where this peak happens: In one consecutive area from Bits 9217 to 9279 ! What is special about that area?

Public Key Entropy

Public keys are also 769 values stored as 64 bit signed integers, but they come from a Fourier Transformation, and therefore they have a much larger value-space. My initial analysis showed that it likely only needs 32 of the 64 bits, and when asked about it, the developer agreed that it could be stored with 32 bits only, but 64 bits are needed again later on for calculations.



49.80% of the bits are always zero.
(which is due to the unused 32 of the 64 bits)
Nearly 50% of the bits are all zero in all public keys! This could be too much.
Why are there 0.2% non-zeros?
The spread is 67, which looks ok to me.
The peak is at 50, which is good as I expected it.

Signature Entropy

To be tested …

Random number Entropy

To be tested …

Public Key Secret Key Bit correlation

I have not found any correlations between bits in the public key and bits in the secret key beyond fixed bits. (Bits that are always zero in the public key obviously correlate with Bits that are 66.6% zero)

I am still thinking about what is the best way to visualize it.

…

To be continued