

Re: Pseudorandom Hashing

Source: <http://sci.tech-archive.net/Archive/sci.electronics.design/2004-09/6048.html>

From: Mac (*foo_at_bar.net*)

Date: 09/24/04

Date: Fri, 24 Sep 2004 03:48:30 GMT

On Thu, 23 Sep 2004 15:26:18 -0700, Tim Wescott wrote:

> *Jim Thompson wrote:*

>> *On Thu, 23 Sep 2004 14:40:28 -0700, Tim Wescott*

>> *<tim@wescottmospamdesign.com> wrote:*

>>

>>

>>> *I am having trouble coming up with the right keywords to do a web*

>>> *search, so help me out here:*

>>>

>>> *There is a technique where, to significantly reduce the probability of*

>>> *getting a long string of zeros, a message is run through a CRC*

>>> *generator, and the output bits are taken off. The transmitted message*

>>> *is thoroughly hashed, yet it is a simple matter of a shift register and*

>>> *some XOR gates to decode the message on the other end.*

>>>

>>> *I thought I knew how to do this, yet in trying to actually make it work*

>>> *I find that over half of my brain cells appear to be attending a*

>>> *management seminar.*

>>

>>

>> *"attending a management seminar".... I like that ;-)*

>>

>>

>>> *So, know where I can find out how to do this right? "pseudorandom" and*

>>> *"hash" get me tons of cryptography, but not what I'm looking for.*

>>>

>>> *Thanks.*

>>

>>

>> *My mind is slow today also, but isn't it the same in AND out...*

>>

>> *XMIT: 2-bit SR, XOR the two SR Q's together to get output*

>>

>> *RECVR: SAME*

>>

>> *...Jim Thompson*

> *IIRC you XOR and feed back on the input, and just XOR on the output.*

>

> *But IIRC then I wouldn't need to ask.*

I think you are talking about two things.

One is a linear feedback shift register pseudo random noise generator. (LFSR PRNG). These can be made with just a shift register (or d flip flops) and a few xor gates.

The other is some kind of encoder, such as a 4-bit to 5-bit encoder. (Two of them together can make an 8-bit/10-bit encoder)

Since there are only 16 4-bit sequences, but 32 5-bit sequences, you can map each 4-bit sequence to a 5-bit sequence with a balance of ones and zeroes. There is a standard way to do this. I believe ethernet does this (well, the 8b/10b version) to guarantee that there is no net DC in the ethernet signal. Fibre Channel also does this to guarantee that there is no DC, and also to ensure that there are enough edges for clock recovery.

Let me return to the topic of the LFSR. There is a whole literature on these things, and the combinations that work are well known. I would advise you to find a list of the known good ones rather than trying to find one by trial and error. I think the A of E has a section on this. Could be wrong.

AFAIK, the LFSR does not decrease the probability of having long runs of zeroes provided that the data going over the wire are random.

On the other hand, if the data going over the wire are known to contain long runs of zeroes, then I guess that the LFSR would offer an improvement to that situation, but it couldn't guarantee it by any means.

So if long runs of zeroes are an inconvenience, then the LFSR might solve your problem. But if the long runs of zeroes are impermissible, then you need to go with something more like the 8b/10b encoding.

Good luck.

--Mac