

Re: Pseudorandom Hashing

Source: <http://sci.tech-archive.net/Archive/sci.electronics.design/2004-09/6047.html>

From: Mike Monett (*no_at_spam.com*)

Date: 09/24/04

Date: Thu, 23 Sep 2004 23:31:13 -0400

Tim Wescott wrote:

[...]

> *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.*
>
> *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.*
>
> *Tim Wescott*
> *Wescott Design Services*
> <http://www.wescottdesign.com>

Tim,

It's called scrambling – but are you sure you want to do it? Recall the probability of a given bit pattern in a random sequence is $1/2^n$, where n is the length of the pattern.

So in any random group of 8 bits, you have a $1/256$ chance of finding the bit pattern 000000. If this matches or partially matches a similar pattern in the data, the output will be a long string of zeros. If this causes a system failure, it may occur fairly often.

Many encoding schemes have no dc component and can tolerate long strings of ones or zeros. The penalty is low efficiency. Other methods limit the length of the runs, for example the RLL codes that were used in hard disk drives. Optical data transmission uses this technique, and I believe

Ethernet.

One method that achieves high bit efficiency and has no dc component is the old MFM, or Miller code that was used on hard disk drives prior to RLL. It is very simple to encode and decode, and has twice the efficiency of Manchester encoding.

So if the data absolutely has got to get there, any of the RLL encoding methods may be a better choice than data scrambling.

Mike