A PAINLESS GUIDE TO CRC ERROR DETECTION ALGORITHMS

"Everything you wanted to know about CRC algorithms, but were afraid to ask for fear that errors in your understanding might be detected."

Version : 3.
Date : 19 August 1993.
Author : Ross N. Williams.
Net : ross@guest.adelaide.edu.au.
FTP : ftp.adelaide.edu.au/pub/rocksoft/crc_v3.txt
Company : Rocksoft^tm Pty Ltd.
Snail : 16 Lerwick Avenue, Hazelwood Park 5066, Australia.
Fax : +61 8 373-4911 (c/- Internode Systems Pty Ltd).
Phone : +61 8 379-9217 (10am to 10pm Adelaide Australia time).

Thanks : Thanks to Jean-loup Gailly (jloup@chorus.fr) and Mark Adler (me@quest.jpl.nasa.gov) who both proof read this document and picked out lots of nits as well as some big fat bugs.

Table of Contents

Abstract
1. Introduction: Error Detection
2. The Need For Complexity
3. The Basic Idea Behind CRC Algorithms
4. Polynomical Arithmetic
5. Binary Arithmetic with No Carries
6. A Fully Worked Example
7. Choosing A Poly
8. A Straightforward CRC Implementation
9. A Table-Driven Implementation
10. A Slightly Mangled Table-Driven Implementation
11. "Reflected" Table-Driven Implementations
12. "Reversed" Polys
13. Initial and Final Values
14. Defining Algorithms Absolutely
15. A Parameterized Model For CRC Algorithms
16. A Catalog of Parameter Sets for Standards
17. An Implementation of the Model Algorithm
18. Roll Your Own Table-Driven Implementation
19. Generating A Lookup Table
20. Summary
21. Corrections
A. Glossary
B. References
C. References I Have Detected But Haven't Yet Sighted

Abstract

This document explains CRCs (Cyclic Redundancy Codes) and their table-driven implementations in full, precise detail. Much of the literature on CRCs, and in particular on their table-driven

implementations, is a little obscure (or at least seems so to me). This document is an attempt to provide a clear and simple no-nonsense explanation of CRCs and to absolutely nail down every detail of the operation of their high-speed implementations. In addition to this, this document presents a parameterized model CRC algorithm called the "Rocksoft^tm Model CRC Algorithm". The model algorithm can be parameterized to behave like most of the CRC implementations around, and so acts as a good reference for describing particular algorithms. A low-speed implementation of the model CRC algorithm is provided in the C programming language. Lastly there is a section giving two forms of high-speed table driven implementations, and providing a program that generates CRC lookup tables.

1. Introduction: Error Detection

The aim of an error detection technique is to enable the receiver of a message transmitted through a noisy (error-introducing) channel to determine whether the message has been corrupted. To do this, the transmitter constructs a value (called a checksum) that is a function of the message, and appends it to the message. The receiver can then use the same function to calculate the checksum of the received message and compare it with the appended checksum to see if the message was correctly received. For example, if we chose a checksum function which was simply the sum of the bytes in the message mod 256 (i.e. modulo 256), then it might go something as follows. All numbers are in decimal.

```
   Message                    :  6 23  4
   Message with checksum      :  6 23  4 33
   Message after transmission :  6 27  4 33
```

In the above, the second byte of the message was corrupted from 23 to 27 by the communications channel. However, the receiver can detect this by comparing the transmitted checksum (33) with the computer checksum of 37 (6 + 27 + 4). If the checksum itself is corrupted, a correctly transmitted message might be incorrectly identified as a corrupted one. However, this is a safe-side failure. A dangerous-side failure occurs where the message and/or checksum is corrupted in a manner that results in a transmission that is internally consistent. Unfortunately, this possibility is completely unavoidable and the best that can be done is to minimize its probability by increasing the amount of information in the checksum (e.g. widening the checksum from one byte to two bytes).

Other error detection techniques exist that involve performing complex transformations on the message to inject it with redundant information. However, this document addresses only CRC algorithms, which fall into the class of error detection algorithms that leave the data intact and append a checksum on the end. i.e.:

```
      <original intact message> <checksum>
```

2. The Need For Complexity

In the checksum example in the previous section, we saw how a corrupted message was detected using a checksum algorithm that simply

sums the bytes in the message mod 256:

```
   Message                    :  6 23  4
   Message with checksum      :  6 23  4 33
   Message after transmission :  6 27  4 33
```

A problem with this algorithm is that it is too simple. If a number of random corruptions occur, there is a 1 in 256 chance that they will not be detected. For example:

```
   Message                    :  6 23  4
   Message with checksum      :  6 23  4 33
   Message after transmission :  8 20  5 33
```

To strengthen the checksum, we could change from an 8-bit register to a 16-bit register (i.e. sum the bytes mod 65536 instead of mod 256) so as to apparently reduce the probability of failure from 1/256 to 1/65536. While basically a good idea, it fails in this case because the formula used is not sufficiently "random"; with a simple summing formula, each incoming byte affects roughly only one byte of the summing register no matter how wide it is. For example, in the second example above, the summing register could be a Megabyte wide, and the error would still go undetected. This problem can only be solved by replacing the simple summing formula with a more sophisticated formula that causes each incoming byte to have an effect on the entire checksum register.

Thus, we see that at least two aspects are required to form a strong checksum function:

   WIDTH: A register width wide enough to provide a low a-priori probability of failure (e.g. 32-bits gives a 1/2^32 chance of failure).

   CHAOS: A formula that gives each input byte the potential to change any number of bits in the register.

Note: The term "checksum" was presumably used to describe early summing formulas, but has now taken on a more general meaning encompassing more sophisticated algorithms such as the CRC ones. The CRC algorithms to be described satisfy the second condition very well, and can be configured to operate with a variety of checksum widths.

3. The Basic Idea Behind CRC Algorithms

Where might we go in our search for a more complex function than summing? All sorts of schemes spring to mind. We could construct tables using the digits of pi, or hash each incoming byte with all the bytes in the register. We could even keep a large telephone book on-line, and use each incoming byte combined with the register bytes to index a new phone number which would be the next register value. The possibilities are limitless.

However, we do not need to go so far; the next arithmetic step suffices. While addition is clearly not strong enough to form an effective checksum, it turns out that division is, so long as the divisor is about as wide as the checksum register.

The basic idea of CRC algorithms is simply to treat the message as an enormous binary number, to divide it by another fixed binary number, and to make the remainder from this division the checksum. Upon receipt of the message, the receiver can perform the same division and compare the remainder with the "checksum" (transmitted remainder).

Example: Suppose the the message consisted of the two bytes (6,23) as in the previous example. These can be considered to be the hexadecimal number 0617 which can be considered to be the binary number 0000-0110-0001-0111. Suppose that we use a checksum register one-byte wide and use a constant divisor of 1001, then the checksum is the remainder after 0000-0110-0001-0111 is divided by 1001. While in this case, this calculation could obviously be performed using common garden variety 32-bit registers, in the general case this is messy. So instead, we'll do the division using good-'ol long division which you learnt in school (remember?). Except this time, it's in binary:

```
          ...0000010101101 = 00AD =  173 = QUOTIENT
          ____-___-___-___-
9= 1001 ) 0000011000010111 = 0617 = 1559 = DIVIDEND
DIVISOR   0000.,,....,.,,,
          ----.,,....,.,,,
           0000,,....,.,,,
           0000,,....,.,,,
           ----,,....,.,,,
            0001,....,.,,,
            0000,....,.,,,
            ----,....,.,,,
             0011....,.,,,
             0000....,.,,,
             ----....,.,,,
              0110...,.,,,
              0000...,.,,,
              ----...,.,,,
               1100..,.,,,
               1001..,.,,,
               ====..,.,,,
                0110.,.,,,
                0000.,.,,,
                ----.,.,,,
                 1100,.,,,
                 1001,.,,,
                 ====,.,,,
                  0111.,,,
                  0000.,,,
                  ----.,,,
                   1110,,,
                   1001,,,
                   ====,,,
                    1011,,
                    1001,,
                    ====,,
                     0101,
                     0000,
                     ----
                      1011
                      1001
```

```
                    ====
                    0010 = 02 = 2 = REMAINDER
```

In decimal this is "1559 divided by 9 is 173 with a remainder of 2".

Although the effect of each bit of the input message on the quotient is not all that significant, the 4-bit remainder gets kicked about quite a lot during the calculation, and if more bytes were added to the message (dividend) it's value could change radically again very quickly. This is why division works where addition doesn't.

In case you're wondering, using this 4-bit checksum the transmitted message would look like this (in hexadecimal): 06172 (where the 0617 is the message and the 2 is the checksum). The receiver would divide 0617 by 9 and see whether the remainder was 2.

4. Polynomical Arithmetic

While the division scheme described in the previous section is very very similar to the checksumming schemes called CRC schemes, the CRC schemes are in fact a bit weirder, and we need to delve into some strange number systems to understand them.

The word you will hear all the time when dealing with CRC algorithms is the word "polynomial". A given CRC algorithm will be said to be using a particular polynomial, and CRC algorithms in general are said to be operating using polynomial arithmetic. What does this mean?

Instead of the divisor, dividend (message), quotient, and remainder (as described in the previous section) being viewed as positive integers, they are viewed as polynomials with binary coefficients. This is done by treating each number as a bit-string whose bits are the coefficients of a polynomial. For example, the ordinary number 23 (decimal) is 17 (hex) and 10111 binary and so it corresponds to the polynomial:

```
   1*x^4 + 0*x^3 + 1*x^2 + 1*x^1 + 1*x^0
```

or, more simply:

```
   x^4 + x^2 + x^1 + x^0
```

Using this technique, the message, and the divisor can be represented as polynomials and we can do all our arithmetic just as before, except that now it's all cluttered up with Xs. For example, suppose we wanted to multiply 1101 by 1011. We can do this simply by multiplying the polynomials:

```
(x^3 + x^2 + x^0)(x^3 + x^1 + x^0)
= (x^6 + x^4 + x^3
 + x^5 + x^3 + x^2
 + x^3 + x^1 + x^0) = x^6 + x^5 + x^4 + 3*x^3 + x^2 + x^1 + x^0
```

At this point, to get the right answer, we have to pretend that x is 2 and propagate binary carries from the 3*x^3 yielding

```
   x^7 + x^3 + x^2 + x^1 + x^0
```

It's just like ordinary arithmetic except that the base is abstracted and brought into all the calculations explicitly instead of being there implicitly. So what's the point?

The point is that IF we pretend that we DON'T know what x is, we CAN'T perform the carries. We don't know that 3*x^3 is the same as x^4 + x^3 because we don't know that x is 2. In this true polynomial arithmetic the relationship between all the coefficients is unknown and so the coefficients of each power effectively become strongly typed; coefficients of x^2 are effectively of a different type to coefficients of x^3.

With the coefficients of each power nicely isolated, mathematicians came up with all sorts of different kinds of polynomial arithmetics simply by changing the rules about how coefficients work. Of these schemes, one in particular is relevant here, and that is a polynomial arithmetic where the coefficients are calculated MOD 2 and there is no carry; all coefficients must be either 0 or 1 and no carries are calculated. This is called "polynomial arithmetic mod 2". Thus, returning to the earlier example:

```
(x^3 + x^2 + x^0)(x^3 + x^1 + x^0)
= (x^6 + x^4 + x^3
 + x^5 + x^3 + x^2
 + x^3 + x^1 + x^0)
= x^6 + x^5 + x^4 + 3*x^3 + x^2 + x^1 + x^0
```

Under the other arithmetic, the 3*x^3 term was propagated using the carry mechanism using the knowledge that x=2. Under "polynomial arithmetic mod 2", we don't know what x is, there are no carries, and all coefficients have to be calculated mod 2. Thus, the result becomes:

```
= x^6 + x^5 + x^4 + x^3 + x^2 + x^1 + x^0
```

As Knuth [Knuth81] says (p.400):

> "The reader should note the similarity between polynomial arithmetic and multiple-precision arithmetic (Section 4.3.1), where the radix b is substituted for x. The chief difference is that the coefficient u_k of x^k in polynomial arithmetic bears little or no relation to its neighboring coefficients x^{k-1} [and x^{k+1}], so the idea of "carrying" from one place to another is absent. In fact polynomial arithmetic modulo b is essentially identical to multiple precision arithmetic with radix b, except that all carries are suppressed."

Thus polynomical arithmetic mod 2 is just binary arithmetic mod 2 with no carries. While polynomials provide useful mathematical machinery in more analytical approaches to CRC and error-correction algorithms, for the purposes of exposition they provide no extra insight and some encumbrance and have been discarded in the remainder of this document in favour of direct manipulation of the arithmetical system with which they are isomorphic: binary arithmetic with no carry.

5. Binary Arithmetic with No Carries

Having dispensed with polynomials, we can focus on the real arithmetic issue, which is that all the arithmetic performed during CRC calculations is performed in binary with no carries. Often this is called polynomial arithmetic, but as I have declared the rest of this document a polynomial free zone, we'll have to call it CRC arithmetic instead. As this arithmetic is a key part of CRC calculations, we'd better get used to it. Here we go:

Adding two numbers in CRC arithmetic is the same as adding numbers in ordinary binary arithmetic except there is no carry. This means that each pair of corresponding bits determine the corresponding output bit without reference to any other bit positions. For example:

```
 10011011
+11001010
 --------
 01010001
 --------
```

There are only four cases for each bit position:

```
0+0=0
0+1=1
1+0=1
1+1=0  (no carry)
```

Subtraction is identical:

```
 10011011
-11001010
 --------
 01010001
 --------
```

with

```
0-0=0
0-1=1  (wraparound)
1-0=1
1-1=0
```

In fact, both addition and subtraction in CRC arithmetic is equivalent to the XOR operation, and the XOR operation is its own inverse. This effectively reduces the operations of the first level of power (addition, subtraction) to a single operation that is its own inverse. This is a very convenient property of the arithmetic.

By collapsing of addition and subtraction, the arithmetic discards any notion of magnitude beyond the power of its highest one bit. While it seems clear that 1010 is greater than 10, it is no longer the case that 1010 can be considered to be greater than 1001. To see this, note that you can get from 1010 to 1001 by both adding and subtracting the same quantity:

```
1010 = 1010 + 0011
1010 = 1010 - 0011
```

This makes nonsense of any notion of order.

Having defined addition, we can move to multiplication and division. Multiplication is absolutely straightforward, being the sum of the first number, shifted in accordance with the second number.

```
        1101
      x 1011
        ----
        1101
       1101.
      0000..
     1101...
     -------
     1111111  Note: The sum uses CRC addition
     -------
```

Division is a little messier as we need to know when "a number goes into another number". To do this, we invoke the weak definition of magnitude defined earlier: that X is greater than or equal to Y iff the position of the highest 1 bit of X is the same or greater than the position of the highest 1 bit of Y. Here's a fully worked division (nicked from [Tanenbaum81]).

```
            1100001010
          ______________
10011 ) 11010110110000
        10011,,.,,....
        -----,,.,,....
         10011,.,,....
         10011,.,,....
         -----,.,,....
          00001.,,....
          00000.,,....
          -----.,,....
           00010,,....
           00000,,....
           -----,,....
            00101,....
            00000,....
            -----,....
             01011....
             00000....
             -----....
              10110...
              10011...
              -----...
               01010..
               00000..
               -----..
                10100.
                10011.
                -----.
                 01110
                 00000
                 -----
                  1110 = Remainder
```

That's really it. Before proceeding further, however, it's worth our while playing with this arithmetic a bit to get used to it.

We've already played with addition and subtraction, noticing that they are the same thing. Here, though, we should note that in this arithmetic A+0=A and A-0=A. This obvious property is very useful later.

In dealing with CRC multiplication and division, it's worth getting a feel for the concepts of MULTIPLE and DIVISIBLE.

If a number A is a multiple of B then what this means in CRC arithmetic is that it is possible to construct A from zero by XORing in various shifts of B. For example, if A was 0111010110 and B was 11, we could construct A from B as follows:

```
  0111010110
= .......11.
+ ....11....
+ ...11.....
  .11.......
```

However, if A is 0111010111, it is not possible to construct it out of various shifts of B (can you see why? - see later) so it is said to be not divisible by B in CRC arithmetic.

Thus we see that CRC arithmetic is primarily about XORing particular values at various shifting offsets.

6. A Fully Worked Example

Having defined CRC arithmetic, we can now frame a CRC calculation as simply a division, because that's all it is! This section fills in the details and gives an example.

To perform a CRC calculation, we need to choose a divisor. In maths marketing speak the divisor is called the "generator polynomial" or simply the "polynomial", and is a key parameter of any CRC algorithm. It would probably be more friendly to call the divisor something else, but the poly talk is so deeply ingrained in the field that it would now be confusing to avoid it. As a compromise, we will refer to the CRC polynomial as the "poly". Just think of this number as a sort of parrot. "Hello poly!"

You can choose any poly and come up with a CRC algorithm. However, some polys are better than others, and so it is wise to stick with the tried an tested ones. A later section addresses this issue.

The width (position of the highest 1 bit) of the poly is very important as it dominates the whole calculation. Typically, widths of 16 or 32 are chosen so as to simplify implementation on modern computers. The width of a poly is the actual bit position of the highest bit. For example, the width of 10011 is 4, not 5. For the purposes of example, we will chose a poly of 10011 (of width W of 4).

Having chosen a poly, we can proceed with the calculation. This is

simply a division (in CRC arithmetic) of the message by the poly. The only trick is that W zero bits are appended to the message before the CRC is calculated. Thus we have:

```
   Original message                : 1101011011
   Poly                            :      10011
   Message after appending W zeros : 11010110110000
```

Now we simply divide the augmented message by the poly using CRC arithmetic. This is the same division as before:

```
            1100001010 = Quotient (nobody cares about the quotient)
          ______________
10011 ) 11010110110000 = Augmented message (1101011011 + 0000)
=Poly  10011,,.,,....
        -----,,.,,....
         10011,.,,....
         10011,.,,....
         -----,.,,....
          00001.,,....
          00000.,,....
          -----.,,....
           00010,,....
           00000,,....
           -----,,....
            00101,....
            00000,....
            -----,....
             01011....
             00000....
             -----....
              10110...
              10011...
              -----...
               01010..
               00000..
               -----..
                10100.
                10011.
                -----.
                 01110
                 00000
                 -----
                  1110 = Remainder = THE CHECKSUM!!!!
```

The division yields a quotient, which we throw away, and a remainder, which is the calculated checksum. This ends the calculation.

Usually, the checksum is then appended to the message and the result transmitted. In this case the transmission would be: 11010110111110.

At the other end, the receiver can do one of two things:

   a. Separate the message and checksum. Calculate the checksum for the message (after appending W zeros) and compare the two checksums.

   b. Checksum the whole lot (without appending zeros) and see if it

comes out as zero!

These two options are equivalent. However, in the next section, we will be assuming option b because it is marginally mathematically cleaner.

A summary of the operation of the class of CRC algorithms:

1. Choose a width W, and a poly G (of width W).
2. Append W zero bits to the message. Call this M'.
3. Divide M' by G using CRC arithmetic. The remainder is the checksum.

That's all there is to it.

7. Choosing A Poly

Choosing a poly is somewhat of a black art and the reader is referred to [Tanenbaum81] (p.130-132) which has a very clear discussion of this issue. This section merely aims to put the fear of death into anyone who so much as toys with the idea of making up their own poly. If you don't care about why one poly might be better than another and just want to find out about high-speed implementations, choose one of the arithmetically sound polys listed at the end of this section and skip to the next section.

First note that the transmitted message T is a multiple of the poly. To see this, note that 1) the last W bits of T is the remainder after dividing the augmented (by zeros remember) message by the poly, and 2) addition is the same as subtraction so adding the remainder pushes the value up to the next multiple. Now note that if the transmitted message is corrupted in transmission that we will receive T+E where E is an error vector (and + is CRC addition (i.e. XOR)). Upon receipt of this message, the receiver divides T+E by G. As T mod G is 0, (T+E) mod G = E mod G. Thus, the capacity of the poly we choose to catch particular kinds of errors will be determined by the set of multiples of G, for any corruption E that is a multiple of G will be undetected. Our task then is to find classes of G whose multiples look as little like the kind of line noise (that will be creating the corruptions) as possible. So let's examine the kinds of line noise we can expect.

SINGLE BIT ERRORS: A single bit error means E=1000...0000. We can ensure that this class of error is always detected by making sure that G has at least two bits set to 1. Any multiple of G will be constructed using shifting and adding and it is impossible to construct a value with a single bit by shifting an adding a single value with more than one bit set, as the two end bits will always persist.

TWO-BIT ERRORS: To detect all errors of the form 100...000100...000 (i.e. E contains two 1 bits) choose a G that does not have multiples that are 11, 101, 1001, 10001, 100001, etc. It is not clear to me how one goes about doing this (I don't have the pure maths background), but Tanenbaum assures us that such G do exist, and cites G with 1 bits (15,14,1) turned on as an example of one G that won't divide anything less than 1...1 where ... is 32767 zeros.

ERRORS WITH AN ODD NUMBER OF BITS: We can catch all corruptions where E has an odd number of bits by choosing a G that has an even number of

bits. To see this, note that 1) CRC multiplication is simply XORing a constant value into a register at various offsets, 2) XORing is simply a bit-flip operation, and 3) if you XOR a value with an even number of bits into a register, the oddness of the number of 1 bits in the register remains invariant. Example: Starting with E=111, attempt to flip all three bits to zero by the repeated application of XORing in 11 at one of the two offsets (i.e. "E=E XOR 011" and "E=E XOR 110") This is nearly isomorphic to the "glass tumblers" party puzzle where you challenge someone to flip three tumblers by the repeated application of the operation of flipping any two. Most of the popular CRC polys contain an even number of 1 bits. (Note: Tanenbaum states more specifically that all errors with an odd number of bits can be caught by making G a multiple of 11).

BURST ERRORS: A burst error looks like E=000...000111...11110000...00. That is, E consists of all zeros except for a run of 1s somewhere inside. This can be recast as E=(10000...00)(1111111...111) where there are z zeros in the LEFT part and n ones in the RIGHT part. To catch errors of this kind, we simply set the lowest bit of G to 1. Doing this ensures that LEFT cannot be a factor of G. Then, so long as G is wider than RIGHT, the error will be detected. See Tanenbaum for a clearer explanation of this; I'm a little fuzzy on this one. Note: Tanenbaum asserts that the probability of a burst of length greater than W getting through is (0.5)^W.

That concludes the section on the fine art of selecting polys.

Some popular polys are:

16 bits:	(16,12,5,0)	[X25 standard]
	(16,15,2,0)	["CRC-16"]
32 bits:	(32,26,23,22,16,12,11,10,8,7,5,4,2,1,0)	[Ethernet]

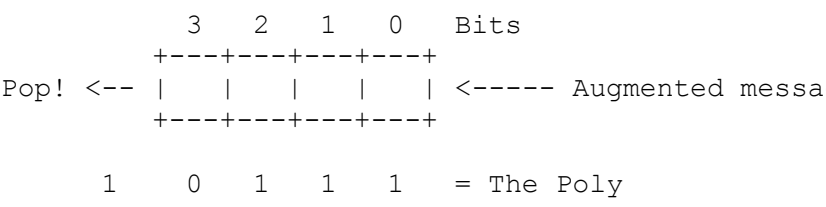
8. A Straightforward CRC Implementation

That's the end of the theory; now we turn to implementations. To start with, we examine an absolutely straight-down-the-middle boring straightforward low-speed implementation that doesn't use any speed tricks at all. We'll then transform that program progessively until we end up with the compact table-driven code we all know and love and which some of us would like to understand.

To implement a CRC algorithm all we have to do is implement CRC division. There are two reasons why we cannot simply use the divide instruction of whatever machine we are on. The first is that we have to do the divide in CRC arithmetic. The second is that the dividend might be ten megabytes long, and todays processors do not have registers that big.

So to implement CRC division, we have to feed the message through a division register. At this point, we have to be absolutely precise about the message data. In all the following examples the message will be considered to be a stream of bytes (each of 8 bits) with bit 7 of each byte being considered to be the most significant bit (MSB). The bit stream formed from these bytes will be the bit stream with the MSB (bit 7) of the first byte first, going down to bit 0 of the first byte, and then the MSB of the second byte and so on.

With this in mind, we can sketch an implementation of the CRC division. For the purposes of example, consider a poly with W=4 and the poly=10111. Then, the perform the division, we need to use a 4-bit register:



(Reminder: The augmented message is the message followed by W zero bits.)

To perform the division perform the following:

```
   Load the register with zero bits.
   Augment the message by appending W zero bits to the end of it.
   While (more message bits)
      Begin
      Shift the register left by one bit, reading the next bit of the
         augmented message into register bit position 0.
      If (a 1 bit popped out of the register during step 3)
         Register = Register XOR Poly.
      End
   The register now contains the remainder.
```

(Note: In practice, the IF condition can be tested by testing the top bit of R before performing the shift.)

We will call this algorithm "SIMPLE".

This might look a bit messy, but all we are really doing is "subtracting" various powers (i.e. shiftings) of the poly from the message until there is nothing left but the remainder. Study the manual examples of long division if you don't understand this.

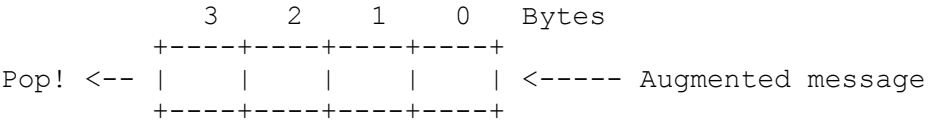
It should be clear that the above algorithm will work for any width W.

9. A Table-Driven Implementation

The SIMPLE algorithm above is a good starting point because it corresponds directly to the theory presented so far, and because it is so SIMPLE. However, because it operates at the bit level, it is rather awkward to code (even in C), and inefficient to execute (it has to loop once for each bit). To speed it up, we need to find a way to enable the algorithm to process the message in units larger than one bit. Candidate quantities are nibbles (4 bits), bytes (8 bits), words (16 bits) and longwords (32 bits) and higher if we can achieve it. Of these, 4 bits is best avoided because it does not correspond to a byte boundary. At the very least, any speedup should allow us to operate at byte boundaries, and in fact most of the table driven algorithms operate a byte at a time.

For the purposes of discussion, let us switch from a 4-bit poly to a 32-bit one. Our register looks much the same, except the boxes represent bytes instead of bits, and the Poly is 33 bits (one implicit

1 bit at the top and 32 "active" bits) (W=32).



The SIMPLE algorithm is still applicable. Let us examine what it does. Imagine that the SIMPLE algorithm is in full swing and consider the top 8 bits of the 32-bit register (byte 3) to have the values:

t7 t6 t5 t4 t3 t2 t1 t0

In the next iteration of SIMPLE, t7 will determine whether the Poly will be XORed into the entire register. If t7=1, this will happen, otherwise it will not. Suppose that the top 8 bits of the poly are g7 g6.. g0, then after the next iteration, the top byte will be:

```
        t6 t5 t4 t3 t2 t1 t0 ??
+ t7 * (g7 g6 g5 g4 g3 g2 g1 g0)    [Reminder: + is XOR]
```

The NEW top bit (that will control what happens in the next iteration) now has the value t6 + t7*g7. The important thing to notice here is that from an informational point of view, all the information required to calculate the NEW top bit was present in the top TWO bits of the original top byte. Similarly, the NEXT top bit can be calculated in advance SOLELY from the top THREE bits t7, t6, and t5. In fact, in general, the value of the top bit in the register in k iterations can be calculated from the top k bits of the register. Let us take this for granted for a moment.

Consider for a moment that we use the top 8 bits of the register to calculate the value of the top bit of the register during the next 8 iterations. Suppose that we drive the next 8 iterations using the calculated values (which we could perhaps store in a single byte register and shift out to pick off each bit). Then we note three things:

* The top byte of the register now doesn't matter. No matter how many times and at what offset the poly is XORed to the top 8 bits, they will all be shifted out the right hand side during the next 8 iterations anyway.

* The remaining bits will be shifted left one position and the rightmost byte of the register will be shifted in the next byte

AND

* While all this is going on, the register will be subjected to a series of XOR's in accordance with the bits of the pre-calculated control byte.

Now consider the effect of XORing in a constant value at various offsets to a register. For example:

```
0100010  Register
...0110  XOR this
..0110.  XOR this
0110...  XOR this
-------
0011000
-------
```

The point of this is that you can XOR constant values into a register to your heart's delight, and in the end, there will exist a value which when XORed in with the original register will have the same effect as all the other XORs.

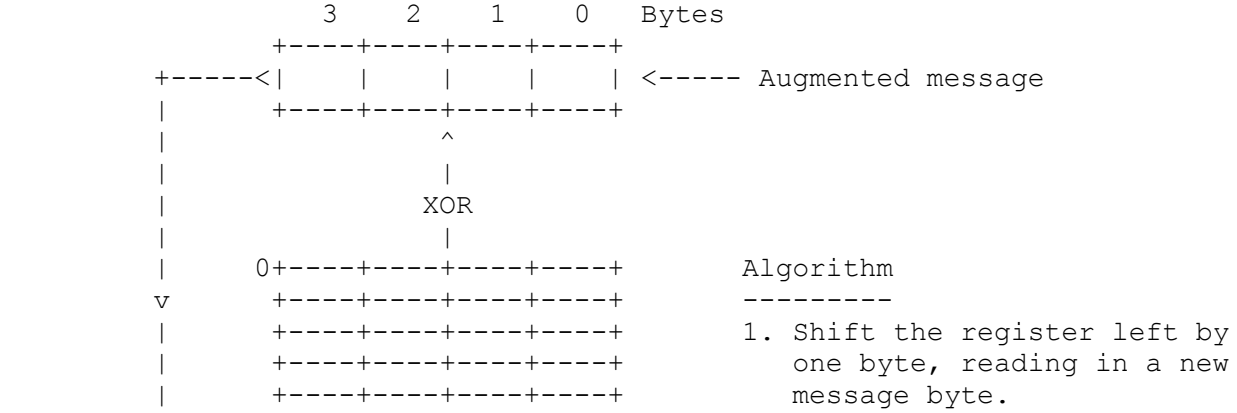
Perhaps you can see the solution now. Putting all the pieces together we have an algorithm that goes like this:

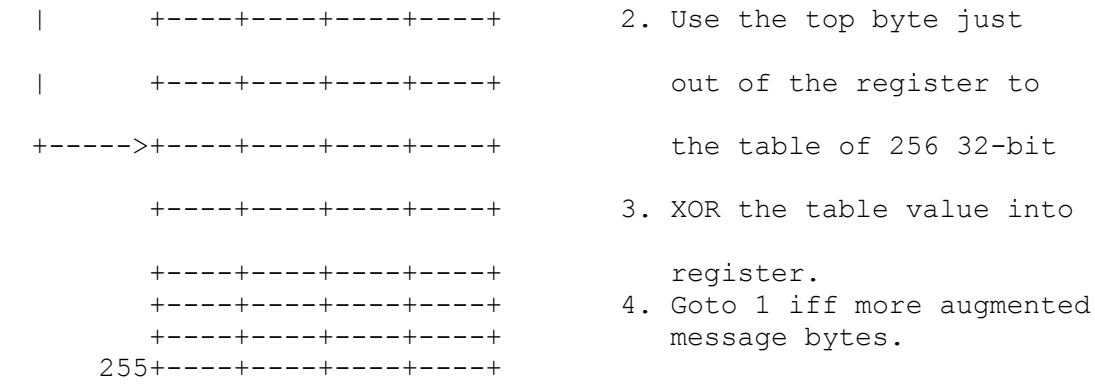
```
While (augmented message is not exhausted)
   Begin
   Examine the top byte of the register
   Calculate the control byte from the top byte of the register
   Sum all the Polys at various offsets that are to be XORed into
      the register in accordance with the control byte
   Shift the register left by one byte, reading a new message byte
      into the rightmost byte of the register
   XOR the summed polys to the register
   End
```

As it stands this is not much better than the SIMPLE algorithm. However, it turns out that most of the calculation can be precomputed and assembled into a table. As a result, the above algorithm can be reduced to:

```
While (augmented message is not exhaused)
   Begin
   Top = top_byte(Register);
   Register = (Register << 24) | next_augmessage_byte;
   Register = Register XOR precomputed_table[Top];
   End
```

There! If you understand this, you've grasped the main idea of table-driven CRC algorithms. The above is a very efficient algorithm requiring just a shift, and OR, an XOR, and a table lookup per byte. Graphically, it looks like this:





In C, the algorithm main loop looks like this:

```
r=0;
while (len--)
  {
   byte t = (r >> 24) & 0xFF;
   r = (r << 8) | *p++;
   r^=table[t];
  }
```

where len is the length of the augmented message in bytes, p points to the augmented message, r is the register, t is a temporary, and table is the computed table. This code can be made even more unreadable as follows:

```
r=0; while (len--) r = ((r << 8) | *p++) ^ t[(r >> 24) & 0xFF];
```

This is a very clean, efficient loop, although not a very obvious one to the casual observer not versed in CRC theory. We will call this the TABLE algorithm.

10. A Slightly Mangled Table-Driven Implementation

Despite the terse beauty of the line

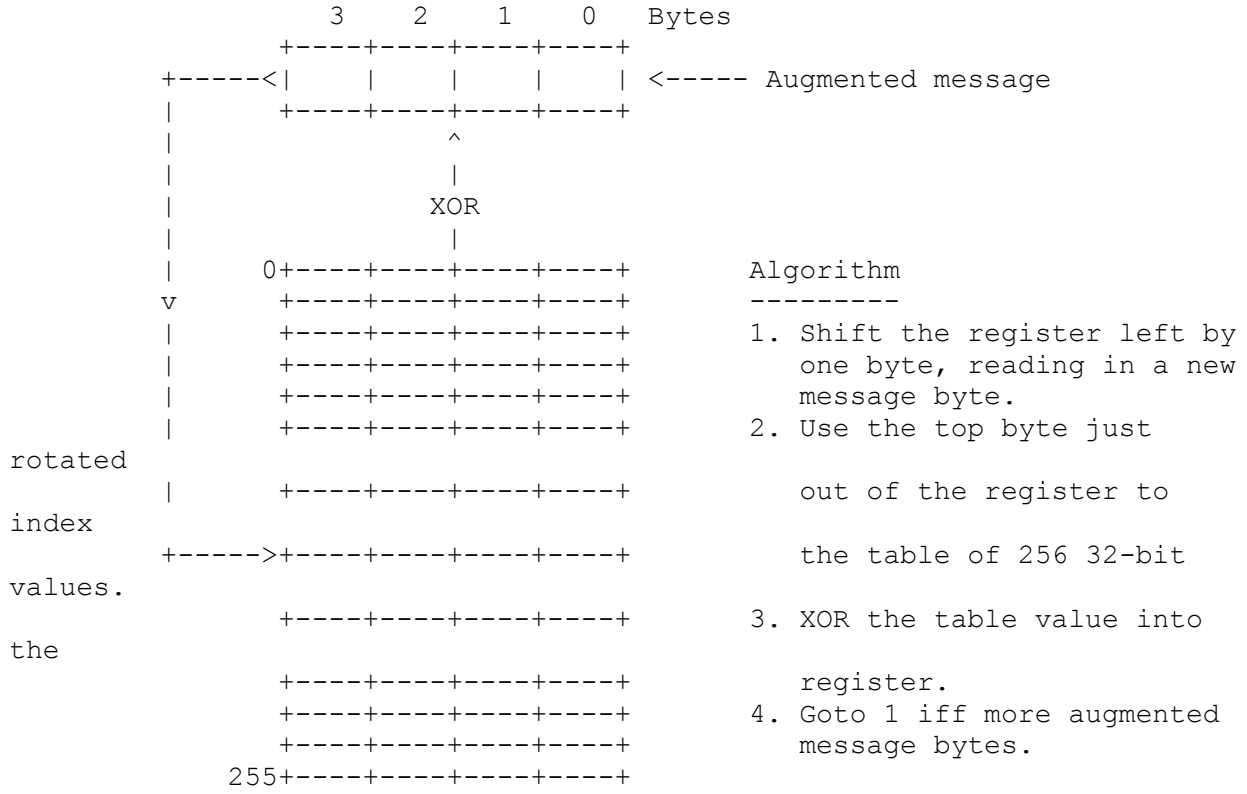
```
r=0; while (len--) r = ((r << 8) | *p++) ^ t[(r >> 24) & 0xFF];
```

those optimizing hackers couldn't leave it alone. The trouble, you see, is that this loop operates upon the AUGMENTED message and in order to use this code, you have to append W/8 zero bytes to the end of the message before pointing p at it. Depending on the run-time environment, this may or may not be a problem; if the block of data was handed to us by some other code, it could be a BIG problem. One alternative is simply to append the following line after the above loop, once for each zero byte:

```
for (i=0; i<W/4; i++) r = (r << 8) ^ t[(r >> 24) & 0xFF];
```

This looks like a sane enough solution to me. However, at the further expense of clarity (which, you must admit, is already a pretty scare commodity in this code) we can reorganize this small loop further so as to avoid the need to either augment the message with zero bytes, or to explicitly process zero bytes at the end as above. To explain the

optimization, we return to the processing diagram given earlier.



Now, note the following facts:

TAIL: The W/4 augmented zero bytes that appear at the end of the message will be pushed into the register from the right as all the other bytes are, but their values (0) will have no effect whatsoever on the register because 1) XORing with zero does not change the target byte, and 2) the four bytes are never propagated out the left side of the register where their zeroness might have some sort of influence. Thus, the sole function of the W/4 augmented zero bytes is to drive the calculation for another W/4 byte cycles so that the end of the REAL data passes all the way through the register.

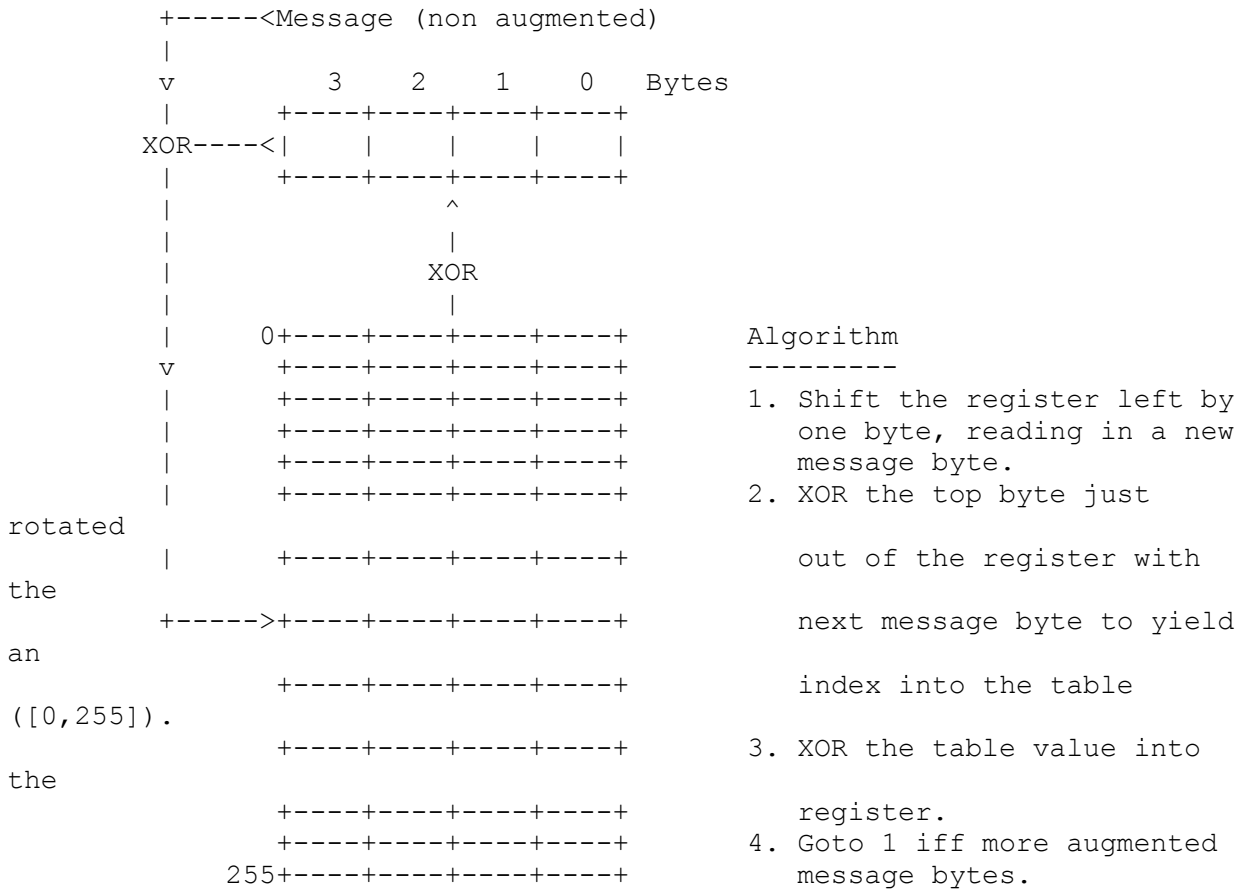
HEAD: If the initial value of the register is zero, the first four iterations of the loop will have the sole effect of shifting in the first four bytes of the message from the right. This is because the first 32 control bits are all zero and so nothing is XORed into the register. Even if the initial value is not zero, the first 4 byte iterations of the algorithm will have the sole effect of shifting the first 4 bytes of the message into the register and then XORing them with some constant value (that is a function of the initial value of the register).

These facts, combined with the XOR property

```
(A xor B) xor C = A xor (B xor C)
```

mean that message bytes need not actually travel through the W/4 bytes of the register. Instead, they can be XORed into the top byte just before it is used to index the lookup table. This leads to the

following modified version of the algorithm.



Note: The initial register value for this algorithm must be the initial value of the register for the previous algorithm fed through the table four times. Note: The table is such that if the previous algorithm used 0, the new algorithm will too.

This is an IDENTICAL algorithm and will yield IDENTICAL results. The C code looks something like this:

```
   r=0; while (len--) r = (r<<8) ^ t[(r >> 24) ^ *p++];
```

and THIS is the code that you are likely to find inside current table-driven CRC implementations. Some FF masks might have to be ANDed in here and there for portability's sake, but basically, the above loop is IT. We will call this the DIRECT TABLE ALGORITHM.

During the process of trying to understand all this stuff, I managed to derive the SIMPLE algorithm and the table-driven version derived from that. However, when I compared my code with the code found in real-implementations, I was totally bamboozled as to why the bytes were being XORed in at the wrong end of the register! It took quite a while before I figured out that theirs and my algorithms were actually the same. Part of why I am writing this document is that, while the link between division and my earlier table-driven code is vaguely apparent, any such link is fairly well erased when you start pumping bytes in at the "wrong end" of the register. It looks all wrong!

If you've got this far, you not only understand the theory, the practice, the optimized practice, but you also understand the real code you are likely to run into. Could get any more complicated? Yes it can.

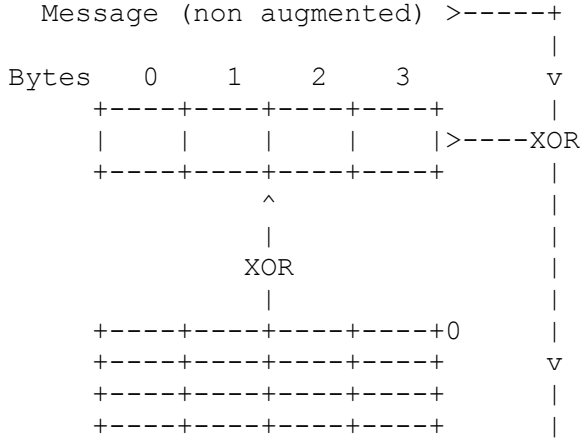
11. "Reflected" Table-Driven Implementations

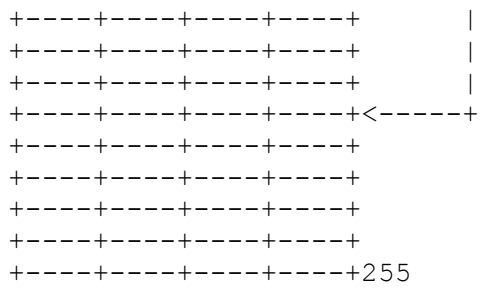
Despite the fact that the above code is probably optimized about as much as it could be, this did not stop some enterprising individuals from making things even more complicated. To understand how this happened, we have to enter the world of hardware.

DEFINITION: A value/register is reflected if it's bits are swapped around its centre. For example: 0101 is the 4-bit reflection of 1010. 0011 is the reflection of 1100.
0111-0101-1010-1111-0010-0101-1011-1100 is the reflection of 0011-1101-1010-0100-1111-0101-1010-1110.

Turns out that UARTs (those handy little chips that perform serial IO) are in the habit of transmitting each byte with the least significant bit (bit 0) first and the most significant bit (bit 7) last (i.e. reflected). An effect of this convention is that hardware engineers constructing hardware CRC calculators that operate at the bit level took to calculating CRCs of bytes streams with each of the bytes reflected within itself. The bytes are processed in the same order, but the bits in each byte are swapped; bit 0 is now bit 7, bit 1 is now bit 6, and so on. Now this wouldn't matter much if this convention was restricted to hardware land. However it seems that at some stage some of these CRC values were presented at the software level and someone had to write some code that would interoperate with the hardware CRC calculation.

In this situation, a normal sane software engineer would simply reflect each byte before processing it. However, it would seem that normal sane software engineers were thin on the ground when this early ground was being broken, because instead of reflecting the bytes, whoever was responsible held down the byte and reflected the world, leading to the following "reflected" algorithm which is identical to the previous one except that everything is reflected except the input bytes.





Notes:

* The table is identical to the one in the previous algorithm except that each entry has been reflected.

* The initial value of the register is the same as in the previous algorithm except that it has been reflected.

* The bytes of the message are processed in the same order as before (i.e. the message itself is not reflected).

* The message bytes themselves don't need to be explicitly reflected, because everything else has been!

At the end of execution, the register contains the reflection of the final CRC value (remainder). Actually, I'm being rather hard on whoever cooked this up because it seems that hardware implementations of the CRC algorithm used the reflected checksum value and so producing a reflected CRC was just right. In fact reflecting the world was probably a good engineering solution - if a confusing one.

We will call this the REFLECTED algorithm.

Whether or not it made sense at the time, the effect of having reflected algorithms kicking around the world's FTP sites is that about half the CRC implementations one runs into are reflected and the other half not. It's really terribly confusing. In particular, it would seem to me that the casual reader who runs into a reflected, table-driven implementation with the bytes "fed in the wrong end" would have Buckley's chance of ever connecting the code to the concept of binary mod 2 division.

It couldn't get any more confusing could it? Yes it could.

12. "Reversed" Polys

As if reflected implementations weren't enough, there is another concept kicking around which makes the situation bizaarly confusing. The concept is reversed Polys.

It turns out that the reflection of good polys tend to be good polys too! That is, if G=11101 is a good poly value, then 10111 will be as well. As a consequence, it seems that every time an organization (such as CCITT) standardizes on a particularly good poly ("polynomial"), those in the real world can't leave the poly's reflection alone either. They just HAVE to use it. As a result, the set of "standard" poly's has a corresponding set of reflections, which are also in use.

To avoid confusion, we will call these the "reversed" polys.

```
   X25   standard: 1-0001-0000-0010-0001
   X25   reversed: 1-0000-1000-0001-0001

   CRC16 standard: 1-1000-0000-0000-0101
   CRC16 reversed: 1-0100-0000-0000-0011
```

Note that here it is the entire poly that is being reflected/reversed, not just the bottom W bits. This is an important distinction. In the reflected algorithm described in the previous section, the poly used in the reflected algorithm was actually identical to that used in the non-reflected algorithm; all that had happened is that the bytes had effectively been reflected. As such, all the 16-bit/32-bit numbers in the algorithm had to be reflected. In contrast, the ENTIRE poly includes the implicit one bit at the top, and so reversing a poly is not the same as reflecting its bottom 16 or 32 bits.

The upshot of all this is that a reflected algorithm is not equivalent to the original algorithm with the poly reflected. Actually, this is probably less confusing than if they were duals.

If all this seems a bit unclear, don't worry, because we're going to sort it all out "real soon now". Just one more section to go before that.

13. Initial and Final Values

In addition to the complexity already seen, CRC algorithms differ from each other in two other regards:

   * The initial value of the register.

   * The value to be XORed with the final register value.

For example, the "CRC32" algorithm initializes its register to FFFFFFFF and XORs the final register value with FFFFFFFF.

Most CRC algorithms initialize their register to zero. However, some initialize it to a non-zero value. In theory (i.e. with no assumptions about the message), the initial value has no affect on the strength of the CRC algorithm, the initial value merely providing a fixed starting point from which the register value can progress. However, in practice, some messages are more likely than others, and it is wise to initialize the CRC algorithm register to a value that does not have "blind spots" that are likely to occur in practice. By "blind spot" is meant a sequence of message bytes that do not result in the register changing its value. In particular, any CRC algorithm that initializes its register to zero will have a blind spot of zero when it starts up and will be unable to "count" a leading run of zero bytes. As a leading run of zero bytes is quite common in real messages, it is wise to initialize the algorithm register to a non-zero value.

14. Defining Algorithms Absolutely

At this point we have covered all the different aspects of

table-driven CRC algorithms. As there are so many variations on these algorithms, it is worth trying to establish a nomenclature for them. This section attempts to do that.

We have seen that CRC algorithms vary in:

* Width of the poly (polynomial).
* Value of the poly.
* Initial value for the register.
* Whether the bits of each byte are reflected before being processed.
* Whether the algorithm feeds input bytes through the register or xors them with a byte from one end and then straight into the table.
* Whether the final register value should be reversed (as in reflected versions).
* Value to XOR with the final register value.

In order to be able to talk about particular CRC algorithms, we need to able to define them more precisely than this. For this reason, the next section attempts to provide a well-defined parameterized model for CRC algorithms. To refer to a particular algorithm, we need then simply specify the algorithm in terms of parameters to the model.

15. A Parameterized Model For CRC Algorithms

In this section we define a precise parameterized model CRC algorithm which, for want of a better name, we will call the "Rocksoft^tm Model CRC Algorithm" (and why not? Rocksoft^tm could do with some free advertising :-).

The most important aspect of the model algorithm is that it focusses exclusively on functionality, ignoring all implementation details. The aim of the exercise is to construct a way of referring precisely to particular CRC algorithms, regardless of how confusingly they are implemented. To this end, the model must be as simple and precise as possible, with as little confusion as possible.

The Rocksoft^tm Model CRC Algorithm is based essentially on the DIRECT TABLE ALGORITHM specified earlier. However, the algorithm has to be further parameterized to enable it to behave in the same way as some of the messier algorithms out in the real world.

To enable the algorithm to behave like reflected algorithms, we provide a boolean option to reflect the input bytes, and a boolean option to specify whether to reflect the output checksum value. By framing reflection as an input/output transformation, we avoid the confusion of having to mentally map the parameters of reflected and non-reflected algorithms.

An extra parameter allows the algorithm's register to be initialized to a particular value. A further parameter is XORed with the final value before it is returned.

By putting all these pieces together we end up with the parameters of the algorithm:

NAME: This is a name given to the algorithm. A string value.

WIDTH: This is the width of the algorithm expressed in bits. This is one less than the width of the Poly.

POLY: This parameter is the poly. This is a binary value that should be specified as a hexadecimal number. The top bit of the poly should be omitted. For example, if the poly is 10110, you should specify 06. An important aspect of this parameter is that it represents the unreflected poly; the bottom bit of this parameter is always the LSB of the divisor during the division regardless of whether the algorithm being modelled is reflected.

INIT: This parameter specifies the initial value of the register when the algorithm starts. This is the value that is to be assigned to the register in the direct table algorithm. In the table algorithm, we may think of the register always commencing with the value zero, and this value being XORed into the register after the N'th bit iteration. This parameter should be specified as a hexadecimal number.

REFIN: This is a boolean parameter. If it is FALSE, input bytes are processed with bit 7 being treated as the most significant bit (MSB) and bit 0 being treated as the least significant bit. If this parameter is FALSE, each byte is reflected before being processed.

REFOUT: This is a boolean parameter. If it is set to FALSE, the final value in the register is fed into the XOROUT stage directly, otherwise, if this parameter is TRUE, the final register value is reflected first.

XOROUT: This is an W-bit value that should be specified as a hexadecimal number. It is XORed to the final register value (after the REFOUT) stage before the value is returned as the official checksum.

CHECK: This field is not strictly part of the definition, and, in the event of an inconsistency between this field and the other field, the other fields take precedence. This field is a check value that can be used as a weak validator of implementations of the algorithm. The field contains the checksum obtained when the ASCII string "123456789" is fed through the specified algorithm (i.e. 313233... (hexadecimal)).

With these parameters defined, the model can now be used to specify a particular CRC algorithm exactly. Here is an example specification for a popular form of the CRC-16 algorithm.

```
Name   : "CRC-16"
Width  : 16
Poly   : 8005
Init   : 0000
RefIn  : True
RefOut : True
XorOut : 0000
Check  : BB3D
```

16. A Catalog of Parameter Sets for Standards

At this point, I would like to give a list of the specifications for commonly used CRC algorithms. However, most of the algorithms that I have come into contact with so far are specified in such a vague way that this has not been possible. What I can provide is a list of polys for various CRC standards I have heard of:

```
X25   standard : 1021       [CRC-CCITT, ADCCP, SDLC/HDLC]
X25   reversed : 0811

CRC16 standard : 8005
CRC16 reversed : 4003       [LHA]

CRC32          : 04C11DB7   [PKZIP, AUTODIN II, Ethernet, FDDI]
```

I would be interested in hearing from anyone out there who can tie down the complete set of model parameters for any of these standards.

However, a program that was kicking around seemed to imply the following specifications. Can anyone confirm or deny them (or provide the check values (which I couldn't be bothered coding up and calculating)).

```
Name   : "CRC-16/CITT"
Width  : 16
Poly   : 1021
Init   : FFFF
RefIn  : False
RefOut : False
XorOut : 0000
Check  : ?


Name   : "XMODEM"
Width  : 16
Poly   : 8408
Init   : 0000
RefIn  : True
RefOut : True
XorOut : 0000
Check  : ?


Name   : "ARC"
Width  : 16
Poly   : 8005
Init   : 0000
RefIn  : True
RefOut : True
XorOut : 0000
Check  : ?
```

Here is the specification for the CRC-32 algorithm which is reportedly used in PKZip, AUTODIN II, Ethernet, and FDDI.

```
Name   : "CRC-32"
Width  : 32
Poly   : 04C11DB7
Init   : FFFFFFFF
```

```
   RefIn  : True
   RefOut : True
   XorOut : FFFFFFFF
   Check  : CBF43926
```

17. An Implementation of the Model Algorithm

Here is an implementation of the model algorithm in the C programming language. The implementation consists of a header file (.h) and an implementation file (.c). If you're reading this document in a sequential scroller, you can skip this code by searching for the string "Roll Your Own".

To ensure that the following code is working, configure it for the CRC-16 and CRC-32 algorithms given above and ensure that they produce the specified "check" checksum when fed the test string "123456789" (see earlier).

```
/*****************************************************************
******/
/*                              Start of crcmodel.h
*/
/*****************************************************************
******/
/*
*/
/* Author : Ross Williams (ross@guest.adelaide.edu.au.).
*/
/* Date   : 3 June 1993.
*/
/* Status : Public domain.
*/
/*
*/
/* Description : This is the header (.h) file for the reference
*/
/* implementation of the Rocksoft^tm Model CRC Algorithm. For more
*/
/* information on the Rocksoft^tm Model CRC Algorithm, see the document
*/
/* titled "A Painless Guide to CRC Error Detection Algorithms" by Ross
*/
/* Williams (ross@guest.adelaide.edu.au.). This document is likely to be
in   */
/* "ftp.adelaide.edu.au/pub/rocksoft".
*/
/*
*/
/* Note: Rocksoft is a trademark of Rocksoft Pty Ltd, Adelaide,
Australia.    */
/*
*/
/*****************************************************************
******/
/*
*/
```

```
/* How to Use This Package
*/
/* -----------------------
*/
/* Step 1: Declare a variable of type cm_t. Declare another variable
*/
/*         (p_cm say) of type p_cm_t and initialize it to point to the
first  */
/*         variable (e.g. p_cm_t p_cm = &cm_t).
*/
/*
*/
/* Step 2: Assign values to the parameter fields of the structure.
*/
/*         If you don't know what to assign, see the document cited
earlier.  */
/*         For example:
*/
/*            p_cm->cm_width = 16;
*/
/*            p_cm->cm_poly  = 0x8005L;
*/
/*            p_cm->cm_init  = 0L;
*/
/*            p_cm->cm_refin = TRUE;
*/
/*            p_cm->cm_refot = TRUE;
*/
/*            p_cm->cm_xorot = 0L;
*/
/*         Note: Poly is specified without its top bit (18005 becomes
8005).  */
/*         Note: Width is one bit less than the raw poly width.
*/
/*
*/
/* Step 3: Initialize the instance with a call cm_ini(p_cm);
*/
/*
*/
/* Step 4: Process zero or more message bytes by placing zero or more
*/
/*         successive calls to cm_nxt. Example: cm_nxt(p_cm,ch);
*/
/*
*/
/* Step 5: Extract the CRC value at any time by calling crc =
cm_crc(p_cm);   */
/*         If the CRC is a 16-bit value, it will be in the bottom 16
bits.    */
/*
*/
/**************************************************************************
******/
/*
*/
/* Design Notes
*/
```

```
/* ------------
*/
/* PORTABILITY: This package has been coded very conservatively so that
*/
/* it will run on as many machines as possible. For example, all external
*/
/* identifiers have been restricted to 6 characters and all internal ones
to  */
/* 8 characters. The prefix cm (for Crc Model) is used as an attempt to
avoid */
/* namespace collisions. This package is endian independent.
*/
/*
*/
/* EFFICIENCY: This package (and its interface) is not designed for
*/
/* speed. The purpose of this package is to act as a well-defined
reference   */
/* model for the specification of CRC algorithms. If you want speed, cook
up  */
/* a specific table-driven implementation as described in the document
cited  */
/* above. This package is designed for validation only; if you have found
or  */
/* implemented a CRC algorithm and wish to describe it as a set of
parameters */
/* to the Rocksoft^tm Model CRC Algorithm, your CRC algorithm
implementation  */
/* should behave identically to this package under those parameters.
*/
/*
*/
/************************************************************************
******/

/* The following #ifndef encloses this entire */
/* header file, rendering it indempotent.     */
#ifndef CM_DONE
#define CM_DONE

/************************************************************************
******/

/* The following definitions are extracted from my style header file
which    */
/* would be cumbersome to distribute with this package. The DONE_STYLE is
the */
/* idempotence symbol used in my style header file.
*/

#ifndef DONE_STYLE

typedef unsigned long   ulong;
typedef unsigned        bool;
typedef unsigned char * p_ubyte_;

#ifndef TRUE
#define FALSE 0
```

```
#define TRUE  1
#endif

/* Change to the second definition if you don't have prototypes. */
#define P_(A) A
/* #define P_(A) () */

/* Uncomment this definition if you don't have void. */
/* typedef int void; */

#endif

/*****************************************************************
******/

/* CRC Model Abstract Type */
/* ----------------------- */
/* The following type stores the context of an executing instance of the
*/
/* model algorithm. Most of the fields are model parameters which must be
*/
/* set before the first initializing call to cm_ini.
*/
typedef struct
  {
   int   cm_width;   /* Parameter: Width in bits [8,32].       */
   ulong cm_poly;    /* Parameter: The algorithm's polynomial. */
   ulong cm_init;    /* Parameter: Initial register value.     */
   bool  cm_refin;   /* Parameter: Reflect input bytes?        */
   bool  cm_refot;   /* Parameter: Reflect output CRC?         */
   ulong cm_xorot;   /* Parameter: XOR this to output CRC.     */

   ulong cm_reg;     /* Context: Context during execution.     */
  } cm_t;
typedef cm_t *p_cm_t;

/*****************************************************************
******/

/* Functions That Implement The Model */
/* ---------------------------------- */
/* The following functions animate the cm_t abstraction. */

void cm_ini P_((p_cm_t p_cm));
/* Initializes the argument CRC model instance.          */
/* All parameter fields must be set before calling this. */

void cm_nxt P_((p_cm_t p_cm,int ch));
/* Processes a single message byte [0,255]. */

void cm_blk P_((p_cm_t p_cm,p_ubyte_ blk_adr,ulong blk_len));
/* Processes a block of message bytes. */

ulong cm_crc P_((p_cm_t p_cm));
/* Returns the CRC value for the message bytes processed so far. */

/*****************************************************************
******/
```

```
/* Functions For Table Calculation */
/* ------------------------------ */
/* The following function can be used to calculate a CRC lookup table.
*/
/* It can also be used at run-time to create or check static tables.
*/

ulong cm_tab P_((p_cm_t p_cm,int index));
/* Returns the i'th entry for the lookup table for the specified
algorithm.   */
/* The function examines the fields cm_width, cm_poly, cm_refin, and the
*/
/* argument table index in the range [0,255] and returns the table entry
in   */
/* the bottom cm_width bytes of the return value.
*/

/*********************************************************************
******/

/* End of the header file idempotence #ifndef */
#endif

/*********************************************************************
******/
/*                              End of crcmodel.h
*/
/*********************************************************************
******/


/*********************************************************************
******/
/*                              Start of crcmodel.c
*/
/*********************************************************************
******/
/*
*/
/* Author : Ross Williams (ross@guest.adelaide.edu.au.).
*/
/* Date   : 3 June 1993.
*/
/* Status : Public domain.
*/
/*
*/
/* Description : This is the implementation (.c) file for the reference
*/
/* implementation of the Rocksoft^tm Model CRC Algorithm. For more
*/
/* information on the Rocksoft^tm Model CRC Algorithm, see the document
*/
/* titled "A Painless Guide to CRC Error Detection Algorithms" by Ross
*/
/* Williams (ross@guest.adelaide.edu.au.). This document is likely to be
in   */
```

```
/* "ftp.adelaide.edu.au/pub/rocksoft".
*/
/*
*/
/* Note: Rocksoft is a trademark of Rocksoft Pty Ltd, Adelaide,
Australia.    */
/*
*/
/*****************************************************************
******/
/*
*/
/* Implementation Notes
*/
/* --------------------
*/
/* To avoid inconsistencies, the specification of each function is not
echoed */
/* here. See the header file for a description of these functions.
*/
/* This package is light on checking because I want to keep it short and
*/
/* simple and portable (i.e. it would be too messy to distribute my
entire    */
/* C culture (e.g. assertions package) with this package.
*/
/*
*/
/*****************************************************************
******/

#include "crcmodel.h"

/*****************************************************************
******/

/* The following definitions make the code more readable. */

#define BITMASK(X) (1L << (X))
#define MASK32 0xFFFFFFFFL
#define LOCAL static

/*****************************************************************
******/

LOCAL ulong reflect P_((ulong v,int b));
LOCAL ulong reflect (v,b)
/* Returns the value v with the bottom b [0,32] bits reflected. */
/* Example: reflect(0x3e23L,3) == 0x3e26                         */
ulong v;
int   b;
{
 int   i;
 ulong t = v;
 for (i=0; i<b; i++)
   {
    if (t & 1L)
       v|=  BITMASK((b-1)-i);
```

```
    else
       v&= ~BITMASK((b-1)-i);
    t>>=1;
   }
 return v;
}

/*****************************************************************************
******/

LOCAL ulong widmask P_((p_cm_t));
LOCAL ulong widmask (p_cm)
/* Returns a longword whose value is (2^p_cm->cm_width)-1.     */
/* The trick is to do this portably (e.g. without doing <<32). */
p_cm_t p_cm;
{
 return (((1L<<(p_cm->cm_width-1))-1L)<<1)|1L;
}

/*****************************************************************************
******/

void cm_ini (p_cm)
p_cm_t p_cm;
{
 p_cm->cm_reg = p_cm->cm_init;
}

/*****************************************************************************
******/

void cm_nxt (p_cm,ch)
p_cm_t p_cm;
int    ch;
{
 int   i;
 ulong uch  = (ulong) ch;
 ulong topbit = BITMASK(p_cm->cm_width-1);

 if (p_cm->cm_refin) uch = reflect(uch,8);
 p_cm->cm_reg ^= (uch << (p_cm->cm_width-8));
 for (i=0; i<8; i++)
   {
    if (p_cm->cm_reg & topbit)
       p_cm->cm_reg = (p_cm->cm_reg << 1) ^ p_cm->cm_poly;
    else
       p_cm->cm_reg <<= 1;
    p_cm->cm_reg &= widmask(p_cm);
   }
}

/*****************************************************************************
******/

void cm_blk (p_cm,blk_adr,blk_len)
p_cm_t   p_cm;
p_ubyte_ blk_adr;
ulong    blk_len;
```

```
{
 while (blk_len--) cm_nxt(p_cm,*blk_adr++);
}

/**********************************************************************
******/

ulong cm_crc (p_cm)
p_cm_t p_cm;
{
 if (p_cm->cm_refot)
    return p_cm->cm_xorot ^ reflect(p_cm->cm_reg,p_cm->cm_width);
 else
    return p_cm->cm_xorot ^ p_cm->cm_reg;
}

/**********************************************************************
******/

ulong cm_tab (p_cm,index)
p_cm_t p_cm;
int    index;
{
 int   i;
 ulong r;
 ulong topbit = BITMASK(p_cm->cm_width-1);
 ulong inbyte = (ulong) index;

 if (p_cm->cm_refin) inbyte = reflect(inbyte,8);
 r = inbyte << (p_cm->cm_width-8);
 for (i=0; i<8; i++)
    if (r & topbit)
       r = (r << 1) ^ p_cm->cm_poly;
    else
       r<<=1;
 if (p_cm->cm_refin) r = reflect(r,p_cm->cm_width);
 return r & widmask(p_cm);
}

/**********************************************************************
******/
/*                              End of crcmodel.c
*/
/**********************************************************************
******/
```

18. Roll Your Own Table-Driven Implementation

Despite all the fuss I've made about understanding and defining CRC algorithms, the mechanics of their high-speed implementation remains trivial. There are really only two forms: normal and reflected. Normal shifts to the left and covers the case of algorithms with Refin=FALSE and Refot=FALSE. Reflected shifts to the right and covers algorithms with both those parameters true. (If you want one parameter true and the other false, you'll have to figure it out for yourself!) The polynomial is embedded in the lookup table (to be discussed). The other parameters, Init and XorOt can be coded as macros. Here is the

32-bit normal form (the 16-bit form is similar).

```
   unsigned long crc_normal ();
   unsigned long crc_normal (blk_adr,blk_len)
   unsigned char *blk_adr;
   unsigned long  blk_len;
   {
    unsigned long crc = INIT;
    while (blk_len--)
       crc = crctable[((crc>>24) ^ *blk_adr++) & 0xFFL] ^ (crc << 8);
    return crc ^ XOROT;
   }
```

Here is the reflected form:

```
   unsigned long crc_reflected ();
   unsigned long crc_reflected (blk_adr,blk_len)
   unsigned char *blk_adr;
   unsigned long  blk_len;
   {
    unsigned long crc = INIT_REFLECTED;
    while (blk_len--)
       crc = crctable[(crc ^ *blk_adr++) & 0xFFL] ^ (crc >> 8));
    return crc ^ XOROT;
   }
```

Note: I have carefully checked the above two code fragments, but I haven't actually compiled or tested them. This shouldn't matter to you, as, no matter WHAT you code, you will always be able to tell if you have got it right by running whatever you have created against the reference model given earlier. The code fragments above are really just a rough guide. The reference model is the definitive guide.

Note: If you don't care much about speed, just use the reference model code!

19. Generating A Lookup Table

The only component missing from the normal and reversed code fragments in the previous section is the lookup table. The lookup table can be computed at run time using the cm_tab function of the model package given earlier, or can be pre-computed and inserted into the C program. In either case, it should be noted that the lookup table depends only on the POLY and RefIn (and RefOt) parameters. Basically, the polynomial determines the table, but you can generate a reflected table too if you want to use the reflected form above.

The following program generates any desired 16-bit or 32-bit lookup table. Skip to the word "Summary" if you want to skip over this code.

```
/**********************************************************************
******/
/*                              Start of crctable.c
*/
```

```
/*****************************************************************************/
/*
*/
/* Author  : Ross Williams (ross@guest.adelaide.edu.au.).
*/
/* Date    : 3 June 1993.
*/
/* Version : 1.0.
*/
/* Status  : Public domain.
*/
/*
*/
/* Description : This program writes a CRC lookup table (suitable for
*/
/* inclusion in a C program) to a designated output file. The program can
be  */
/* statically configured to produce any table covered by the Rocksoft^tm
*/
/* Model CRC Algorithm. For more information on the Rocksoft^tm Model CRC
*/
/* Algorithm, see the document titled "A Painless Guide to CRC Error
*/
/* Detection Algorithms" by Ross Williams (ross@guest.adelaide.edu.au.).
This */
/* document is likely to be in "ftp.adelaide.edu.au/pub/rocksoft".
*/
/*
*/
/* Note: Rocksoft is a trademark of Rocksoft Pty Ltd, Adelaide,
Australia.    */
/*
*/
/*****************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include "crcmodel.h"

/*****************************************************************************/

/* TABLE PARAMETERS
*/
/* ================
*/
/* The following parameters entirely determine the table to be generated.
You */
/* should need to modify only the definitions in this section before
running  */
/* this program.
*/
/*
*/
/*    TB_FILE  is the name of the output file.
*/
```

```
/*    TB_WIDTH is the table width in bytes (either 2 or 4).
*/
/*    TB_POLY  is the "polynomial", which must be TB_WIDTH bytes wide.
*/
/*    TB_REVER indicates whether the table is to be reversed (reflected).
*/
/*
*/
/* Example:
*/
/*
*/
/*    #define TB_FILE   "crctable.out"
*/
/*    #define TB_WIDTH  2
*/
/*    #define TB_POLY   0x8005L
*/
/*    #define TB_REVER  TRUE
*/

#define TB_FILE   "crctable.out"
#define TB_WIDTH  4
#define TB_POLY   0x04C11DB7L
#define TB_REVER  TRUE

/************************************************************************
******/

/* Miscellaneous definitions. */

#define LOCAL static
FILE *outfile;
#define WR(X) fprintf(outfile,(X))
#define WP(X,Y) fprintf(outfile,(X),(Y))

/************************************************************************
******/

LOCAL void chk_err P_((char *));
LOCAL void chk_err (mess)
/* If mess is non-empty, write it out and abort. Otherwise, check the
error   */
/* status of outfile and abort if an error has occurred.
*/
char *mess;
{
 if (mess[0] != 0   ) {printf("%s\n",mess); exit(EXIT_FAILURE);}
 if (ferror(outfile)) {perror("chk_err");   exit(EXIT_FAILURE);}
}

/************************************************************************
******/

LOCAL void chkparam P_((void));
LOCAL void chkparam ()
{
 if ((TB_WIDTH != 2) && (TB_WIDTH != 4))
```

```
    chk_err("chkparam: Width parameter is illegal.");
 if ((TB_WIDTH == 2) && (TB_POLY & 0xFFFF0000L))
    chk_err("chkparam: Poly parameter is too wide.");
 if ((TB_REVER != FALSE) && (TB_REVER != TRUE))
    chk_err("chkparam: Reverse parameter is not boolean.");
}

/*****************************************************************
******/

LOCAL void gentable P_((void));
LOCAL void gentable ()
{

WR("/*****************************************************************/\n
");
 WR("/*
*/\n");
 WR("/* CRC LOOKUP TABLE
*/\n");
 WR("/* ================
*/\n");
 WR("/* The following CRC lookup table was generated automagically
*/\n");
 WR("/* by the Rocksoft^tm Model CRC Algorithm Table Generation
*/\n");
 WR("/* Program V1.0 using the following model parameters:
*/\n");
 WR("/*
*/\n");
 WP("/*    Width   : %1lu bytes.
*/\n",
    (ulong) TB_WIDTH);
 if (TB_WIDTH == 2)
 WP("/*    Poly    : 0x%04lX
*/\n",
    (ulong) TB_POLY);
 else
 WP("/*    Poly    : 0x%08lXL                                          */\n",
    (ulong) TB_POLY);
 if (TB_REVER)
 WR("/*    Reverse : TRUE.
*/\n");
 else
 WR("/*    Reverse : FALSE.
*/\n");
 WR("/*
*/\n");
 WR("/* For more information on the Rocksoft^tm Model CRC Algorithm,
*/\n");
 WR("/* see the document titled \"A Painless Guide to CRC Error
*/\n");
 WR("/* Detection Algorithms\" by Ross Williams
*/\n");
 WR("/* (ross@guest.adelaide.edu.au.). This document is likely to be
*/\n");
 WR("/* in the FTP archive \"ftp.adelaide.edu.au/pub/rocksoft\".
*/\n");
```

```
 WR("/*
*/\n");

WR("/*********************************************************************/\n
");
 WR("\n");
 switch (TB_WIDTH)
   {
    case 2: WR("unsigned short crctable[256] =\n{\n"); break;
    case 4: WR("unsigned long  crctable[256] =\n{\n"); break;
    default: chk_err("gentable: TB_WIDTH is invalid.");
   }
 chk_err("");

 {
  int i;
  cm_t cm;
  char *form    = (TB_WIDTH==2) ? "0x%04lX" : "0x%08lXL";
  int   perline = (TB_WIDTH==2) ? 8 : 4;

  cm.cm_width = TB_WIDTH*8;
  cm.cm_poly  = TB_POLY;
  cm.cm_refin = TB_REVER;

  for (i=0; i<256; i++)
    {
     WR(" ");
     WP(form,(ulong) cm_tab(&cm,i));
     if (i != 255) WR(",");
     if (((i+1) % perline) == 0) WR("\n");
     chk_err("");
    }

 WR("};\n");
 WR("\n");

WR("/*********************************************************************/\n
");
 WR("/*                    End of CRC Lookup Table
*/\n");

WR("/*********************************************************************/\n
");
 WR("");
 chk_err("");
}
}

/*********************************************************************
******/

main ()
{
 printf("\n");
 printf("Rocksoft^tm Model CRC Algorithm Table Generation Program
V1.0\n");
 printf("----------------------------------------------------------------
\n");
```

```
 printf("Output file is \"%s\".\n",TB_FILE);
 chkparam();
 outfile = fopen(TB_FILE,"w"); chk_err("");
 gentable();
 if (fclose(outfile) != 0)
    chk_err("main: Couldn't close output file.");
 printf("\nSUCCESS: The table has been successfully written.\n");
}

/**********************************************************************
******/
/*                              End of crctable.c
*/
/**********************************************************************
******/
```

20. Summary

This document has provided a detailed explanation of CRC algorithms explaining their theory and stepping through increasingly sophisticated implementations ranging from simple bit shifting through to byte-at-a-time table-driven implementations. The various implementations of different CRC algorithms that make them confusing to deal with have been explained. A parameterized model algorithm has been described that can be used to precisely define a particular CRC algorithm, and a reference implementation provided. Finally, a program to generate CRC tables has been provided.

21. Corrections

If you think that any part of this document is unclear or incorrect, or have any other information, or suggestions on how this document could be improved, please context the author. In particular, I would like to hear from anyone who can provide Rocksoft^tm Model CRC Algorithm parameters for standard algorithms out there.

A. Glossary

CHECKSUM - A number that has been calculated as a function of some message. The literal interpretation of this word "Check-Sum" indicates that the function should involve simply adding up the bytes in the message. Perhaps this was what early checksums were. Today, however, although more sophisticated formulae are used, the term "checksum" is still used.

CRC - This stands for "Cyclic Redundancy Code". Whereas the term "checksum" seems to be used to refer to any non-cryptographic checking information unit, the term "CRC" seems to be reserved only for algorithms that are based on the "polynomial" division idea.

G - This symbol is used in this document to represent the Poly.

MESSAGE - The input data being checksummed. This is usually structured as a sequence of bytes. Whether the top bit or the bottom bit of each byte is treated as the most significant or least significant is a parameter of CRC algorithms.

POLY - This is my friendly term for the polynomial of a CRC.

POLYNOMIAL - The "polynomial" of a CRC algorithm is simply the divisor in the division implementing the CRC algorithm.

REFLECT - A binary number is reflected by swapping all of its bits around the central point. For example, 1101 is the reflection of 1011.

ROCKSOFT^TM MODEL CRC ALGORITHM - A parameterized algorithm whose purpose is to act as a solid reference for describing CRC algorithms. Typically CRC algorithms are specified by quoting a polynomial. However, in order to construct a precise implementation, one also needs to know initialization values and so on.

WIDTH - The width of a CRC algorithm is the width of its polynomical minus one. For example, if the polynomial is 11010, the width would be 4 bits. The width is usually set to be a multiple of 8 bits.

B. References

[Griffiths87] Griffiths, G., Carlyle Stones, G., "The Tea-Leaf Reader Algorithm: An Efficient Implementation of CRC-16 and CRC-32", Communications of the ACM, 30(7), pp.617-620. Comment: This paper describes a high-speed table-driven implementation of CRC algorithms. The technique seems to be a touch messy, and is superceded by the Sarwete algorithm.

[Knuth81] Knuth, D.E., "The Art of Computer Programming", Volume 2: Seminumerical Algorithms, Section 4.6.

[Nelson 91] Nelson, M., "The Data Compression Book", M&T Books, (501 Galveston Drive, Redwood City, CA 94063), 1991, ISBN: 1-55851-214-4. Comment: If you want to see a real implementation of a real 32-bit checksum algorithm, look on pages 440, and 446-448.

[Sarwate88] Sarwate, D.V., "Computation of Cyclic Redundancy Checks via Table Look-Up", Communications of the ACM, 31(8), pp.1008-1013. Comment: This paper describes a high-speed table-driven implementation for CRC algorithms that is superior to the tea-leaf algorithm. Although this paper describes the technique used by most modern CRC implementations, I found the appendix of this paper (where all the good stuff is) difficult to understand.

[Tanenbaum81] Tanenbaum, A.S., "Computer Networks", Prentice Hall, 1981, ISBN: 0-13-164699-0. Comment: Section 3.5.3 on pages 128 to 132 provides a very clear description of CRC codes. However, it does not describe table-driven implementation techniques.

C. References I Have Detected But Haven't Yet Sighted

Boudreau, Steen, "Cyclic Redundancy Checking by Program," AFIPS Proceedings, Vol. 39, 1971.

Davies, Barber, "Computer Networks and Their Protocols," J. Wiley & Sons, 1979.

Higginson, Kirstein, "On the Computation of Cyclic Redundancy Checks by Program," The Computer Journal (British), Vol. 16, No. 1, Feb 1973.

McNamara, J. E., "Technical Aspects of Data Communication," 2nd Edition, Digital Press, Bedford, Massachusetts, 1982.

Marton and Frambs, "A Cyclic Redundancy Checking (CRC) Algorithm," Honeywell Computer Journal, Vol. 5, No. 3, 1971.

Nelson M., "File verification using CRC", Dr Dobbs Journal, May 1992, pp.64-67.

Ramabadran T.V., Gaitonde S.S., "A tutorial on CRC computations", IEEE Micro, Aug 1988.

Schwaderer W.D., "CRC Calculation", April 85 PC Tech Journal, pp.118-133.

Ward R.K, Tabandeh M., "Error Correction and Detection, A Geometric Approach" The Computer Journal, Vol. 27, No. 3, 1984, pp.246-253.

Wecker, S., "A Table-Lookup Algorithm for Software Computation of Cyclic Redundancy Check (CRC)," Digital Equipment Corporation memorandum, 1974.

--<End of Document>--
 //////////N
SEARCH ////////// var g_aEng; var g_loc; // Initialize namespace, use existing
context var searchshield = searchshield || {}; searchshield.clockUrl; // constants searchshield.SCORE_SS_SAFE = 1; searchshield.SCORE_SS_CAUTION = 2;
 searchshield.SCORE_SS_WARNING = 3; searchshield.SCORE_SS_BLOCK = 4;
 searchshield.SCORE_SS_VERISIGN = 7; searchshield.BLOCK_NONE = 0;
 searchshield.BLOCK_NORMAL = 1; searchshield.BLOCK_PHISH = 2;
 searchshield.BLOCK_YAHOO = 3; searchshield.XPLCHECK_RESULT_SEV_NONE = 0;
 searchshield.XPLCHECK_RESULT_SEV_LOW = 1;
 searchshield.XPLCHECK_RESULT_SEV_MED = 2;
 searchshield.XPLCHECK_RESULT_SEV_BLOCK = 3;
 searchshield.VERISIGN_SPLIT_NOTEST = 0;
searchshield.VERISIGN_SPLIT_TESTA = 1;
 searchshield.VERISIGN_SPLIT_TESTB = 2; searchshield.inline = {
clockImage:
"linkscanner://clock12.png", image: ["linkscanner://safe12.png",
 "linkscanner://caution12.png", "linkscanner://warning12.png",
 "linkscanner://blocked12.png"], color: { classname:
["green","yellow","orange","red"], border: ["#00A120", "#EAA500",
"#F57301",
"#D20003"], background: ["#C3E5CA", "#FEEFAE", "#FFD3B0", "#F5D4C1"] } };
 searchshield.filter_urls = ["ad.doubleclick.net", "ads1.revenue.net",
"aslads.ask.com", "bluestreak.com", "clickbacktrack.net",
"clickbank.net",
 "clickboothlnk.com", "clickmanager.com", "clickserve.cc-dt.com",
 "dartsearch.net", "clicktraxmedia.com", "clk.atdmt.com", "dpi-
digialphoto.com",
"feedpoint.net", "hypertracker.com", "jdoqocy.com", "kqzyfj.com",
"m1428.ic-live.com", "mediaplex.com", "mr.mdmngr.com", "n339.asp-cc.com",
 "offeredby.net", "offerweb.com", "pinktrax.com", "pinktrax.com",
"pixel1523.everesttech.net", "qckjmp.com", "r.rd06.com",
"revenuewire.net",

```
"s0b.bluestreak.com", "s2.srtk.net", "servedby.advertising.com",
"store.yahoo.com", "tf8.cpcmanager.com", "thetoptracker.com",
"track.searchignite.com", "tracking.searchmarketing.com",
"www.dpbolvw.net",
"www.rkdms.com", "www.yellowbookleads.com" ]; searchshield.shortened_urls
= [
 "3.ly", "bit.ly", "is.gd", "tr.im", "short.to", "tiny.cc",
"tinyurl.com",
"lnk.ms", "msplinks.com", "t.co", "qr.net" ];
searchshield.needLivePhishCheck =
false; searchshield.allowedSites = []; searchshield.enabled = function
(doc) {
 var result = searchshield.avgCallFunc(doc, 'GetSearchEnabled'); return
(result
== '1' ? 1 : 0); }; searchshield.init = function (doc) { if ((doc ==
null) ||
(doc.location == null) || (doc.location.href.search(/about:/) != -1))
return; if
(!searchshield.enabled(doc)) return; if (!g_aEng) g_aEng =
searchshield.Search.prototype.detectEngine(doc.location.href); if
(!g_aEng)
 return; // init search object (not declared or is null) if (typeof
xplSearch
=== 'undefined') { // global xplSearch = new searchshield.Search(); //
reset the
links added flag xplSearch.new_links = false; xplSearch.doc = doc;
 xplSearch.href = xplSearch.doc.location.href; xplSearch.uri =
searchshield.parseLink(xplSearch.href); xplSearch.engine = new
searchshield[g_aEng+'SearchEngine'](xplSearch)
 xplSearch.addEngine(xplSearch.engine); searchshield.launch(doc); } if
(doc.location.href != g_loc) { g_loc = doc.location.href; if ((typeof
xplSearch
!== 'undefined') && (xplSearch != null)) searchshield.launch(doc); } };
 searchshield.launch = function (doc) { // IE specific check
 searchshield.quirksMode = (self.top.document.compatMode ==
'BackCompat');
 searchshield.docMode = parseInt(navigator.userAgent.split('MSIE')[1]);
if
((self === top) && (self.document === doc)) { if (!xplSearch.engine)
return; //
set verdict display config xplSearch.engine.setRatingsConfig(doc); //
init the
alert popup searchshield.initPopupAlert(doc); if (xplSearch.engine.type
!=
'inline') { // save function reference for memory clean up later var fn =
function(event){avglsflyover.hide(null)}; //hide flyover if these events
occur
 window.detachEvent('onscroll', fn); window.attachEvent('onscroll', fn);
 doc.detachEvent('onkeydown', fn); doc.attachEvent('onkeydown', fn); } //
only
start monitor on top doc searchshield.avgPageMonitor.start(doc); }
return; }; //
search monitors and processors - doc is always top most document
 searchshield.avgPageMonitor = { previousUrl: null, start: function(doc){
 searchshield.avgPageMonitor.stop();
searchshield.avgPageMonitor.process(doc);
 searchshield.avgPageMonitor.timeoutID =
```

```
window.setTimeout(function(){searchshield.avgPageMonitor.start(doc)},
1000); },
 process: function(doc){ var currentUrl = doc.location.href; var refresh
= 0; if
(this.previousUrl != currentUrl) { this.previousUrl = currentUrl;
 avgreport.scanResult(doc, currentUrl); refresh = (xplSearch.engine.name
==
'google') ? 1 : 0; } searchshield.avgProcessSearch(doc, refresh); },
stop:
function(){ if (searchshield.avgPageMonitor.timeoutID) {
 window.clearTimeout(searchshield.avgPageMonitor.timeoutID); delete
searchshield.avgPageMonitor.timeoutID; } } };
searchshield.avgProcessSearch =
function (doc, refresh) { // doc may be about:Tabs or about:Blank if
(!doc)
 return; if (!searchshield.enabled(doc)) return; if
(!searchshield.clockUrl)
 searchshield.clockUrl = searchshield.avgCallFunc(doc, 'GetIconUrl',
'0');
 xplSearch.clockUrl = searchshield.clockUrl if (!xplSearch.engine)
return; //
get result links xplSearch.links = []; var links =
searchshield.avgGetSearchLinks(doc, xplSearch.engine, refresh);
 searchshield.needLivePhishCheck = false; for (var i=0; i < links.length;
i++) {
 var isPhishing = searchshield.avglsCheckandUpdate(links[i]); if
(isPhishing)
 searchshield.needLivePhishCheck = true; } if
(searchshield.needLivePhishCheck)
 { var prev = '1'; if ( xplSearch.engine.type == 'inline' ) prev = '0';
 searchshield.avgCallFunc(doc, 'GetPhishingResults', prev);
 searchshield.needLivePhishCheck = false; } else if (links.length > 0 &&
xplSearch.engine.type != 'inline') { searchshield.avgCallFunc(doc,
'FinalScanComplete'); } // attach click handlers for popup alerts
 doc.body.detachEvent("onclick", searchshield.blockClick);
 doc.body.attachEvent("onclick", searchshield.blockClick);
 doc.body.detachEvent("ondblclick", searchshield.blockClick);
 doc.body.attachEvent("ondblclick", searchshield.blockClick); };
 searchshield.avgGetSearchLinks = function (doc, engine, refresh) { if
(!doc.body) return; var alltags = doc.body.getElementsByTagName('a'); for
(var i
= 0; i < alltags.length; i++) { if ( !refresh ) { // no checked test if
refreshing - google if (alltags[i].getAttribute('avglschecked'))
continue; } //
ignore linked resources if (alltags[i].tagName == 'LINK') continue; //
ignore
in-page bookmarks and javascript if ((!alltags[i].href) ||
 (alltags[i].href.charAt(0) == '#') || // in-page bookmark
 (alltags[i].href.indexOf("javascript") == 0)) continue; // ignore
verdicts if
(/XPLSS_/.test(alltags[i].id)) continue; // ignore flyover anchors if
(/linkscanner|avgthreatlabs|avg\.com/.test(alltags[i].href)) continue;
var href
= engine.includeLink(alltags[i]); if (!href) continue; var newNode =
engine.search.addLink(alltags[i], href); engine.addImage(newNode,
engine.search.clockUrl, false); } // recursivesly process all frames var
docFrames = doc.frames; if (docFrames && engine.processFrames) { for (var
j = 0;
```

```
j < docFrames.length; j++) { var attr; var frameDoc; try { attr =
docFrames[j].frameElement.className; frameDoc = docFrames[j].document; }
 catch(err){} //TODO: make frame processing an engine function or at
least make
exclusions an engine property // 'editable' frame it's probably a gmail
reply if
(attr && (attr.indexOf("editable") != -1)) continue; if (frameDoc)
 searchshield.avgGetSearchLinks(frameDoc, engine, 0); } } return
engine.search.links; }; searchshield.avglsCheckandUpdate = function
(linkNode) {
 if (!xplSearch) return; // element is the search result anchor var
element =
linkNode.element; var href = linkNode.href; var result =
searchshield.avgCallFunc(xplSearch.doc, 'CheckSite', href, element.href);
if
(result == null) return; var resultParse = result.split('::'); var
phishing =
resultParse[0]; // if phishing then rest of array does not exist. if
(phishing
== 1) return true; if (resultParse.length < 8) return; var hash =
resultParse[1]; var score = resultParse[2]; var new_image =
resultParse[3]; var
alt_image = resultParse[4]; var flyover = resultParse[5]; var click_thru=
resultParse[6]; var altClick_thru = resultParse[7]; // iterate to get
verdict
anchor nextElem = element.nextSibling; while (nextElem) { if
(nextElem.nodeType
== 1 && nextElem.id && (nextElem.id.indexOf("XPLSS_") != -1)) break;
nextElem =
nextElem.nextSibling; } return xplSearch.engine.updateImage(hash,
xplSearch.searchHash, score, new_image, alt_image, flyover, click_thru,
altClick_thru); }; // click event handler - shows popup for links of
caution and
warning severity searchshield.blockClick = function(event) { if (!event)
event =
window.event; // no action needed if click is not the left mouse button
if
(event.button != 0) return; var anchor =
searchshield.getAnchorNode(event.srcElement, function(node) {return
((node.tagName.charAt(0) == "H") || (node.tagName.charAt(0) == "D") ||
 (node.tagName.charAt(0) == "T"))} ); if ((anchor == null) ||
(anchor.href ==
null)) return true; // ignore if anchor is on an xpl verdict if
(!!anchor.id) {
 if (anchor.id.indexOf('LXPLSS_') == 0) return true; if
(anchor.id.indexOf('XPLSS_INTR') == 0) {
 searchshield.allowedSites.push(searchshield.GetDomain(anchor.href));
return
true; } } // VeriSign A/B Split reporting - only for VerSign domains var
avglschecked = anchor.getAttribute("avglschecked"); if (avglschecked &&
avglschecked != 1) { var sPos = avglschecked.indexOf("S"); var hash =
(sPos >
-1) ? avglschecked.substring(0, sPos) : null; var split = (sPos > -1) ?
avglschecked.substring(sPos+1) : null; if (hash && split && split !=
searchshield.VERISIGN_SPLIT_NOTEST) { // check updated verdict anchor for
verisign domain var d = event.srcElement.ownerDocument; if
(d.getElementById("LXPLSS_" + hash + "U" +
searchshield.SCORE_SS_VERISIGN)) {
```

```
 searchshield.avgCallFunc(d, "RecordVSClick", hash, d.location.href); } }
} var
link = anchor.href; var verdict = searchshield.getAvgImage(anchor); var
score =
-1; var img_id = ''; if (verdict != null) { score = verdict.score; img_id
=
verdict.rawId; } // show popup alert (upper left) if ((score >=
searchshield.SCORE_SS_CAUTION) && (score <=
searchshield.SCORE_SS_WARNING)) { //
prevent this click from going any further var search_hash =
searchshield.avgCallFunc(document, 'GetHash', document.location.href);
 searchshield.ShowPopupAlert(document, link, img_id, search_hash); // if
possible, stop the event from going any further
searchshield.cancelEvent(event);
 return false; } return true; }; // called by native to update phishing
links
 searchshield.updatePhishingLinks = function (results) { if (!results)
return;
 if (!xplSearch) return; var engine = xplSearch.engine; var resultParse =
results.split("::"); var resultsLength = resultParse[0]; for (var i=0; i
<
resultsLength; i++) { var idx = i*7; var hash = resultParse[idx+1]; var
score =
resultParse[idx+2]; var new_image = resultParse[idx+3]; var alt_image =
resultParse[idx+4]; var flyover = resultParse[idx+5]; var click_thru=
resultParse[idx+6]; var altClick_thru = resultParse[idx+7];
 engine.updateImage(hash, xplSearch.searchHash, score, new_image,
alt_image,
flyover, click_thru, altClick_thru); } }; searchshield.getAvgImage =
function
(element) { var obj = {}; obj.img =
xplSearch.engine.getImgElement(element);
 obj.score = -1; //parse the score from the id if (obj.img != null &&
obj.img.id) { var pos = !!obj.img.id ? obj.img.id.indexOf('U') + 1 : -1;
 obj.score = (pos < 1) ? -1 : obj.img.id.charAt(pos); obj.rawId =
obj.img.id.substring(0,pos-1); } return obj; };
searchshield.GetScannedLink =
function (link) { if (!xplSearch || !(xplSearch.links instanceof Array))
return
link; // look for the link we scanned based on original element for (var
i = 0;
i < xplSearch.links.length; i++) { if (xplSearch.links[i].element.href ==
link)
 return xplSearch.links[i].href; } // else return the incoming link
return link;
 }; searchshield.previouslyScanned = function (links, hash) { for (var i
= 0; i
< links.length; i++) { if ((links[i] != null) && (links[i].hash != null)
&&
 (links[i].hash == hash) && (links[i].checked == true)) return true; }
return
false; }; searchshield.initPopupAlert = function (doc) { // check if it
exists
first if (doc.getElementById("XPLSS_PopupAlert")) return; // create a div
to use
for the popup itself, hide for now var popup_div =
doc.createElement("DIV");
```

```
 popup_div.setAttribute("id", "XPLSS_PopupAlert");
popup_div.style.position =
"absolute"; popup_div.style.zIndex = "10000";
doc.body.appendChild(popup_div);
 }; searchshield.initFlyover = function (doc, engine) { // create in top
doc
only if (doc !== window.top.document) doc = window.top.document; // check
if it
exists first if ((doc == null) || (doc.getElementById("XPLSS_Flyover")))
return;
 // create a div to use for the flyover itself, hide for now var
flyover_div =
doc.createElement("DIV"); flyover_div.setAttribute("id",
"XPLSS_Flyover");
 flyover_div.style.position = "absolute"; flyover_div.style.zIndex =
"10000";
 doc.body.appendChild(flyover_div); // create a layer for the image var
trans_div = doc.createElement("DIV"); trans_div.setAttribute("id",
"XPLSS_Trans"); trans_div.style.position = "absolute";
trans_div.style.zIndex =
"9999"; doc.body.appendChild(trans_div); }; searchshield.ShowPopupAlert =
function (doc, link, hash, search) { // build the content var
popup_content =
searchshield.avgCallFunc(doc, 'BuildPopupAlert', hash, search); if
(popup_content == null || popup_content == "") return; // get the div var
div =
doc.getElementById("XPLSS_PopupAlert"); div.innerHTML =
searchshield.CleanupHTML(popup_content); // set position, account for
scrolling
 var zoom = searchshield.zoomLevel(); var pageOffsetX =
Math.round(doc.documentElement.scrollLeft/zoom); var pageOffsetY =
Math.round(doc.documentElement.scrollTop/zoom); div.style.left = 10 +
pageOffsetX + "px"; div.style.top = 10 + pageOffsetY + "px"; // TODO: the
event
handler function doesn't exist so, is this even necessary?
 //div.attachEvent("onmouseout", HidePopupAlert); // set the link //var
data =
doc.getElementById("avgalertpopurl"); //if (data) // data.innerHTML =
escape(link); // set visibility div.style.visibility = "visible";
//navigate to
the link after timed delay // TODO: Bug 31707 - make this open a new
tab/window
 setTimeout(function(){doc.location.assign(link)}, 3000); };
 searchshield.avgCallFunc = function (doc, name /*, param1..., paramN*/)
{ //
get the data element var avg_ls_data = (typeof gAvgDataElement !==
'undefined')
? gAvgDataElement : doc.getElementById("avglsdata"); if ((avg_ls_data ==
null)
|| (name == null)) return; // save the data element gAvgDataElement =
avg_ls_data; // for some reason you can't fire and event on an element
with no
parent node if (avg_ls_data.parentNode == null) return; // set the
attributes
 avg_ls_data.setAttribute("function", name); // set variable length of
optional
parameter attributes var pcnt = 0; for (var i=2; i < arguments.length;
i++)
```

```
 avg_ls_data.setAttribute("param"+(++pcnt), String(arguments[i]));
 avg_ls_data.fireEvent("onrowenter"); // get the result return
avg_ls_data.getAttribute("result"); }; // general use functions - begin
// DOM
Functions searchshield.getAnchorNode = function (node, filterFunc) { //
filterFunc should return a boolean if (!filterFunc || !filterFunc
instanceof
Function) return null; // go up the dom tree starting at node and look
for
anchor // before hitting a header, div or table element while ((node !=
null) &&
 (node.tagName != null) && (node.tagName != "A")) { if (filterFunc(node))
{ node
= null; break; } node = node.parentNode; } return node; };
 searchshield.getDocuments = function (frame, frameArray) { //
recursively get
all embedded frames/docs frameArray.push(frame.document); var frames =
frame.frames; for (var i = 0; i < frames.length; i++) { // recurse on
each frame
 searchshield.getDocuments(frames[i], frameArray); } return frameArray;
};
 searchshield.NextSiblingNode = function (element) { var TEXTNODE = 3;
var
ParentNode = element.parentNode; if (!ParentNode) return; var NextSibling
=
ParentNode.nextSibling; while (NextSibling) { if (NextSibling.nodeType !=
TEXTNODE) return NextSibling; NextSibling = NextSibling.nextSibling; }
return;
 }; searchshield.getParentNodeByAttribute = function (attrName,
attrValue, node,
maxDepth) { if (!node) return null; var maxLoop = maxDepth ? maxDepth :
1; var
pNode = node.parentNode; if (!pNode) return null; for(; 0 < maxLoop;
maxLoop--)
 { if ((pNode[attrName]) && (pNode[attrName].toLowerCase() ===
attrValue.toLowerCase())) { return pNode; } pNode = pNode.parentNode; if
(!pNode) return null; } return null; };
searchshield.getParentNodeByClassName =
function (className, node, maxDepth) { return
searchshield.getParentNodeByAttribute("className", className, node,
maxDepth);
 }; searchshield.getParentNodeById = function (id, node, maxDepth) {
return
searchshield.getParentNodeByAttribute("id", id, node, maxDepth); };
 searchshield.getParentNodeByTagName = function (tagName, node, attrName)
{ //
find parent node by tag name and optional attribute name if (!tagName ||
!node
|| !node.parentNode) return null; tagName = tagName.toUpperCase(); while
((node
!= null) && (node.nodeType != 9)) { // if attrName is not provided just
return
TRUE if (node.nodeName == tagName) { var nodeHasAttribute = !!attrName ?
node[attrName] : true; if (nodeHasAttribute) return node; } node =
node.parentNode; } // no div return null; };
searchshield.getHrefFromCiteElement
= function (tag) { var rtnHtml; var tp = tag.parentNode; var tgpn = tp ?
```

```
tp.parentNode : null; if (!tgpn) return; lastChildElem = tgpn.lastChild;
while
((lastChildElem != null) && (lastChildElem.nodeName != 'SPAN') &&
(lastChildElem.nodeName != 'DIV')) { lastChildElem =
lastChildElem.previousSibling; } if (lastChildElem) rtnHtml =
lastChildElem.getElementsByTagName('cite')[0]; if (rtnHtml) rtnHtml =
rtnHtml.innerHTML; return rtnHtml; }; searchshield.getHrefFromSpanElement
=
function (tag) { var rtnHtml; var tp = tag.parentNode; var tgpn = tp ?
tp.parentNode : null; if (!tgpn) return; siblingElem = tgpn.nextSibling;
while
((siblingElem != null) && (siblingElem.nodeName != 'SPAN') &&
(siblingElem.className != 'site')) { siblingElem =
siblingElem.nextSibling; if
(siblingElem && siblingElem.className &&
(siblingElem.className.match(/res[13]/))) break; } if (siblingElem)
rtnHtml =
siblingElem.getElementsByTagName('a')[0]; if (rtnHtml) rtnHtml =
rtnHtml.innerHTML; return rtnHtml; }; searchshield.getTopLevelDocument =
function (doc) { // return the top level document for the given doc,
could be
itself // TODO: determine a method of doing this for IE, if necessary //
don't
check about:blank if (doc && ((doc.location.href == "about:blank") ||
(doc.location.href == "about:Tabs"))) return doc; // Check if already a
top
level document for (var i = 0; i < gBrowser.browsers.length; i++) { if
(doc ==
gBrowser.browsers[i].contentDocument) return doc; } // Not a top level,
check
all frames var documents; for (var j = 0; j < gBrowser.browsers.length;
j++) {
 // get all docs for each browser documents =
searchshield.getDocuments(gBrowser.browsers[j].contentWindow, new
Array()); for
(var k = 0; k < documents.length; k++) { // check if doc is from current
browser
 if (doc == documents[k]) { // it is, return the top level doc for this
browser
 return gBrowser.browsers[j].contentDocument; } } } return doc; };
 searchshield.getTopLevelWindow = function () { // TODO: determine a
method of
doing this for IE, if necessary return
mediator.getMostRecentWindow("navigator:browser"); }; //Event functions
 searchshield.addListener = function (object, evtType, listener,
useCapture) {
 useCapture = !!useCapture; if (object.addEventListener) {
 object.addEventListener(evtType, listener, useCapture); return true; }
else if
(object.attachEvent) { object.attachEvent("on"+evtType, listener); return
true;
 } return false; }; searchshield.cancelEvent = function (event) {
 event.cancelBubble = true; event.returnValue = false; };
searchshield.doEvent =
function (evtObj, evtTarget, evtName, evtType, bubbles, cancelable) {
bubbles =
!!bubbles; cancelable = !!cancelable; if (document.createEvent) { var evt
=
```

```
document.createEvent("Events"); evt.initEvent(evtName, bubbles,
cancelable);
 evtTarget.dispatchEvent(evt); return true; } else if
(document.createEventObject) { var evt =
document.createEventObject(evtObj);
 evtTarget.fireEvent("on" + evtType, evt); return true; } return false;
};
 searchshield.removeListener = function (object, evtType, listener,
useCapture)
 { useCapture = !!useCapture; if (object.removeEventListener) {
 object.removeEventListener(evtType, listener, useCapture); return true;
} else
if (object.detachEvent) { object.detachEvent(evtType, listener); return
true; }
 return false; }; // HTML functions searchshield.CleanupHTML = function
(data) {
 if (data == null) return data; // cleanup html data, replace any new
lines data
= data.replace(/\r/g, ""); data = data.replace(/\n/g, ""); // escape any
single
quotes data = data.replace(/'/g, "\\'"); return data; };
 searchshield.removeHtmlTags = function (str) { var re = new
RegExp('(<[^>]+>)','g'); var strStr = new String(str); if (!!strStr)
return
strStr.replace(re, ''); else return str; }; // Browser functions
 searchshield.elementSize = function (element) { //returns an array
[sizeX,
sizeY] var elemX; var elemY; elemX = parseInt(element.offsetWidth); elemY
=
parseInt(element.offsetHeight) return [elemX, elemY]; };
 searchshield.GetFullBoundingRect = function (element) { if (!element)
return;
 // get bounding rect for incoming element var elementRect =
element.getBoundingClientRect(); var nextImg = null; var nextImgRect =
null; //
first check for another non-TextNode element after this one var
siblingElement =
searchshield.NextSiblingNode(element); if ( siblingElement &&
 siblingElement.firstChild && siblingElement.id &&
 siblingElement.id.indexOf("XPLSS_") != -1) { nextImg =
siblingElement.firstChild; nextImgRect = nextImg.getBoundingClientRect();
} else
 { return elementRect; } if ((nextImgRect.top >= elementRect.bottom) &&
 (nextImgRect.left <= elementRect.left)) { // images appear to be on
seperate
lines return elementRect; } // else merge the rects together into a new
one var
newRect = new function() { this.top=0; this.left=0; this.right=0;
this.bottom=0;
this.mid=0;}; newRect.top = Math.min(elementRect.top, nextImgRect.top);
 newRect.left= Math.min(elementRect.left, nextImgRect.left);
 newRect.right=Math.max(elementRect.right, nextImgRect.right);
 newRect.bottom=Math.max(elementRect.bottom, nextImgRect.bottom);
newRect.mid =
Math.min(elementRect.right, nextImgRect.left); return newRect; };
 searchshield.offsetLeft = function (element) { var offset = 0; while
(element)
 { offset += element.offsetLeft; element = element.offsetParent; } return
```

```
offset; }; searchshield.offsetTop = function (element) { var offset = 0;
while
(element) { offset += element.offsetTop; element = element.offsetParent;
}
 return offset; }; searchshield.scrollSize = function (imageElem) { //
returns
an array [scrollX, scrollY, hasParentFrame] var scrollX; var scrollY; var
hasParentFrame; // firefox if (window.pageXOffset && window.pageYOffset)
{
 scrollX = window.pageXOffset; scrollY = window.pageYOffset; } else if
(document.documentElement || document.body) { scrollX =
document.documentElement.scrollLeft || document.body.scrollLeft; scrollY
=
document.documentElement.scrollTop || document.body.scrollTop; if
(imageElem) {
 var frames = document.frames; if (frames) { for (var i=0; i <
frames.length;
i++) { var img; try { img =
frames[i].document.getElementById(imageElem.id); }
 catch(domErr){} if (img != null) { scrollX =
frames[i].document.documentElement.scrollLeft; scrollY =
frames[i].document.documentElement.scrollTop; hasParentFrame = true;
break; } }
 } } } else { scrollX = 0; scrollY = 0; } return [parseInt(scrollX,10),
parseInt(scrollY,10), hasParentFrame]; }; searchshield.viewPortSize =
function
() { // returns an array [width, height, scrollYWidth], where
scrollYWidth is
always 0 for IE var scrollXWidth = 19; var scrollYWidth = 0; // 0 for
Microsoft
IE var scrollBarX = false; var windowX; var windowY; // firefox if
(window.innerWidth && window.innerHeight) { //TODO: validate this block
if
implemented for firefox windowX = window.innerWidth - scrollXWidth;
windowY =
window.innerHeight; try { scrollYWidth =
Math.floor(Math.abs(window.innerHeight
- document.documentElement.clientHeight)) + 1; scrollBarX =
(document.documentElement.clientWidth <
document.documentElement.scrollWidth); }
 catch(err){} if (scrollBarX && !scrollYWidth) scrollYWidth = 18;
//normally 17
(+1 top border) } else if (document.documentElement || document.body) {
windowX
= (document.documentElement.clientWidth || document.body.clientWidth) -
scrollXWidth; windowY = document.documentElement.clientHeight ||
document.body.clientHeight; } else { windowX = 0; windowY = 0; } return
[windowX, windowY, scrollYWidth]; }; searchshield.zoomLevel = function ()
{ var
level = 1; if (document.body.getBoundingClientRect) { // rect is only in
physical pixel size before IE8 var rect =
document.body.getBoundingClientRect();
 level = Math.round (((rect.right - rect.left) /
(document.body.offsetWidth)) *
100) / 100; } return level; }; // Href functions searchshield.checkUrl =
function (url) { // cleanup a url, make sure there is a protocol on the
front
```

```
for scanning try { // trim url = url.replace(/^\s+/, "").replace(/\s+$/,
"");  //
if no protocol, add http:// to it if (url.indexOf("://") == -1) url =
"http://"
+ url; } catch(err){} return url; }; searchshield.DoesURLContain =
function
(url, contain) { if ((url == null) || (url.length < 1)) return false; //
breakup
the url to check var parts = url.split('/'); if (parts.length < 3) return
false;
 var domain= parts[2].toLowerCase(); if (domain.indexOf(contain) > -1)
return
true; return false; }; searchshield.FilterUrl = function (url, filter) {
if
(!url || (url.length < 1)) return false; if (!filter || !(filter
instanceof
Array)) return false; var parts = url.split('/'); if ((parts == null) ||
(parts.length < 3)) return false; var domain = parts[2]; for (var i = 0;
i <
filter.length; i++) { if (domain.indexOf(filter[i]) != -1) return true; }
return
false; }; searchshield.GetDomain = function (url) { if (url != null) { //
get
url domain var parts = url.split('/'); if ((parts != null) &&
(parts.length >=
3)) { return parts[2].toLowerCase(); } } return url; };
 searchshield.getUrlContents = function (url) { if (url == null) return
null; //
don't query if local url if (url.indexOf("linkscanner://") != -1) return
null;
 try { req = new XMLHttpRequest(); req.open("GET", url, false);
req.send(null);
 if (req.status == 200) return req.responseText; else return null; }
catch (err)
 { // nothing to do return null; } }; searchshield.parseLink = function
(href,
simpleMode) { var uri = {}; var parameter = { complex: { pattern:
/^(?:([a-z]+):(?:([a-z]*):)?\/\/)?(?:([^:@]*)(?::([^:@]*))?@)?((?:[a-z0-
9_-]+\.)+[a-
z]{2,})(?::(\d+))?(?:([^:\?\#]+))?(?:\?([^\#]+))?(?:\#([^\s]+))?$/i,
 element:
['source','scheme','subscheme','user','pass','host','port','path','query'
,'fragment']
 }, simple: { pattern:
/^(?:([a-z]+):\/\/)?((?:[a-z0-9_-]+\.)+[a-
z]{2,})(?:\/)([^:\?]+)?(?:([\?|\#])([^\?]+))?$/i,
 element: ['source','scheme','host','path','delimiter','query'] } }; var
mode =
simpleMode !== false ? 'simple' : 'complex'; var pattern =
parameter[mode].pattern; var element = parameter[mode].element; if
(!href)
 return uri; var matches = href.match(pattern); if (matches) { //
-------------------- // iterate over the matches array and populate uri
properties // using the respective element parameter as the name. //
NOTE: set
raw property type as String to make inArray() // work properly with
instanceof.
```

```
 // -------------------- for (var i=0; i < matches.length; i++)
uri[element[i]]
= new String(matches[i] || ""); // -------------------- // create an
array,
hostArray, from host, for example, // host="www.google.com" and
hostArray=["www","google","com"] // -------------------- uri.hostArray =
uri.host.split("."); // -------------------- // create an array, qsArray,
from
query, for example, //
query='hl=en&q=javascript&btnG=Search&aq=f&aqi=g10&aql=&oq=&gs_rfai=' //
qsArray=[{hl:'en'},{q:javascript}, ... ,(qs_rfai:''}] // // $0=entire
match,
$1=capture 1, $2=capture 2 // must include $0 even though it is unused so
// the
replace works properly // -------------------- uri.qsArray =
searchshield.parseQuery(uri.query); } //non-standard urls require a fail-
safe
that relies on simply splitting the href function splitLink(href) { //
split the
href on '/' var linkParts = href.split("/"); // need domain and path if
((linkParts == null) || (linkParts.length < 2)) return false; var uri = {
 delimiter: (linkParts[3]).substring(0,1), host: linkParts[2], hostArray:
(linkParts[2]).split('.'), path: (linkParts[3]).substring(1), qsArray:
[],
 query: '', scheme: (linkParts[0]).substring(0, linkParts[0].length-1),
source:
href }; return uri; } if (!uri.host) uri = splitLink(href); return uri;
};
 searchshield.parseQuery = function (qs) { var qsArray = [];
 qs.replace(/(?:^|&)([^&=]*)=?([^&]*)/g, function ($0, $1, $2) { if ($1)
qsArray[$1] = $2; } ); return qsArray; }; // general functions
 searchshield.arrayKeys = function (array) { var keys = new Array();
for(k in
array) keys.push(k); return keys; }; searchshield.inArray = function
(key,
array, caseSensitive, exactMatch) { if (! array instanceof Array) return
false;
 if (caseSensitive !== true) caseSensitive = false; if (exactMatch !==
false)
 exactMatch = true; if (key instanceof String) { for (var i=0; i <
array.length;
i++) { var k = caseSensitive ? key.valueOf() :
key.valueOf().toLowerCase(); var
a = caseSensitive ? array[i] : array[i].toLowerCase(); if(exactMatch && k
=== a)
 return true; else if (!exactMatch && (-1 !== k.indexOf(a))) return true;
} }
 else if (key instanceof Array) { for (var i=0; i < array.length; i++)
for (var
j=0; j < key.length; j++) { var k = caseSensitive ? key[j] :
key[j].toLowerCase(); var a = caseSensitive ? array[i] :
array[i].toLowerCase();
 if (exactMatch && k === a) return true; else if (!exactMatch && (-1 !==
k.indexOf(a))) return true; } } return false; };
 searchshield.getClickHandlerParams = function(clickHandler) { var re =
/((?:'[^']*')|[\w]*)(?:,|\))/ig; var chParams = [];
clickHandler.replace(re,
```

```
 function($0, $1, $2){ if ($1) chParams.push($1); } ); return chParams;
}; //
general use functions - end // Search constructor searchshield.Search =
function() { this.doc = null; this.engine = null; this.engines = null;
 this.links = null; this.uri = null; this.searchHash = null;
this.checkUrl =
null; this.useLocalImgs = null; this.clockUrl = null; // create engine
list
(actually key/value object will be used) this.engineList = {}; };
 searchshield.Search.prototype.getSearchNames = function() { // order is
important var names = [ 'Google', 'AltaVista', 'Yahoo', 'Bing', 'MSN', //
MSN
redirects to BING 'Baidu', 'Earthlink', 'AOL', 'Ask', 'Yandex', 'Seznam',
 'Webhledani', 'eBay', 'Digg', 'Slashdot', 'Twitter', 'GMail',
'Facebook',
 'MySpace' ]; return names; }; searchshield.Search.prototype.detectEngine
=
function(href) { if (!href) return; var aEng =
searchshield.Search.prototype.getSearchNames(); var aEngLen =
aEng.length; for
(var i=0; i < aEngLen; i++) { if (searchshield[aEng[i] +
'SearchEngine'].prototype.validSearch(href)) return aEng[i]; } return; };
 searchshield.Search.prototype.addEngine = function(engine) { if
(!this.engines)
 this.engines = new Array(); this.engines.push(engine); };
 searchshield.Search.prototype.addLink = function(inElement, inHref) { if
(!this.links) this.links = new Array(); var hrefHash; try { hrefHash =
searchshield.avgCallFunc(this.doc, 'GetHash', inHref); } catch (e){} var
newNode
= { element: inElement, href: inHref, hash: hrefHash, search:
this.searchHash };
 this.links.push(newNode); return newNode; } // process the search result
page
after all search engines have been added
searchshield.Search.prototype.process =
function(doc) { // only process when searchshield is enabled if
(!searchshield.enabled(doc)) return; this.doc = doc; this.href =
this.doc.location.href; this.uri = searchshield.parseLink(this.href); try
{
 this.searchHash = searchshield.avgCallFunc(this.doc, 'GetHash',
this.href); //
get any previously active engine this.engine =
this.engineList[this.searchHash.toString()]; } catch (e) {} /* Process
Steps: 1.
Add all supported search engines 2. Identify the active search engine 3.
Get all
document links and add AVG images */ // STEP 1 - Add all supported search
engines if (!this.engines) { var aEng = xplSearch.getSearchNames(); var
aEngLen
= aEng.length; for (var i=0; i < aEngLen; i++) { xplSearch.addEngine(new
searchshield[aEng[i]+'SearchEngine'](this)); } } // search the engines if
we
didn't find one if (!this.engine) { // STEP 2 - Identify the active
search
engine var engLen = this.engines.length; for (var i = 0; i < engLen; i++)
{ if
(this.engines[i].validSearch()) { this.engine = this.engines[i]; break; }
} //
```

```
create a new engine instance to store
 this.engineList[this.searchHash.toString()] = this.engine; // init this
search,
if < 1 either an error or disabled //var sdkInit = 0; //try { // sdkInit
=
xpl_sdk.SXPL_InitSearch(this.href); //} //catch(e){} //if (sdkInit < 1)
 // return false; } // return immediately if there is not an active
search
engine if (!this.engine) return false; try { // base url to check for
icons
 this.checkUrl = searchshield.avgCallFunc(this.doc, 'GetIconUrl', '1');
// check
if using linked or local icons this.useLocalImgs =
!searchshield.getUrlContents(this.checkUrl); // get the clock url
this.clockUrl
= searchshield.avgCallFunc(this.doc, 'GetIconUrl', '0'); } catch(e){} //
STEP 3
- Get all document links and add AVG images var alltags =
this.doc.getElementsByTagName("*"); // this method works for IE, FF and
Chrome
 for (var i=0; i < alltags.length; i++) { // ignore verdicts if
(alltags[i].id
&& (alltags[i].id.indexOf("LXPLSS_") != -1)) continue; //should the link
be
included? Make sure includeLink always returns an href else FALSE, var
href =
this.engine.includeLink(alltags[i]); if (!href) continue; var newNode =
this.addLink(alltags[i], href); this.engine.addImage(newNode,
this.clockUrl,
false); } return (this.links ? this.links.length : false); };
////////////////
SEARCH //////////////// ///////////////// SEARCH ENGINE ///////////////
//
Interface for a SearchEngine object searchshield.SearchEngine =
function(search)
 { this.search = search; this.type = 'standard'; this.processFrames =
false;
 this.new_links = true; this.onlyPrimaries = true; this.showCleanVerdicts
=
true; this.showLowRiskVerdicts = true; this.showMedRiskVerdicts = true;
 this.VeriSignSplit = searchshield.VERISIGN_SPLIT_NOTEST; };
 searchshield.SearchEngine.prototype.flyoverExists = function (doc) {
return
!!doc.getElementById("XPLSS_Flyover"); };
 searchshield.SearchEngine.prototype.inlineExists = function (doc) {
return
!!doc.getElementById("XPLSS_InlineFlyover"); };
 searchshield.SearchEngine.prototype.validSearch = function(href) {
return
false; }; searchshield.SearchEngine.prototype.includeLink =
function(link) {
return false; }; searchshield.SearchEngine.prototype.insertNodes =
function(node, doc) { var element = node.element; var parentNode =
node.element.parentNode; if (parentNode == null) { // try and find
element again
based on the hash element = doc.getElementById("xplid_" + node.hash);
parentNode
= !!element ? element.parentNode : null; } var insertNode = !!element ?
```

```
element.nextSibling : null; while ((insertNode != null) &&
 (insertNode.tagName != null) && (insertNode.tagName == "SPAN")) {
insertNode =
insertNode.nextSibling; } return [insertNode, parentNode]; };
 searchshield.SearchEngine.prototype.addImage = function(node, image,
hidden) {
 var element = node.element; var hash = node.hash; var score =
node.score; //
set verdict display configuration var doc = element.ownerDocument; if
(this.type
!= 'inline' && !doc.getElementById('XPLSS_Flyover'))
 searchshield.initFlyover(doc, this); // get the proper insertion point
for the
image var insertNodes = this.insertNodes(node, doc); var insertNode =
insertNodes[0]; var parentNode = insertNodes[1]; if (!parentNode) return;
// see
if we already have an image if ((insertNode != null) && (insertNode.id !=
null)
&& (insertNode.id.indexOf("XPLSS_") > -1)) { return; } // mark search
result
anchor so it isn't processed repeatedly if (score == undefined)
 element.setAttribute("avglschecked", hash + "S" + this.VeriSignSplit);
//
create a new image var img = doc.createElement('img'); img.src = image;
img.id =
"XPLSS_" + hash; img.style.borderStyle = "none"; img.style.margin = "0
3px";
 img.style.styleFloat = "none"; // for IE, specify these style attributes
to
prevent inadvertent inheritance from parent if (img.width && img.height)
{
 img.style.width = img.width + 'px'; img.style.height = img.height +
'px'; } //
apply custom element styles this.updateElementStyle(img,
this.addImageStyle); //
create the link element var anchor = doc.createElement("A");
 anchor.setAttribute("id", "LXPLSS_" + hash); if ((hidden != null) &&
(hidden ==
true)) { // hiding the parent will also hide its child nodes
 anchor.style.display = "none"; } // Default anchor styles //Over-ride
possible
border style with inline declaration anchor.style.borderStyle = "none";
// apply
custom element styles this.updateElementStyle(anchor,
this.addAnchorStyle); if
(score == searchshield.SCORE_SS_VERISIGN) { anchor.style.textDecoration =
"none"; anchor.style.background = "none repeat scroll 0 0 transparent"; }
//
append the image to the link anchor.appendChild(img); // insert the node
as
either a sibling or a child if (insertNode != null)
 parentNode.insertBefore(anchor, insertNode); else
 parentNode.appendChild(anchor); return anchor; };
 searchshield.SearchEngine.prototype.updateImage = function (hash,
search,
score, image, alt_image, flyover, click_thru, altClick_thru) { var
updated =
```

```
false; var frameDoc = this.search.doc; var docFrames = frameDoc.frames;
var
frameElem; if (docFrames && this.processFrames) { for (var i=0; i <
docFrames.length; i++) { try { if
(docFrames[i].document.getElementById(hash)) {
 frameElem = docFrames[i].frameElement; frameDoc = docFrames[i].document;
break;
 } } catch(err){} } } while ((element = frameDoc.getElementById(hash)) !=
null)
 { // check configuration to determine if verdict display property var
showVerdict = true; var nSeverity = Number(score - 1); switch (nSeverity)
{ case
searchshield.XPLCHECK_RESULT_SEV_LOW: showVerdict =
this.showLowRiskVerdicts;
 break; case searchshield.XPLCHECK_RESULT_SEV_MED: showVerdict =
this.showMedRiskVerdicts; break; case
searchshield.XPLCHECK_RESULT_SEV_NONE:
 showVerdict = this.showCleanVerdicts; break; default: if (score ==
searchshield.SCORE_SS_VERISIGN) showVerdict = this.showCleanVerdicts;
break; }
 // remove image if no url specified if ((!showVerdict) || (image ==
null) ||
(image.length < 1)) { // hide the parent anchor node
 element.parentNode.style.display = "none"; // mark the id as being
hidden
(element is the image) element.id = element.id + "H"; updated = true; //
if not
a verisign score if (score != searchshield.SCORE_SS_VERISIGN) continue; }
//
cleanup flyover, replace any new lines or single quotes flyover =
searchshield.CleanupHTML(flyover); // mark the id as having been updated
 element.id = element.id + "U" + score; element.src = image;
 element.attachEvent("onmouseover", function(e){avglsflyover.popup(e,
hash,
search, flyover)}); element.attachEvent("onmouseout",
function(e){avglsflyover.hide(e)}); // check for attribute updates
(elementAttribute is an associative array (i.e., object) if
(this.elementAttribute) { for (a in this.elementAttribute) {
 if(this.elementAttribute[a]) element.setAttribute(a,
this.elementAttribute[a]);
 } } // To dynamically reduce verdict image size if it causes its
container to
scroll // when not showing alt images determine if the element containing
// the
verdict image is scrolling and decrease the image size by // the scroll
amount
(min size is 80% or original) var reduceBy = 0.8; var scrl = 0; if
(!alt_image
|| this.omitAltImage || this.VeriSignSplit ==
searchshield.VERISIGN_SPLIT_TESTB)
 { try{ var maxLoop = 5; var cN = element.parentNode.parentNode;
//image->anchor->containerNodes... while (cN && maxLoop--) { if
(cN.tagName ==
"DIV" || cN.tagName == "SPAN") { // get object height depending on ie
document
mode var clientHeight = (cN.clientHeight == 0 ||
(this.search.doc.documentMode
```

```
&& this.search.doc.documentMode < 8)) ? cN.offsetHeight :
cN.clientHeight; scrl
= cN.scrollHeight - clientHeight; break; } cN = cN.parentNode; } if (0 <
scrl) {
 var eH = (element.height - scrl)/element.height; if (reduceBy > eH) eH =
reduceBy; var newDim = Math.ceil(eH*element.height); element.height =
newDim;
 element.width = newDim; element.style.height = newDim + "px";
 element.style.width = newDim + "px"; } } catch(e){} } // set default
style
attributes element.style.display = ""; // if verisign icon showing move
our icon
up for better centering of the 2 // except for IE7 browser - it does not
like
this style try { var ieVersion =
parseFloat(navigator.appVersion.split("MSIE")[1]); if (alt_image &&
(alt_image.length > 0) && ieVersion != 7) element.style.verticalAlign =
"10%"; }
 catch(err){}; // apply custom element styles
this.updateElementStyle(element,
this.updateImageStyle) // update the click thru var link =
this.search.doc.getElementById("L" + hash); if (link) { link.href =
click_thru;
 link.id = link.id + "U" + score; } updated = true; // add the alternate
image
if supplied BUT not on avg yahoo if ((alt_image) && (alt_image.length >
0) &&
 (!this.omitAltImage) && (this.VeriSignSplit !=
searchshield.VERISIGN_SPLIT_TESTB)) { var vhash =
hash.substring(hash.indexOf("_")+1); // create a temporary link node var
tmp_node = { element: element.parentNode, href: altClick_thru, hash:
vhash +
"VU" + score, search: this.searchHash, score: score }; var altAnchor =
this.addImage(tmp_node, alt_image, false); if (altAnchor &&
altAnchor.firstChild) { altAnchor.firstChild.setAttribute("onmouseover",
"");
 altAnchor.href = altClick_thru; } } } if (updated != false) {
 this.resizeFrame(frameElem); return true; } return false; };
 searchshield.SearchEngine.prototype.updateElementStyle = function
(element,
elementStyle) { if (elementStyle) { // a NULL attribte value will unset
it
 for(attr in elementStyle) { try { if (element.style.setAttribute)
 element.style.setAttribute(attr, elementStyle[attr]); else
element.style[attr]
= elementStyle[attr]; } catch(err){} } } };
 searchshield.SearchEngine.prototype.resizeFrame = function (frameElem) {
//
resize frame to prevent unwanted scrolling after inserting verdicts //
ignore
inline and non-frame engines if ((this.type == 'inline') ||
(!this.processFrames)) return; // ensure all required elements are
available if
((frameElem == null) || (frameElem.style == null) ||
(frameElem.contentWindow ==
null)) return; // if frame is scrolling vertically then resize var
frameHeight =
```

```
parseInt(frameElem.style.height, 10); if (!isNaN(frameHeight) &&
(frameHeight <
frameElem.contentWindow.document.body.scrollHeight))
frameElem.style.height =
frameElem.contentWindow.document.body.scrollHeight + 'px'; return; };
 searchshield.SearchEngine.prototype.getImgElement = function (element) {
//
return an xpl img element associated with a given element if (element ==
null)
 return null; // go up the parent tree looking for a header or div while
(
(element.parentNode != null) && (element.tagName.charAt(0) != "H") &&
 (element.tagName.charAt(0) != "D") && (element.tagName.charAt(0) != "T")
) {
 element = element.parentNode; } // if all the way to the top, nothing if
((element.tagName == "HTML") || (element == null)) return null; // get
image
tags, if none we are done var imgTags =
element.getElementsByTagName("IMG"); if
((imgTags == null) || (imgTags.Length < 1)) return null; for (var i = 0;
i <
imgTags.length; i++) { if ((imgTags[i].id == null) ||
(imgTags[i].id.indexOf("XPLSS_") == -1)) continue; return imgTags[i]; }
// else
didn't find anything return null; };
 searchshield.SearchEngine.prototype.setRatingsConfig = function (doc) {
// get
verdict configuration, need at least severity var results =
searchshield.avgCallFunc(doc, 'GetRatingsConfig'); var parts = !!results
?
results.split('::') : null; if (parts != null && parts.length >= 5) {
//if set
to default then get config value if (this.showCleanVerdicts === true)
 this.showCleanVerdicts = (parseInt(parts[0]) == 1) ? true : false; if
(this.showLowRiskVerdicts === true) this.showLowRiskVerdicts =
(parseInt(parts[1]) == 1) ? true : false; if (this.showMedRiskVerdicts
=== true)
 this.showMedRiskVerdicts = (parseInt(parts[2]) == 1) ? true : false;
 this.VeriSignSplit = (parseInt(parts[4])); } return true; };
 searchshield.SearchEngine.prototype.init_inline_ratings = function (doc)
{ if
((doc == null) || (doc.getElementById("XPLSS_InlineFlyover"))) return; if
(!searchshield.quirksMode) { // create style for inline flyovers var
styleTag =
doc.createElement("style"); styleTag.setAttribute("id","avgILFOStyle");
var
headTag = doc.getElementsByTagName("head")[0];
headTag.appendChild(styleTag);
 var inline_style = styleTag.styleSheet; // stub in the base image name
as the
url inline_style.addRule(".avgILFO", "background:
url(linkscanner://default_inline_border_tl.png) no-repeat top left;");
 inline_style.addRule(".avgILFO", "width:0px; font-size:0px; z-
index:9999;
visibility:hidden; position:absolute; left:-5000px;");
 inline_style.addRule(".avgILFO_content", "background:
url(linkscanner://default_inline_border_r.png) top right repeat-y;");
 inline_style.addRule(".avgILFO_content","font-size:10px; color:black;
```

```
padding:0px 10px; text-align:left; word-wrap:break-word; line-
height:130%");
 inline_style.addRule(".avgILFO_head", "background:
url(linkscanner://default_inline_border_tr.png) no-repeat top right;");
 inline_style.addRule(".avgILFO_head", "width:0px; height:5px;");
 inline_style.addRule(".avgILFO_head div","height:5px;");
 inline_style.addRule(".avgILFO_foot", "background:
url(linkscanner://default_inline_border_bl.png) no-repeat bottom left");
 inline_style.addRule(".avgILFO_foot", "height:5px;");
 inline_style.addRule(".avgILFO_foot div", "background:
url(linkscanner://default_inline_border_br.png) no-repeat bottom right");
 inline_style.addRule(".avgILFO_foot div", "height:5px; width:0px;"); }
try { //
create the popup box var box = doc.createElement("DIV"); if
(searchshield.quirksMode) { box.style.visibility = "hidden";
box.style.position
= "absolute"; box.style.left = "-5000px"; } box.setAttribute("id",
"XPLSS_InlineFlyover"); box.setAttribute("class", "avgILFO");
 doc.body.appendChild(box); box = null; } catch(boxErr){} };
 searchshield.SearchEngine.prototype.show_inline_ratings = function (doc,
node,
image) { var href = node.href; var anchor = node.element; if ((href ==
null) ||
(href.length < 1)) return; if (avglsinlineflyover.imageExists(anchor))
return;
 // mark search result anchor so it isn't processed repeatedly
 anchor.setAttribute("avglschecked", "1"); // get verdict
 this.display_inline(doc, anchor, href, node, false); };
 searchshield.SearchEngine.prototype.display_inline = function (doc,
anchor,
href, node, update, min_severity) { // min_severity is the lowest
severity to
display, so setting it to // 1 would not display safe icons var results =
searchshield.avgCallFunc(doc, 'MalsiteCheck', href); if (results == null)
 return; var parts = results.split('::'); // need at least severity if
(parts ==
null) return; var nSeverity = parseInt(parts[0]); if (!update &&
nSeverity ==
searchshield.XPLCHECK_RESULT_SEV_NONE) { var shortUrl =
searchshield.FilterUrl(href, searchshield.shortened_urls); if (shortUrl)
{ //
shortened url verdicts display later var engine = this;
 anchor.attachEvent("onmouseover",
function(event){avglsinlineflyover.mouseOverHandler(event, doc,
engine)});
 return; } } // severity -1 signifies sb.dat load failure if ( nSeverity
== -1 )
 nSeverity = searchshield.XPLCHECK_RESULT_SEV_NONE; //blacklist url var
blShortUrl = false; // if (nSeverity ==
searchshield.XPLCHECK_RESULT_SEV_BLOCK)
 // { // var shortUrl = searchshield.FilterUrl(href,
searchshield.shortened_urls); // if (shortUrl) // blShortUrl = true; // }
//
need xlated cat tag and category if (parts.length < 3) return; // check
the
minimum to display if ((min_severity != null) && (nSeverity <
min_severity))
 return; if (nSeverity == searchshield.XPLCHECK_RESULT_SEV_LOW &&
```

```
!this.showLowRiskVerdicts) { if (update)
 this.avg_ls_inline_hide_verdict(anchor); return; } if (nSeverity ==
searchshield.XPLCHECK_RESULT_SEV_MED && !this.showMedRiskVerdicts) { if
(update)
 this.avg_ls_inline_hide_verdict(anchor); return; } if (nSeverity ==
searchshield.XPLCHECK_RESULT_SEV_NONE && !this.showCleanVerdicts) { if
(update)
 this.avg_ls_inline_hide_verdict(anchor); return; } if (update)
 this.update_inline_image(anchor, nSeverity, parts); else
 this.add_inline_image(doc, anchor, nSeverity, parts, blShortUrl); };
 searchshield.SearchEngine.prototype.avg_ls_inline_hide_verdict =
function
(anchor) { var image = avglsinlineflyover.getImage(anchor); if (image) {
 image.style.display = "none"; if (image.parentNode &&
image.parentNode.id ==
"avg_ls_anch") image.parentNode.style.display = "none"; } };
 searchshield.SearchEngine.prototype.update_inline_image = function
(anchor,
nSeverity, aRisk) { // update the image already in the page if (anchor &&
anchor.firstChild) { var html = ''; var image = ''; if (aRisk != null &&
nSeverity != null) { var riskCategory = aRisk[1]; var riskName =
aRisk[2]; var
bgColor = searchshield.inline.color.background[nSeverity]; var
borderColor =
searchshield.inline.color.border[nSeverity]; image =
searchshield.inline.image[nSeverity]; html =
avglsinlineflyover.build(riskCategory, riskName, bgColor, borderColor); }
var
imageElem = anchor.firstChild; imageElem.src = image; if ( html &&
html.length >
0 ) { imageElem.setAttribute("title", "");
imageElem.attachEvent("onmouseover",
function(e){avglsinlineflyover.popup(e, html, nSeverity)});
 imageElem.attachEvent("onmouseout",
function(e){avglsinlineflyover.hide(e)}); }
 } }; // add the image to the page
 searchshield.SearchEngine.prototype.add_inline_image = function (doc,
anchor,
nSeverity, aRisk, blShortUrl) { if (anchor == null || anchor.parentNode
== null)
 return null; // get the proper insertion point for the image var
insertNode =
anchor.nextSibling; while ((insertNode != null) && (insertNode.tagName !=
null)
&& (insertNode.tagName == "SPAN")) { insertNode= insertNode.nextSibling;
} //
see if we already have an image anchor if ((insertNode != null) &&
 (insertNode.id != null) && (insertNode.id == "avg_ls_anch")) { return
null; }
 var html = ''; var image = searchshield.inline.clockImage; if (aRisk !=
null &&
nSeverity != null) { var riskCategory = aRisk[1]; var riskName =
aRisk[2]; var
bgColor = searchshield.inline.color.background[nSeverity]; var
borderColor =
searchshield.inline.color.border[nSeverity]; image =
searchshield.inline.image[nSeverity]; var blUrl; if (blShortUrl) { var
aRiskName
```

```
= riskName.split(':'); var sUrl = searchshield.checkUrl(aRiskName[1]);
blUrl =
{}; blUrl.riskNameLabel = aRiskName[0] + ': '; blUrl.riskCategory =
riskCategory; blUrl.bgColor = bgColor; blUrl.borderColor = borderColor;
 blUrl.sUrl = sUrl; } else { html =
avglsinlineflyover.build(riskCategory,
riskName, bgColor, borderColor); } } doc = anchor.ownerDocument; var img
=
doc.createElement("img"); img.src = image;
 img.setAttribute("id","avg_ls_image"); img.style.width = "12px";
 img.style.height = "12px"; img.style.border = "none"; img.style.padding
= "0
3px"; img.style.margin = "0"; if ((html && html.length > 0) || (blUrl !=
undefined)) { img.setAttribute("title", "");
img.attachEvent("onmouseover",
function(e){avglsinlineflyover.popup(e, html, nSeverity, blUrl)});
 img.attachEvent("onmouseout", function(e){avglsinlineflyover.hide(e)});
} //
create the link element var newAnchor = doc.createElement("A");
 newAnchor.setAttribute("id", "avg_ls_anch"); newAnchor.style.display =
"inline-block"; newAnchor.style.background = "none repeat scroll 0 0
transparent"; newAnchor.appendChild(img); img = null; // insert the node
as
either a sibling or a child if (insertNode != null)
 anchor.parentNode.insertBefore(newAnchor, insertNode); else
 anchor.parentNode.appendChild(newAnchor); return newAnchor; };
////////////////
SEARCH ENGINE //////////////// /////////////// GOOGLE SEARCH ENGINE
/////////////// searchshield.GoogleSearchEngine = function(search) {
 searchshield.SearchEngine.call(this, search); this.onlyPrimaries =
false; };
 searchshield.GoogleSearchEngine.prototype = new
searchshield.SearchEngine();
 searchshield.GoogleSearchEngine.prototype.constructor =
searchshield.GoogleSearchEngine;
searchshield.GoogleSearchEngine.prototype.name
= "google"; // the name by which the search engine is known (always
lowercase)
 searchshield.GoogleSearchEngine.prototype.validSearch = function(href) {
var
uri; if (typeof(this.search) === 'undefined' || null === this.search) uri
=
searchshield.parseLink(href); else uri = this.search.uri; if(!uri ||
!uri.host)
 return false; var hostMatch = false; var domain = uri.host; // re stitch
the
uri path and query elements to // use existing logic var path = uri.path
+
uri.delimiter + uri.query; // For Google the host must match: //
.google.com OR
 // .google.com.XX OR // .google.co.XX OR // .google.XX where XX is a
country
code // one special case is www.googe.off.ai (Anguilla) // Where any
subdomain
can come before the top level domain if (
/(\.(?:google|mozilla)\.(?:com|(?:co|off)\.[a-z]{2}|[a-
z]{2}))/i.test(domain) )
 { //check the path if ((path.indexOf("search?") == 0) ||
```

```
 (path.indexOf("sponsoredlinks?") == 0) || (path.indexOf("webhp?") == 0)
||
 (path.indexOf("webhp#") == 0) || (path.indexOf("#q=") == 0) ||
 (path.indexOf("#hl=") == 0) || (path.indexOf("#sclient=") == 0)) {
return true;
 } } return false; };
searchshield.GoogleSearchEngine.prototype.includeLink =
function(tag) { var href = ""; var outHref = false; var findStr = ""; //
check
for interstitials if (searchshield.DoesURLContain(tag.href,
this.search.uri.host)) { findStr = this.search.uri.host +
"/interstitial?"; if
(tag.className == "l" && tag.href) { if (tag.href.indexOf(findStr) != -1)
{
 findStr = "?url="; var pos = tag.href.indexOf(findStr); if (pos !== -1)
{ pos
+= 5; outHref = tag.href.substring(pos); if
(searchshield.FilterUrl(outHref,
searchshield.filter_urls)) return false; return outHref; } } } if
(tag.className
== "sla") { findStr = "/url?q="; urlPos = tag.href.indexOf(findStr); if
(urlPos
!= -1) { urlPos += 7; outHref = tag.href.substring(urlPos); return
outHref; } }
 // if an ad id if ((tag.id.indexOf("pa") == 0) || (tag.id.indexOf("an")
== 0)
|| (tag.className == "resultLink")) { var urlPos = -1; // ads now need
unescaping href = unescape(tag.href); findStr= "/url?sa="; if
(href.indexOf(findStr) != -1) { // first kind, locate real url findStr=
"&q=http"; urlPos = href.indexOf(findStr); if (urlPos != -1) urlPos += 3;
//
puts it on the http } if (urlPos == -1) { findStr = "/pagead/iclk?sa=";
if
(href.indexOf(findStr) != -1) { // second kind, locate real url findStr =
"&adurl=http"; urlPos = href.indexOf(findStr); if (urlPos != -1) urlPos
+= 7; //
puts it on the http } } if (urlPos == -1) { if (href.indexOf("/aclk?sa=")
!= -1)
 { // third kind urlPos = href.indexOf("&q=http"); if (urlPos != -1)
urlPos +=
3; // puts it on the http else { urlPos = href.indexOf("&lp=http"); if
(urlPos
!= -1) urlPos += 4; else { findStr = "&adurl=http"; urlPos =
href.indexOf(findStr); if (urlPos != -1) urlPos += 7; // puts it on the
http } }
 } } if (urlPos == -1) { if (href.indexOf("/url?cad=") != -1) { // fourth
kind
 urlPos = href.indexOf("&q=http"); if (urlPos != -1) urlPos += 3; // puts
it on
the http } } if (urlPos != -1) { outHref = href.substring(urlPos); // the
destination url is in the href string of this redirector if
(outHref.indexOf('xg4ken.com') > -1) { urlPos =
(unescape(outHref)).indexOf('url[]=') + 6; var destUrl =
(unescape(outHref)).substring(urlPos); if
(searchshield.FilterUrl(destUrl,
searchshield.filter_urls)) { var destUrl =
searchshield.getHrefFromCiteElement(tag); if (destUrl) return
```

```
searchshield.checkUrl(searchshield.removeHtmlTags(destUrl)); return
false; }
 return destUrl; } // filtered url but can get destination from href
string if
(outHref.indexOf('altfarm.mediaplex.com') > -1) { var tmpoh =
unescape(outHref);
 var destUrl = tmpoh.substring(tmpoh.indexOf('DURL=')+5); if (destUrl !=
null) {
 var destUrl = unescape(destUrl); return destUrl; } } // extract any
fragment
text, shouldn't be unescaped var pound = outHref.indexOf("#"); if (pound
!= -1)
 { var fragment = outHref.substring(pound); outHref =
outHref.substring(0,
pound); outHref = unescape(outHref); outHref += fragment; if
(searchshield.FilterUrl(outHref, searchshield.filter_urls)) return false;
return
outHref; } outHref = unescape(outHref); if (outHref.indexOf("?") == -1) {
var
ampPos = outHref.indexOf("&"); if (ampPos != -1) outHref =
outHref.substring(0,
ampPos); } if (searchshield.FilterUrl(outHref, searchshield.filter_urls))
{ var
destUrl = searchshield.getHrefFromCiteElement(tag); if (destUrl) return
searchshield.checkUrl(searchshield.removeHtmlTags(destUrl)); return
false; }
 return outHref; } } // recommended link - use following to see one //
http://www.google.cz/search?hl=cs&q=warey&btnG=Hledat&lr=lang_cs // elem
parent
class = r // href must contain - url? and q=http var parentNode =
tag.parentNode; if (parentNode && (parentNode.className.toLowerCase() ==
"r")) {
 href = tag.href; if (href && (href.indexOf("/url?") != -1)) { // locate
the
real url var urlPos = href.indexOf("q=http"); if (urlPos != -1) { urlPos
+= 2;
 outHref = href.substring(urlPos); // include entire param up to '&' var
ampPos
= outHref.indexOf("&"); if (ampPos != -1) outHref = outHref.substring(0,
ampPos); return outHref; } } } } // no link to self else if
(tag.className &&
(tag.className.charAt(0) == "l" || tag.className == "sla")) { // check
for any
images on the link if (0 === tag.getElementsByTagName("IMG").length)
return
tag.href; } // special case for ie6 results else if (searchshield.docMode
== 6)
 { var parentNodeClass = tag.parentNode ? tag.parentNode.className : '';
if
((tag.className == '') && (parentNodeClass == 'r')) { return tag.href; }
} //
else nothing return false; }; /////////////// GOOGLE ///////////////
 /////////////// YAHOO SEARCH ENGINE ///////////////
 searchshield.YahooSearchEngine = function(search) {
 searchshield.SearchEngine.call(this, search); };
 searchshield.YahooSearchEngine.prototype = new
searchshield.SearchEngine();
 searchshield.YahooSearchEngine.prototype.constructor =
```

```
searchshield.YahooSearchEngine;
searchshield.YahooSearchEngine.prototype.name =
"yahoo"; searchshield.YahooSearchEngine.prototype.validSearch =
function(href) {
 var uri; if (typeof(this.search) === 'undefined' || null ===
this.search) uri =
searchshield.parseLink(href); else uri = this.search.uri; if(!uri ||
!uri.host)
 return false; var domain = uri.host; // re stitch the uri path and query
elements to // use existing logic var path = uri.path + uri.delimiter +
uri.query; // prevent verdicts on news site until 69933 is fixed if
(domain ==
'news.search.yahoo.com') return false; // For Yahoo the host must match:
 // search.yahoo.com OR // xx.search.yahoo.com where xx is the country
code OR
 // search.yahoo.co.jp OR // for Yahoo China: one.cn.yahoo.com,
search.cn.yahoo.com or www.yahoo.cn if
((domain.match(/search\.yahoo\.co(?:m|\.jp)/i) ||
 domain.match(/(?:search|one)\.cn\.yahoo\.com/i) ||
 domain.match(/www\.yahoo\.cn/i)) && path.match(/^(?:search[;?]|s\?)/i))
{
 return true; } return false; };
 searchshield.YahooSearchEngine.prototype.includeLink = function(tag) {
var
href = ""; var outHref = ""; var findStr = ""; // yahoo likes to encode
the url
 href = unescape(tag.href); var spnsdLinks =
searchshield.getParentNodeByTagName("DIV", tag, "className"); if
((spnsdLinks)
&& (spnsdLinks.className.indexOf('ads') > -1)) // sponsored links { if
(!tag.parentNode) return false; // parse ads for em tag var baseNode; if
(spnsdLinks.className.indexOf('ads horiz') > -1) // horizontal ads
sections
 baseNode = tag.parentNode.parentNode; else baseNode = tag.parentNode; if
(!baseNode || !baseNode.lastChild || baseNode.lastChild.tagName != 'EM')
return
false; var outHref =
searchshield.removeHtmlTags(baseNode.lastChild.innerHTML);
 return outHref; } if ((tag.className.indexOf("yschttl") != -1) ||
 (tag.className.indexOf("spt") != -1)) { var da = href.indexOf("**"); var
ad =
href.indexOf("*-"); if (da != -1) outHref = href.substring(da+2); else if
(ad !=
-1) outHref = href.substring(ad+2); else outHref = href ; if
((outHref.indexOf('yahoo.com') != -1) ||
(outHref.toLowerCase().indexOf("overture.") !== -1)) return false; } else
if
((tag.tagName) && (tag.tagName === "A") && (!tag.className)) { var
tagParent =
tag.parentNode; // if anchor without className then search parentNodes if
((tagParent) && (tagParent.tagName !== "EM") &&
 (searchshield.getParentNodeByClassName("yst-web", tag, 4))) { // China
Yahoo
support outHref = href; } else if ((tagParent) && (tagParent.tagName ===
"H3")
&& (!!searchshield.getParentNodeById("WS2m",tag, 5))) { // Japan Yahoo
support
```

```
 var da = href.indexOf("**"); if (da == -1) outHref = href; else outHref
=
href.substring(da+2); } else if ((tagParent) && (tagParent.className !==
"c") &&
 (tagParent.parentNode.id !== "fpn") &&
 (!!searchshield.getParentNodeByClassName("ymc", tag, 7))) { // Korea
Yahoo
support - when not caught by yschttl var da = href.indexOf("**"); if (da
!== -1)
 outHref = href.substring(da+2); } } else if (!this.onlyPrimaries) {
findStr =
"&yargs="; var yargs = href.indexOf(findStr); if (yargs != -1) { outHref
=
href.substring(yargs+findStr.length); // check for prefix if
(outHref.indexOf("://") == -1) outHref = "http://" + outHref; // if
inside an
<I>, probably a paypal link, don't include if (tag.parentNode &&
(tag.parentNode.tagName == "I")) return false; } } // filter domains //
split
the url based on '/' var parts = !!outHref ? outHref.split('/') : null;
//
Filter out domains that match any of the search engine's names if (!parts
||
!parts[2]) return false; var domain = parts[2]; // no verdicts for links
on
yahoo.com domain if (/yahoo\.com/.test(domain)) return false; // set for
yahoo
to get parent node for image insertion var hash =
searchshield.avgCallFunc(this.doc, 'GetHash', outHref);
tag.setAttribute("id",
"xplid_" + hash); return outHref; }; /////////////// YAHOO SEARCH ENGINE
/////////////// /////////////// MSN SEARCH ENGINE ///////////////
 searchshield.MSNSearchEngine = function(search) {
 searchshield.SearchEngine.call(this, search); this.onlyPrimaries =
false; };
 searchshield.MSNSearchEngine.prototype = new
searchshield.SearchEngine();
 searchshield.MSNSearchEngine.prototype.constructor =
searchshield.MSNSearchEngine; searchshield.MSNSearchEngine.prototype.name
=
"msn"; searchshield.MSNSearchEngine.prototype.validSearch =
function(href) { var
uri; if (typeof(this.search) === 'undefined' || null === this.search) uri
=
searchshield.parseLink(href); else uri = this.search.uri; if(!uri ||
!uri.host)
 return false; var hostMatch = false; var domain = uri.host; // re stitch
the
uri path and query elements to // use existing logic var path = uri.path
+
uri.delimiter + uri.query; // For MSN the host must match: //
search.msn.com OR
 // search.live.com if (domain.indexOf("search.msn.co") > -1) { if
(domain.charAt(13) == 'm') hostMatch = true; else if ((domain.charAt(13)
== '.')
&& (domain.length == 16)) hostMatch = true; } else if
(domain.indexOf("search.live.co") > -1) { if (domain.charAt(14) == 'm')
```

```
 hostMatch = true; else if ((domain.charAt(14) == '.') && (domain.length
== 17))
 hostMatch = true; } if (hostMatch) { if (path.indexOf("results.aspx") ==
0)
 return true; } return false; };
 searchshield.MSNSearchEngine.prototype.includeLink = function(tag) { var
outHref = false; // these don't seem common from Firefox, but they are in
IE if
(searchshield.DoesURLContain(tag.href, "g.msn.co")) { var qPos =
tag.href.indexOf("?"); if (qPos != -1) { var postPart =
tag.href.substring(qPos+1); var dblAmp = postPart.indexOf("&&"); if
(dblAmp !=
-1) { outHref = postPart.substring(0, dblAmp); return outHref; } } } else
if
(searchshield.DoesURLContain(tag.href, "r.msn.co")) { var element = tag;
var
parentNode = tag.parentNode; // top links - check for a CITE var
spanElements =
element.getElementsByTagName("CITE"); if ((spanElements != null) &&
 (spanElements.length > 0)) { outHref = spanElements[0].innerHTML; //
replace
any nbsp's outHref = outHref.replace(" ", " "); // url is after the
last
space in the html, after the '-' var space_pos = outHref.lastIndexOf("
"); if
(space_pos != -1) outHref = outHref.slice(space_pos + 1); outHref =
searchshield.checkUrl(outHref); return outHref; } // side links if
(element.lastChild != null) { outHref = element.lastChild.innerHTML; if (
outHref != null) { outHref = searchshield.checkUrl(outHref); return
outHref; } }
 } else if (searchshield.DoesURLContain(tag.href, this.search.uri.host)
||
searchshield.DoesURLContain(tag.href, ".live.com") ||
searchshield.DoesURLContain(tag.href, "msn.") ||
searchshield.DoesURLContain(tag.href, "msnscache.com") ||
searchshield.DoesURLContain(tag.href, "advertising.microsoft.co") ||
searchshield.DoesURLContain(tag.href, "javascript:") ||
searchshield.DoesURLContain(tag.href, "go.microsoft.co") ||
searchshield.DoesURLContain(tag.href, "hotmail.co")) { // not a link
return
false; } else if (tag.id.toLowerCase() == "trademarks") { // don't link
the
trademark at the bottom of the page return false; } else { // include it
return
tag.href; } }; /////////////// MSN SEARCH ENGINE ///////////////
///////////////
BING SEARCH ENGINE /////////////// searchshield.BingSearchEngine =
function(search) { searchshield.SearchEngine.call(this, search); };
 searchshield.BingSearchEngine.prototype = new
searchshield.SearchEngine();
 searchshield.BingSearchEngine.prototype.constructor =
searchshield.BingSearchEngine;
searchshield.BingSearchEngine.prototype.name =
"bing"; searchshield.BingSearchEngine.prototype.validSearch =
function(href) {
 var uri; if (typeof(this.search) === 'undefined' || null ===
this.search) uri =
```

```
searchshield.parseLink(href); else uri = this.search.uri; if(!uri ||
!uri.host)
 return false; var hostMatch = false; var domain = uri.host; // re stitch
the
uri path and query elements to // use existing logic var path = uri.path
+
uri.delimiter + uri.query; // For bing the host must match: //
www.bing.com or
www.bing.net // xx.bing.com or xx.bing.net where xx is a country code
 // bing.com.xx where xx is a country code // bing.search.xxxx.net where
xxxx
may be something like daum if ((domain.indexOf("www.bing.com") !== -1) ||
 (domain.indexOf("www.bing.net") !== -1) || (domain.indexOf("bing.net")
!== -1))
 { hostMatch = true; } else if ((domain.indexOf("bing.search.") !== -1)
&&
(domain.indexOf(".net") === (domain.length-4))) { //bing.search.xxxx.net
 hostMatch = true; } else { // xx.bing.com or bing.com.xx var domainLen =
domain.length; var tldPos = domain.indexOf(".bing.com"); if (tldPos > -1)
{ if
((domainLen - tldPos) == 9) hostMatch = true; } } if (hostMatch) { var
displayStyle = (domain == 'bing.search.daum.net') ? 'inline-block' :
'inline';
 this.addAnchorStyle = { display: displayStyle }; if
(path.indexOf("search?") ==
0) return true; } return false; };
 searchshield.BingSearchEngine.prototype.includeLink = function(tag) {
var
outHref = false; if (tag.tagName == 'IMG') { //no images return false; }
if
(tag.href.charAt(0) == '/') { //no relative links return false; } if
(/trademarks/i.test(tag.id)) { // don't link the trademark at the bottom
of the
page return false; } if ((tag.parentNode) &&
(/sc_stc/i.test(tag.parentNode.id))) { // don't verdict the social sites
- our
verdict doesn't fit return false; } else if
(/vt_tl/i.test(tag.className)) { //
don't verdict the video images return false; } else if
(!!searchshield.getParentNodeByClassName("sw_t",tag,3)) { // no links in
page
header return false; } else if (searchshield.DoesURLContain(tag.href,
"r.msn.co") || searchshield.DoesURLContain(tag.href, "overture.com")) {
//
france has r.msn.co and italy has overture.com sponsored links // with
the link
in CITE element var spanElements = null; // top links - check for a CITE
 spanElements = tag.getElementsByTagName("CITE"); if ((spanElements ==
null) ||
(spanElements.length <= 0)) { if (tag.parentNode &&
tag.parentNode.parentNode)
 spanElements = tag.parentNode.parentNode.getElementsByTagName("CITE"); }
if
((spanElements != null) && (spanElements.length > 0)) { outHref =
spanElements[0].innerHTML; if (outHref != null) { // replace any nbsp's
outHref
= outHref.replace(" ", " "); // url is after the last space in the
html,
```

```
after the '-' var space_pos = outHref.lastIndexOf(" "); if (space_pos !=
-1)
 outHref = outHref.slice(space_pos + 1); outHref =
searchshield.checkUrl(outHref); // save the link return outHref; } } //
side
links if (tag.lastChild != null) { outHref = tag.lastChild.innerHTML; if
(outHref != null) { outHref = searchshield.checkUrl(outHref); // save the
link
 return outHref; } } } // no sponsored links for now else if
(searchshield.DoesURLContain(tag.href, this.search.uri.host) ||
searchshield.DoesURLContain(tag.href, ".live.com") ||
searchshield.DoesURLContain(tag.href, ".bing.com") ||
searchshield.DoesURLContain(tag.href, ".bing.net") ||
searchshield.DoesURLContain(tag.href, ".daum.net") ||
searchshield.DoesURLContain(tag.href, ".gmarket.co") ||
searchshield.DoesURLContain(tag.href, ".multimap.com") ||
searchshield.DoesURLContain(tag.href, "msn.") ||
searchshield.DoesURLContain(tag.href, "ms.ciao.") ||
searchshield.DoesURLContain(tag.href, "ms.ciao-") ||
searchshield.DoesURLContain(tag.href, "advertising.microsoft.co") ||
searchshield.DoesURLContain(tag.href, "javascript:") ||
searchshield.DoesURLContain(tag.href, "go.microsoft.co") ||
searchshield.DoesURLContain(tag.href, "hotmail.co") ||
searchshield.DoesURLContain(tag.href, "cc.bingj.com") ||
searchshield.DoesURLContain(tag.href, "microsofttranslator.com") ||
searchshield.DoesURLContain(tag.href, ".engkoo.com") ||
searchshield.DoesURLContain(tag.href, "sealinfo.verisign.com") ||
searchshield.DoesURLContain(tag.href, "explabs.com") ||
searchshield.DoesURLContain(tag.href, "onlinehelp.microsoft.com") ||
searchshield.DoesURLContain(tag.href, ".myoverture")) { // not a link
return
false; } else return tag.href; };
 searchshield.BingSearchEngine.prototype.addImage = function(node, image,
hidden) { var element = node.element; var parentNode =
node.element.parentNode;
 var grandParentNode = !!parentNode ? parentNode.parentNode : null; // if
there
are redundant links in the same grandparent then skip them if
(!!grandParentNode) { gpChildren = grandParentNode.childNodes; for (var
i=0; i <
gpChildren.length; i++) { if (!!element.href && gpChildren[i].tagName ==
"A" &&
gpChildren[i].href == element.href) return; } } var parent =
searchshield.SearchEngine.prototype.addImage; return parent.call(this,
node,
image, hidden); }; /////////////// BING SEARCH ENGINE ///////////////
 /////////////// BAIDU SEARCH ENGINE ///////////////
 searchshield.BaiduSearchEngine = function(search) {
 searchshield.SearchEngine.call(this, search); this.updateImageStyle = {
 verticalAlign: null }; }; searchshield.BaiduSearchEngine.prototype = new
searchshield.SearchEngine();
 searchshield.BaiduSearchEngine.prototype.constructor =
searchshield.BaiduSearchEngine;
searchshield.BaiduSearchEngine.prototype.name =
"baidu"; searchshield.BaiduSearchEngine.prototype.validSearch =
function(href) {
 var uri; if (typeof(this.search) === 'undefined' || null ===
this.search) uri =
```

```
searchshield.parseLink(href); else uri = this.search.uri; if(!uri ||
!uri.host)
 return false; var hostMatch = false; var domain = uri.host; // re stitch
the
uri path and query elements to // use existing logic var path = uri.path
+
uri.delimiter + uri.query; if (domain == "www.baidu.com" ||
path.indexOf("testBaidu") !== -1) hostMatch = true; if (hostMatch) { if
(path.indexOf("s?") == 0) return true; } return false; };
 searchshield.BaiduSearchEngine.prototype.includeLink = function(tag) {
if
(tag.className && tag.className == "m") return false; if
(searchshield.DoesURLContain(tag.href, this.search.uri.host)) return
false; else
 { // no link to self var traverseElement = tag.parentNode; while (
traverseElement && traverseElement.className != "tbody") { if
(traverseElement.className == "f") return tag.href; traverseElement =
traverseElement.parentNode; } return false; } }; /////////////// BAIDU
SEARCH
ENGINE /////////////// /////////////// EARTHLINK SEARCH ENGINE
///////////////
 searchshield.EarthlinkSearchEngine = function(search) {
 searchshield.SearchEngine.call(this, search); this.onlyPrimaries =
false;
 this.addAnchorStyle = { position: "static" }; };
 searchshield.EarthlinkSearchEngine.prototype = new
searchshield.SearchEngine();
 searchshield.EarthlinkSearchEngine.prototype.constructor =
searchshield.EarthlinkSearchEngine;
 searchshield.EarthlinkSearchEngine.prototype.name = "earthlink";
 searchshield.EarthlinkSearchEngine.prototype.validSearch =
function(href) { var
uri; if (typeof(this.search) === 'undefined' || null === this.search) uri
=
searchshield.parseLink(href); else uri = this.search.uri; if(!uri ||
!uri.host)
 return false; var hostMatch = false; var domain = uri.host; // re stitch
the
uri path and query elements to // use existing logic var path = uri.path
+
uri.delimiter + uri.query; // For EarthLink the host must match:
 // search.earthlink.net if (("search.earthlink.net" == domain) &&
 (path.indexOf("search?") == 0)) { return true; } return false; };
 searchshield.EarthlinkSearchEngine.prototype.includeLink = function(tag)
{ var
outHref = ""; var findStr = ""; // check for an anchor if (tag.tagName ==
"A") {
 // check for sponsored if (tag.id.indexOf("a") == 0) { var q =
tag.href.indexOf("&q="); var qlen = 3; if (q == -1) { q =
tag.href.indexOf("&adurl="); qlen = 7; if (q == -1) return false; } //
find end
of url var end = tag.href.indexOf("&", q+qlen); if (end < 0) end =
tag.href.length; // add the link outHref = tag.href.substring(q+qlen,
end); if
(searchshield.FilterUrl(outHref, searchshield.filter_urls)) return false;
return
outHref; } // don't search url's to self if ((tag.href.indexOf("://") ==
-1) ||
```

```
searchshield.DoesURLContain(tag.href, this.search.uri.host)) return
false; // if
a normal web result add it if (tag.parentNode && (tag.parentNode.tagName
==
"H3") && tag.parentNode.parentNode && (tag.parentNode.parentNode.tagName
==
"LI") && tag.parentNode.parentNode.parentNode &&
(tag.parentNode.parentNode.parentNode.tagName == "UL")) { if
(searchshield.FilterUrl(tag.href, searchshield.filter_urls)) return
false;
 return tag.href; } } return false; }; /////////////// EARTHLINK SEARCH
ENGINE
/////////////// /////////////// AOL SEARCH ENGINE ///////////////
 searchshield.AOLSearchEngine = function(search) {
 searchshield.SearchEngine.call(this, search); this.addImageStyle = {
display:
"inline" }; this.addAnchorStyle = { display: null };
this.updateImageStyle = {
 verticalAlign: null, display: "inline" }; };
 searchshield.AOLSearchEngine.prototype = new
searchshield.SearchEngine();
 searchshield.AOLSearchEngine.prototype.constructor =
searchshield.AOLSearchEngine; searchshield.AOLSearchEngine.prototype.name
=
"aol"; searchshield.AOLSearchEngine.prototype.validSearch =
function(href) { var
uri; if (typeof(this.search) === 'undefined' || null === this.search) uri
=
searchshield.parseLink(href); else uri = this.search.uri; if(!uri ||
!uri.host)
 return false; var domain = uri.host; //path may be 'aol/search' or
'search' var
pathArray = uri.path.split("/"); var aol = pathArray[0]; // re stitch the
uri
path and query elements // to use existing logic var path = (undefined ==
pathArray[1]) ? pathArray[0] : pathArray[1]; path += uri.delimiter +
uri.query;
 if (/search\.aol\.com/.test(domain)) { if ((aol == "aol") &&
 (path.indexOf("search?") == 0)) { return true; } } return false; };
 searchshield.AOLSearchEngine.prototype.includeLink = function(tag) { if
(searchshield.DoesURLContain(tag.href, this.search.uri.host)) return
false; //
sponsored links - google if ((tag.className) &&
(tag.className.indexOf("slLink
topAnchor") != -1)) { //parse for embedded href if
(tag.href.indexOf("/aclk?sa=") == -1) return false; var adurl =
tag.href.indexOf("&adurl=http"); // if an adurl the destination href can
be
acquired from the onclick handler if (adurl != -1) { var destUrl; var
clickHandler = tag.getAttribute('onclick'); if (clickHandler != null) {
// the
destination href is the 2nd parameter (zero-based array) destUrl =
searchshield.getClickHandlerParams(clickHandler)[1]; if (destUrl != null)
{
 destUrl = searchshield.removeHtmlTags(destUrl.replace(/'/g,'')); return
searchshield.checkUrl(destUrl); } } } return false; } if (tag.className
==
"find") return tag.href; return false; };
```

```
 searchshield.AOLSearchEngine.prototype.insertNodes = function(node, doc)
{ var
element = node.element; var score = node.score; if (element &&
element.className
&& element.className.indexOf("slLink") != -1) { //sponsored links only //
for
alt image if (score == searchshield.SCORE_SS_VERISIGN) return
[element.nextSibling, element.parentNode]; // for verdict image var cN =
element.childNodes; var cnLen = cN.length; for (var i=0; i < cnLen; i++)
{ if
((cN[i].nodeType == 1) && (cN[i].nodeName == 'SPAN') && ((cN[i].className
==
'title') || (cN[i].className == 'durl'))) return [cN[i].nextSibling,
cN[i].parentNode]; } } var parent =
searchshield.SearchEngine.prototype.insertNodes; return parent.call(this,
node,
doc); }; /////////////// AOL SEARCH ENGINE ///////////////
/////////////// ASK
SEARCH ENGINE /////////////// searchshield.AskSearchEngine =
function(search) {
 searchshield.SearchEngine.call(this, search); };
 searchshield.AskSearchEngine.prototype = new
searchshield.SearchEngine();
 searchshield.AskSearchEngine.prototype.constructor =
searchshield.AskSearchEngine; searchshield.AskSearchEngine.prototype.name
=
"ask"; searchshield.AskSearchEngine.prototype.validSearch =
function(href) { var
uri; if (typeof(this.search) === 'undefined' || null === this.search) uri
=
searchshield.parseLink(href); else uri = this.search.uri; if(!uri ||
!uri.host)
 return false; var domain= uri.host; // re stitch the uri path and query
elements // to use existing logic var path = uri.path + uri.delimiter +
uri.query; if ("www.ask.com" == domain) { if (path.indexOf("web?") == 0)
return
true; } return false; };
searchshield.AskSearchEngine.prototype.includeLink =
function(tag) { var outHref = ""; var findStr = ""; if
(/nu|info/i.test(tag.className)) { // exclude green links if
((tag.firstChild)
&& (tag.firstChild.className) &&
(tag.firstChild.className.indexOf('attrib') ==
0)) { return false; } // exclude sub links in tables if
(searchshield.getParentNodeByTagName("TD", tag)) { return false; } //
sponsored
ads from redirect var cN = tag.childNodes; var cnLen = cN ? cN.length :
0; for
(var i=0; i < cnLen; i++) { if ((cN[i].nodeType == 1) && (cN[i].nodeName
==
'SPAN') && (cN[i].className == 'T10')) { return
searchshield.checkUrl(searchshield.removeHtmlTags(cN[i].innerHTML)); } }
// ads
link to google with class nu findStr = "www.google.com"; if
(tag.href.indexOf(findStr) != 0) { if (tag.href.indexOf(findStr +
"/aclk?sa=")
!= -1) { findStr = "&adurl=http"; var pos = tag.href.indexOf(findStr); if
(pos
```

```
!= -1) { pos += 7; outHref = tag.href.substring(pos); outHref =
unescape(outHref); // the destination url is in the href string of this
redirector if (outHref.indexOf('xg4ken.com') > -1) { urlPos =
(unescape(outHref)).indexOf('url[]=') + 6; var destUrl =
(unescape(outHref)).substring(urlPos); if
(searchshield.FilterUrl(destUrl,
searchshield.filter_urls)) { var destUrl =
searchshield.getHrefFromCiteElement(tag); if (destUrl) return
searchshield.checkUrl(searchshield.removeHtmlTags(destUrl)); return
false; }
 return destUrl; } if (searchshield.FilterUrl(outHref,
searchshield.filter_urls)) return false; return outHref; } } } else { //
ad not
to google just use href outHref = tag.href; if
(searchshield.FilterUrl(outHref,
searchshield.filter_urls)) return false; return outHref; } } if
(searchshield.DoesURLContain(tag.href, this.search.uri.host)) { return
false; }
 // primary results have class containing title, L2 or L4 ( Wikipedia
links )
 else if (/title|L[0-9]/i.test(tag.className)) { outHref = tag.href; if
(searchshield.FilterUrl(outHref, searchshield.filter_urls)) return false;
return
outHref; } return false; };
searchshield.AskSearchEngine.prototype.insertNodes =
function(node,doc) { var element = node.element; var parentNode =
node.element.parentNode; // insert alt image if
(/XPLSS_/.test(element.id)) {
 return [null, element]; } var cN = element.getElementsByTagName('span');
for
(var i=0; i < cN.length; i++) { if
(/title|newAdFont/.test(cN[i].className)) {
 return [cN[i].nextSibling, cN[i].parentNode]; } } return [null,
parentNode]; };
 /////////////// ASK SEARCH ENGINE /////////////// ///////////////
ALTAVISTA
SEARCH ENGINE /////////////// searchshield.AltaVistaSearchEngine =
function(search) { searchshield.SearchEngine.call(this, search); };
 searchshield.AltaVistaSearchEngine.prototype = new
searchshield.SearchEngine();
 searchshield.AltaVistaSearchEngine.prototype.constructor =
searchshield.AltaVistaSearchEngine;
 searchshield.AltaVistaSearchEngine.prototype.name = "altavista";
 searchshield.AltaVistaSearchEngine.prototype.validSearch =
function(href) { var
uri; if (typeof(this.search) === 'undefined' || null === this.search) uri
=
searchshield.parseLink(href); else uri = this.search.uri; if(!uri ||
!uri.host)
 return false; var domain= uri.host; // re stitch the uri path and query
elements // to use existing logic var path = uri.path + uri.delimiter +
uri.query; // www.atlavista.com ---> now
http://us.yhs4.search.yahoo.com/yhs/search?fr=altavista&fr=altavista&itag
=ody&q=warez&kgs=1&kls=0
 // xx.altavista.com where xx is a country code var hostMatch = false; if
("www.altavista.com" == domain) { hostMatch = true; } else { //
xx.altavista.com
 var pDest = domain.indexOf(".altavista.com"); if ((pDest != -1 ) &&
```

```
((domain.length - pDest) == 14)) { hostMatch = true; } else { // prevent
verdicts on news site until 69933 is fixed if (domain ==
'news.search.yahoo.com') return false; // a reference to altavista must
be
present in yahoo search url var pRef = (path.indexOf('altavista') != -1);
pDest
= domain.indexOf("search.yahoo.com"); if (pDest != -1 && pRef) hostMatch
= true;
 } } if (hostMatch) { //path must start with web/results? if
((path.indexOf("yhs/search?") == 0) || (path.indexOf("search;") == 0) ||
 (path.indexOf("yhs/search;") == 0) || (path.indexOf("web/results?") ==
0) ||
 (path.indexOf("altavista") != -1) ) { return true; } } return false; };
 searchshield.AltaVistaSearchEngine.prototype.includeLink = function(tag)
{ var
outHref = ""; var findStr = ""; // initial checks if ((!tag.href) ||
 (tag.href.charAt(0) == '#') || (tag.href.indexOf("javascript:") == 0)) {
return
false; } if (searchshield.DoesURLContain(tag.href, this.search.uri.host))
return
false; // sponsored links var spnsdLinks =
searchshield.getParentNodeByTagName("DIV", tag, "className"); if
(spnsdLinks &&
/ads/.test(spnsdLinks.className)) { if (!tag.parentNode) return false; //
parse
ads for em tag var baseNode; // horizontal ads sections if
(/ads\shoriz/.test(spnsdLinks.className)) baseNode =
tag.parentNode.parentNode;
 else baseNode = tag.parentNode; if (!baseNode || !baseNode.lastChild ||
baseNode.lastChild.tagName != 'EM') return false; var outHref =
searchshield.removeHtmlTags(baseNode.lastChild.innerHTML); return
outHref; } if
((tag.className == "spt") || (tag.className == "res") || (tag.className
==
"yschttl spt")) { findStr = "/**"; var pos = tag.href.indexOf(findStr);
if (pos
!= -1) { pos += 3; outHref = tag.href.substring(pos); outHref =
unescape(outHref); // no results for overture.com & no yahoo domains
chkHref =
outHref.toLowerCase(); if (chkHref.indexOf("overture.") != -1) return
false; //
split the url based on '/' var parts = outHref.split('/'); // only need a
domain
 if ((parts != null) && (parts[2] != null)) { var domain = parts[2]; //
no
verdicts for links on yahoo.com domain if (/yahoo\.com/.test(domain))
return
false } return outHref; } else { if
(searchshield.DoesURLContain(tag.href,
'yahoo.com')) return false; else return tag.href; } } return false; };
 searchshield.AltaVistaSearchEngine.prototype.getImgElement = function
(element)
 { while (element != null) { element = element.nextSibling; if (element
!= null)
 { if ((element.id == null) || (element.id.indexOf("LXPLSS_") == -1)) {
// not
our id but hit another anchor no verdict if (element.tagName == "A") {
element =
```

```
null; break; } } else if (element.tagName == "A") break; } } var rtnElem
=
!!element ? element.firstChild : element; return rtnElem; };
///////////////
ALTAVISTA SEARCH ENGINE /////////////// /////////////// YANDEX SEARCH
ENGINE
/////////////// searchshield.YandexSearchEngine = function(search) {
 searchshield.SearchEngine.call(this, search); };
 searchshield.YandexSearchEngine.prototype = new
searchshield.SearchEngine();
 searchshield.YandexSearchEngine.prototype.constructor =
searchshield.YandexSearchEngine;
searchshield.YandexSearchEngine.prototype.name
= "yandex"; searchshield.YandexSearchEngine.prototype.validSearch =
function(href) { var uri; if (typeof(this.search) === 'undefined' || null
===
this.search) uri = searchshield.parseLink(href); else uri =
this.search.uri;
 if(!uri || !uri.host) return false; var domain= uri.host; // re stitch
the uri
path and query elements // to use existing logic var path = uri.path +
uri.delimiter + uri.query; if ((domain.match(/yandex\.com|by|kz|ru|ua/i))
&&
 (path.indexOf("yandsearch?") == 0)) { return true; } return false; };
 searchshield.YandexSearchEngine.prototype.includeLink = function(tag) {
if
((tag.href.charAt(0) == '/') || (tag.href.indexOf("/search") != -1)) {
return
false; } if (searchshield.DoesURLContain(tag.href, this.search.uri.host)
||
 searchshield.DoesURLContain(tag.href, "yandex.net") ||
 searchshield.DoesURLContain(tag.href, "yandex.ru") ||
 searchshield.DoesURLContain(tag.href, "moikrug.ru") ||
 searchshield.DoesURLContain(tag.href, "ya.ru") ||
 searchshield.DoesURLContain(tag.href, "yandex.com") ||
 searchshield.DoesURLContain(tag.href, "yandex.st")) { return false; }
 parentClass = tag.parentNode ? tag.parentNode.className : '';
gParentClass =
(tag.parentNode && tag.parentNode.parentNode) ?
tag.parentNode.parentNode.className : ''; if
(parentClass.match(/moreinfo/i) ||
gParentClass.match(/moreinfo/i)) { return false; } // links to alt
searches on
different engines have a classname == b-link if ((tag.className == "b-
link") ||
(tag.className == "b-serp-url__link")) { return false; } return tag.href;
};
 searchshield.YandexSearchEngine.prototype.getImgElement = function
(element) {
 while (element != null) { element = element.nextSibling; if (element !=
null) {
 if ((element.id == null) || (element.id.indexOf("LXPLSS_") == -1)) { //
not our
id but hit another anchor no verdict if (element.tagName == "A") {
element =
null; break; } } else if (element.tagName == "A") break; } } var rtnElem
=
```

```
!!element ? element.firstChild : element; return rtnElem; };
///////////////
YANDEX SEARCH ENGINE /////////////// /////////////// SEZNAM SEARCH ENGINE
/////////////// searchshield.SeznamSearchEngine = function(search) {
 searchshield.SearchEngine.call(this, search); this.elementAttribute = {
width:
"18", height: "18" }; this.updateImageStyle = { width: "18px", height:
"18px" };
 }; searchshield.SeznamSearchEngine.prototype = new
searchshield.SearchEngine();
 searchshield.SeznamSearchEngine.prototype.constructor =
searchshield.SeznamSearchEngine;
searchshield.SeznamSearchEngine.prototype.name
= "seznam"; searchshield.SeznamSearchEngine.prototype.validSearch =
function(href) { var uri; if (typeof(this.search) === 'undefined' || null
===
this.search) uri = searchshield.parseLink(href); else uri =
this.search.uri;
 if(!uri || !uri.host) return false; var domain= uri.host; // re stitch
the uri
path and query elements // to use existing logic var path = uri.path +
uri.delimiter + uri.query; if ((domain.indexOf("search.seznam.cz") > -1)
&&
 ((path.indexOf("?") == 0) || (path.indexOf("svet") == 0) ||
(path.indexOf("searchScreen") == 0))) { return true; } return false; };
 searchshield.SeznamSearchEngine.prototype.includeLink = function(tag) {
if
(/seznam/.test(tag.hostname)) return false; // no verdicts on pictures
unless
broken verdict placement is fixed if ((tag.className == 'picture') ||
(tag.className == 'pict')) return false; var parentNode = tag.parentNode;
if
(parentNode) { var grandParentNode = tag.parentNode.parentNode; if
((parentNode.tagName == "SPAN") && ((parentNode.className.toLowerCase()
==
"sklik-url") || (parentNode.className.toLowerCase() == "sklik-title"))) {
//
locate the real url and unencode it var urlPos =
tag.href.indexOf("&url=http");
 if (urlPos != -1) { urlPos += 5; outHref = tag.href.substring(urlPos);
outHref
= unescape(outHref); return outHref; } } else if (grandParentNode) { var
greatGrandParentNode = tag.parentNode.parentNode.parentNode; if
((grandParentNode.tagName == "DIV") &&
((grandParentNode.className.toLowerCase()
== "text") || (grandParentNode.className.toLowerCase() == "hlasky otz")))
{ //
standard link return tag.href; } else if (greatGrandParentNode &&
greatGrandParentNode.tagName == "DIV" &&
 (greatGrandParentNode.className.toLowerCase() == "hotlinks")) { // hint
link
 return tag.href; } } } return false; }; /////////////// SEZNAM SEARCH
ENGINE
/////////////// /////////////// WEBHLEDANI SEARCH ENGINE ///////////////
 searchshield.WebhledaniSearchEngine = function(search) {
 searchshield.SearchEngine.call(this, search); };
 searchshield.WebhledaniSearchEngine.prototype = new
searchshield.SearchEngine();
```

```
 searchshield.WebhledaniSearchEngine.prototype.constructor =
searchshield.WebhledaniSearchEngine;
 searchshield.WebhledaniSearchEngine.prototype.name = "webhledani";
 searchshield.WebhledaniSearchEngine.prototype.validSearch =
function(href) {
 var uri; if (typeof(this.search) === 'undefined' || null ===
this.search) uri =
searchshield.parseLink(href); else uri = this.search.uri; if(!uri ||
!uri.host)
 return false; var domain= uri.host; // re stitch the uri path and query
elements // to use existing logic var path = uri.path + uri.delimiter +
uri.query; if ((domain.indexOf("webhledani.cz") > -1) &&
 (path.indexOf("results.aspx?") == 0)) { return true; } return false; };
 searchshield.WebhledaniSearchEngine.prototype.includeLink =
function(tag) { if
(tag.href.indexOf("/redir.aspx?") != -1) { var ancestorNode; // sponsored
link
 if ((ancestorNode = searchshield.getParentNodeByClassName("results
sponsored",
tag, 3)) != null) { if ((ancestorNode =
searchshield.getParentNodeByClassName("res1", tag, 2)) != null) { var
spanSibling = ancestorNode.nextSibling; while (spanSibling.nodeName !=
'SPAN') {
 spanSibling = spanSibling.nextSibling; if (spanSibling.nodeName == 'DIV'
||
spanSibling.className == 'res1') { spanSibling = null; break; } } if
(spanSibling) outHref = spanSibling.innerHTML; } if (outHref) return
searchshield.checkUrl(searchshield.removeHtmlTags(outHref)); } if
((ancestorNode
= searchshield.getParentNodeByClassName("right-sponsored", tag, 3)) !=
null) {
 var outHref; if ((ancestorNode =
searchshield.getParentNodeByClassName("res3",
tag, 2)) != null) { var spanSibling = ancestorNode.nextSibling; while
(spanSibling.nodeName != 'SPAN') { spanSibling = spanSibling.nextSibling;
if
(spanSibling.nodeName == 'DIV' || spanSibling.className == 'res3') {
spanSibling
= null; break; } } if (spanSibling) outHref = spanSibling.innerHTML; } if
(outHref) return
searchshield.checkUrl(searchshield.removeHtmlTags(outHref)); }
 // result link if (searchshield.getParentNodeByClassName("results", tag,
3) !=
null) { if ((ancestorNode = searchshield.getParentNodeByClassName("res2",
tag,
2)) != null) { if (tag.parentNode && (tag.parentNode.nodeName != 'P'))
return
getLinkHref(ancestorNode); } } } return false; function
getLinkHref(aNode) { var
spanElems = aNode.getElementsByTagName('span'); for (var i=0; i <
spanElems.length; i++) { if (spanElems[i].className != 'site') continue;
var
outAnchor = spanElems[i].getElementsByTagName('a')[0]; if (outAnchor ==
null)
 outAnchor = spanElems[i]; return
searchshield.checkUrl(searchshield.removeHtmlTags(outAnchor.innerHTML));
}
```

```
 return false; } }; /////////////// WEBHLEDANI SEARCH ENGINE
///////////////
 /////////////// EBAY SEARCH ENGINE ///////////////
 searchshield.eBaySearchEngine = function(search) {
 searchshield.SearchEngine.call(this, search); this.processFrames = true;
};
 searchshield.eBaySearchEngine.prototype = new
searchshield.SearchEngine();
 searchshield.eBaySearchEngine.prototype.constructor =
searchshield.eBaySearchEngine;
searchshield.eBaySearchEngine.prototype.name =
"ebay"; searchshield.eBaySearchEngine.prototype.validSearch =
function(href) {
 var uri; if (typeof(this.search) === 'undefined' || null ===
this.search) uri =
searchshield.parseLink(href); else uri = this.search.uri; if(!uri ||
!uri.host)
 return false; var domain= uri.host; // ebay.com // shop.ebay.xx //
shop.xxxx.ebay.xx like shop.benl.ebay.be // search.auction.co.kr var path
=
uri.path + uri.delimiter + uri.query; if ((domain.indexOf("ebay.com") > -
1) ||
 (domain.indexOf("shop.ebay") > -1) || (domain.indexOf("shop.benl.ebay")
> -1))
 { if ((path.indexOf("?_from=") == 0) || (path.indexOf("i.html") > -1)) {
return
true; } } // ebay.se if ((domain.indexOf("search.eim.ebay") > -1) &&
 ((path.indexOf("?kw=") > -1) || (path.indexOf("?ev=") > -1))) { return
true; }
 if ((domain.indexOf("search.auction.co.kr") > -1 ) &&
 ((path.indexOf("?keyword=") > -1))) { // would normally set these
properties in
the constructor or in an // overriding function but they're only required
for
this domain // must downsize verdicts for proper display
this.elementAttribute =
{ width: "16", height: "16" }; this.updateImageStyle = { width: "16px",
height:
"16px" }; return true; } // latin america ebay //
listado.mercadolibre.com.xx or
listado.mercadolibre.co.xx // or listado.mercadolibre.xx or //
category.mercadolibre.com.xx or listado.mercadolibre.xx/category //
lista.mercadolivre.com.xx or lista.mercadolivre.com.xx/category if
((domain.indexOf("www.") == -1) && ((domain.indexOf(".mercadolibre.") > -
1 ) ||
(domain.indexOf(".mercadolivre.") > -1 ))) { return true; } return false;
};
 searchshield.eBaySearchEngine.prototype.includeLink = function(tag) {
var
parentNode = null; var outHref = false; if
(searchshield.DoesURLContain(tag.href, this.search.uri.host)) return
false; if (
tag.href.indexOf(".ebayrtm.com/clk?") > -1 ) { if ( tag.title == null ||
tag.title.length < 0 ) return false; // to avoid putting links on things
that
don't look like links // need to filter them - only put verdict on last
item in
```

```
group var pN = tag.parentNode; if ((pN != null) && (pN.parentNode !=
null) &&
 (pN.parentNode.nodeName == 'TD')) { // get last element node var
lastChild =
pN.parentNode.children[pN.parentNode.children.length - 1]; while
(lastChild.nodeType != 1) { lastChild = lastChild.previousSibling; } if
(pN !=
lastChild) return false; } outHref =
searchshield.removeHtmlTags(tag.title); if
(outHref.indexOf(" ") == -1) { // no spaces if (outHref.indexOf(".") > -
1) { //
at least one dot if (outHref.indexOf(this.search.uri.host) == -1) { // no
host
 if (outHref.toLowerCase().indexOf("ebay.com") == -1) { return outHref; }
} } }
 } // ebay.se - Google ads parentNode = tag.parentNode; if (parentNode &&
(parentNode.className.toLowerCase() == "google-ad-link")) { if
(tag.href.indexOf("/aclk?sa=") != -1) { findStr = "&adurl=http"; var
urlPos =
tag.href.indexOf(findStr); if (urlPos != -1) { urlPos += 7; // puts it on
the
http outHref = tag.href.substring(urlPos); // extract any fragment text,
shouldn't be unescaped var pound = outHref.indexOf("#"); if (pound != -1)
{ var
fragment = outHref.substring(pound); outHref = outHref.substring(0,
pound);
 outHref = unescape(outHref); outHref += fragment; if
(searchshield.FilterUrl(outHref, searchshield.filter_urls)) return false;
return
outHref; } outHref = unescape(outHref); if (outHref.indexOf("?") == -1) {
var
ampPos = outHref.indexOf("&"); if (ampPos != -1) outHref =
outHref.substring(0,
ampPos); } if (searchshield.FilterUrl(outHref, searchshield.filter_urls))
return
false; return outHref; } } } // auction.co.kr parentNode =
tag.parentNode; if
(parentNode && (parentNode.className.toLowerCase() == "link")) { if
(tag.href.indexOf("adcr.naver.com") > -1) { outHref = tag.innerText; if
((outHref == null) || (outHref.length < 0)) return false; return outHref;
} } //
latin america ebay if ( tag.href.toLowerCase().indexOf("clickcounter?") >
-1 ) {
 var spanElements = tag.getElementsByTagName("span"); if ((spanElements
!=
null) && (spanElements.length > 0)) { if (
spanElements[0].className.toLowerCase() == "mclics-url" ) { outHref =
spanElements[0].innerHTML; if ( outHref == null || outHref.length < 0 )
return
false; outHref = searchshield.removeHtmlTags(outHref); if
(outHref.indexOf(" ")
== -1) { // no spaces if (outHref.indexOf(".") > -1) { // at least one
dot if
((outHref.toLowerCase().indexOf("mercadolibre") == -1) &&
 (outHref.toLowerCase().indexOf("mercadolivre") == -1)) { return outHref;
} } }
 } } } return false; }; /////////////// EBAY SEARCH ENGINE
///////////////
```

```
 /////////////// DIGG SEARCH ENGINE ///////////////
 searchshield.DiggSearchEngine = function(search) {
 searchshield.SearchEngine.call(this, search); this.new_links = false;
 this.addAnchorStyle = { background: "none transparent scroll repeat 0 0"
}; };
 searchshield.DiggSearchEngine.prototype = new
searchshield.SearchEngine();
 searchshield.DiggSearchEngine.prototype.constructor =
searchshield.DiggSearchEngine;
searchshield.DiggSearchEngine.prototype.name =
"digg"; searchshield.DiggSearchEngine.prototype.validSearch =
function(href) {
 var uri; if (typeof(this.search) === 'undefined' || null ===
this.search) uri =
searchshield.parseLink(href); else uri = this.search.uri; if(!uri ||
!uri.host)
 return false; var domain= uri.host; // re stitch the uri path and query
elements // to use existing logic var path = uri.path + uri.delimiter +
uri.query; if ((domain.indexOf("digg.com") > -1) && (path.indexOf("/ad")
== -1))
 { return true; } return false; };
 searchshield.DiggSearchEngine.prototype.includeLink = function(tag) {
var
outHref = false; var findStr = ""; if ((tag.parentNode) &&
 (tag.parentNode.className.toLowerCase() == "digg-count")) { return
false; } if
(searchshield.DoesURLContain(tag.href, this.search.uri.host)) { if
(tag.className && tag.className.indexOf('source') != -1) { findStr =
"/search?q=site:"; var urlPos = tag.href.indexOf(findStr); if (urlPos !=
-1) {
 urlPos += 15; outHref = tag.href.substring(urlPos); return outHref; } }
}
 return false; }; searchshield.DiggSearchEngine.prototype.insertNodes =
function(node,doc) { var element = node.element; var parentNode =
node.element.parentNode; // insert alt image if
(/XPLSS_/.test(element.id)) {
 return [null, element]; } var prevSibling = parentNode.previousSibling;
while
((prevSibling != null) && (prevSibling.className != 'story-item-title'))
{
 prevSibling = prevSibling.previousSibling; } if (prevSibling) parentNode
=
prevSibling; return [prevSibling.lastChild, parentNode]; };
/////////////// DIGG
SEARCH ENGINE /////////////// /////////////// SLASHDOT SEARCH ENGINE
/////////////// searchshield.SlashdotSearchEngine = function(search) {
 searchshield.SearchEngine.call(this, search); this.showCleanVerdicts =
false;
 this.new_links = false; this.type = 'inline'; };
 searchshield.SlashdotSearchEngine.prototype = new
searchshield.SearchEngine();
 searchshield.SlashdotSearchEngine.prototype.constructor =
searchshield.SlashdotSearchEngine;
 searchshield.SlashdotSearchEngine.prototype.name = "slashdot";
 searchshield.SlashdotSearchEngine.prototype.validSearch = function(href)
{ var
uri; if (typeof(this.search) === 'undefined' || null === this.search) uri
=
```

```
searchshield.parseLink(href); else uri = this.search.uri; if(!uri ||
!uri.host)
 return false; var domain= uri.host; // re stitch the uri path and query
elements // to use existing logic var path = uri.path + uri.delimiter +
uri.query; if (domain.indexOf("slashdot.org") != -1) { return true; }
return
false; }; searchshield.SlashdotSearchEngine.prototype.includeLink =
function(tag) { if (searchshield.DoesURLContain(tag.href,
this.search.uri.host))
 return false; if ( tag.href.indexOf("mailto:") != -1 ) return false; if
(
tag.href.indexOf("slashdot.org") != -1 ) return false; if (tag.parentNode
&&
(tag.parentNode.tagName == "DIV")) { if (tag.parentNode.id.indexOf("text-
") !=
-1) { return tag.href; } } else if ( tag.parentNode &&
tag.parentNode.parentNode
&& tag.parentNode.parentNode.tagName == "DIV" ) { if (
tag.parentNode.parentNode.id.indexOf("text-") != -1 ) { return tag.href;
} }
 return false; }; searchshield.SlashdotSearchEngine.prototype.addImage =
function(node, image, hidden) { var doc = this.search.doc;
 this.init_inline_ratings(doc); this.show_inline_ratings(doc, node,
image); };
 /////////////// SLASHDOT SEARCH ENGINE /////////////// ///////////////
TWITTER
SEARCH ENGINE /////////////// searchshield.TwitterSearchEngine =
function(search) { searchshield.SearchEngine.call(this, search);
 this.showCleanVerdicts = false; this.new_links = false; this.type =
'inline';
 }; searchshield.TwitterSearchEngine.prototype = new
searchshield.SearchEngine();
 searchshield.TwitterSearchEngine.prototype.constructor =
searchshield.TwitterSearchEngine;
 searchshield.TwitterSearchEngine.prototype.name = "twitter";
 searchshield.TwitterSearchEngine.prototype.twitter_filter_urls =
["twitpic.com", "twitterfeed.com", "twitter.peoplebrowsr.com"];
 searchshield.TwitterSearchEngine.prototype.validSearch = function(href)
{ var
uri; if (typeof(this.search) === 'undefined' || null === this.search) uri
=
searchshield.parseLink(href); else uri = this.search.uri; if(!uri ||
!uri.host)
 return false; var domain= uri.host; if (domain.indexOf("twitter.com") !=
-1) {
 return true; } return false; };
 searchshield.TwitterSearchEngine.prototype.includeLink = function(tag) {
if
(tag.className == 'twitter-timeline-link') { // can't pass the tag's href
if
domain is t.co cause then new posts // will not get an immediate verdict
even if
it has a dangerous link if (tag.href.indexOf('/t.co/') != -1) { // the
tag's
inner text may be truncated and end in the // unicode suspension
character
(i.e., ...) and it cannot // be used. if
```

```
(tag.innerText.charCodeAt(tag.innerText.length-1) == 8230) { // older
posts may
not always have data-expanded-url var destUrl =
tag.getAttribute('data-expanded-url'); if (!!destUrl) return destUrl; var
finalUrl = searchshield.avgCallFunc(document, 'GetFinalUrl', tag.href);
return
finalUrl; } return searchshield.checkUrl(tag.innerText); } return
tag.href; }
 return false; }; searchshield.TwitterSearchEngine.prototype.addImage =
function(node, image, hidden) { var doc = this.search.doc;
 this.init_inline_ratings(doc); this.show_inline_ratings(doc, node,
image); };
 /////////////// TWITTER SEARCH ENGINE /////////////// ///////////////
GMAIL
SEARCH ENGINE /////////////// searchshield.GMailSearchEngine =
function(search)
 { searchshield.SearchEngine.call(this, search); this.processFrames =
true;
 this.showCleanVerdicts = false; this.type = 'inline'; };
 searchshield.GMailSearchEngine.prototype = new
searchshield.SearchEngine();
 searchshield.GMailSearchEngine.prototype.constructor =
searchshield.GMailSearchEngine;
searchshield.GMailSearchEngine.prototype.name =
"gmail"; // the name by which the search engine is known (always
lowercase)
 searchshield.GMailSearchEngine.prototype.validSearch = function(href) {
var
uri; if (typeof(this.search) === 'undefined' || null === this.search) uri
=
searchshield.parseLink(href); else uri = this.search.uri; if(!uri ||
!uri.host)
 return false; var domain = uri.host; // re stitch the uri path and query
elements to // use existing logic var path = uri.path + uri.delimiter +
uri.query; if ((domain.indexOf("mail.google.") != -1) ||
 (domain.indexOf("gmail.") != -1)) { return true; } return false; };
 searchshield.GMailSearchEngine.prototype.includeLink = function(tag) {
if
(searchshield.DoesURLContain(tag.href, this.search.uri.host)) return
false; //
don't mark anything but http:// if (tag.href.indexOf("mailto") == 0)
return
false; var parentDiv = searchshield.getParentNodeByTagName("DIV", tag,
"className"); if ((parentDiv) && ((parentDiv.className == "ii gt") || //
mail
message body (parentDiv.className == "mv") || // top sponsored links
(parentDiv.className == "vb") || // right sponsored links
(parentDiv.className
== "im") || // inside quotes (parentDiv.className == "gmail_quote") || //
quote
(parentDiv.className == "msg") // basic html mode )) // right ads { //
parse for
any ads var newHref = this.parseAdUrl(tag.href); return newHref; } return
false;
 }; searchshield.GMailSearchEngine.prototype.addImage = function(node,
image,
hidden) { var doc = this.search.doc; this.init_inline_ratings(doc);
 this.show_inline_ratings(doc, node, image); };
```

```
 searchshield.GMailSearchEngine.prototype.parseAdUrl = function(href) {
if
(!href) return href; // check for google ad var regex =
"^http(s)?\:\/\/([a-zA-Z0-9]+)\.googlesyndication\.com.+\&adurl\=(.+)";
var re =
new RegExp(regex); var matches = href.match(re); if (matches &&
(matches.length
>= 4)) { // else we want the fourth value var match = matches[3]; if
(match &&
(match.length > 0)) return match } return href; }; /////////////// GMAIL
SEARCH
ENGINE /////////////// /////////////// FACEBOOK SEARCH ENGINE
///////////////
 searchshield.FacebookSearchEngine = function(search) {
 searchshield.SearchEngine.call(this, search); this.showCleanVerdicts =
false;
 this.type = 'inline'; }; searchshield.FacebookSearchEngine.prototype =
new
searchshield.SearchEngine();
 searchshield.FacebookSearchEngine.prototype.constructor =
searchshield.FacebookSearchEngine;
 searchshield.FacebookSearchEngine.prototype.name = "facebook"; // the
name by
which the search engine is known (always lowercase)
 searchshield.FacebookSearchEngine.prototype.validSearch = function(href)
{ var
uri; if (typeof(this.search) === 'undefined' || null === this.search) uri
=
searchshield.parseLink(href); else uri = this.search.uri; if(!uri ||
!uri.host)
 return false; if (uri.host.indexOf("www.facebook.com") != -1) return
true;
 return false; }; searchshield.FacebookSearchEngine.prototype.includeLink
=
function(tag) { if (searchshield.DoesURLContain(tag.href,
this.search.uri.host)
|| searchshield.DoesURLContain(tag.href, 'ak.fbcdn.net')) { return false;
}
 return tag.href }; searchshield.FacebookSearchEngine.prototype.addImage
=
function(node, image, hidden) { var doc = this.search.doc;
 this.init_inline_ratings(doc); this.show_inline_ratings(doc, node,
image); };
 /////////////// FACEBOOK SEARCH ENGINE /////////////// ///////////////
MYSPACE
SEARCH ENGINE /////////////// searchshield.MySpaceSearchEngine =
function(search) { searchshield.SearchEngine.call(this, search);
 this.showCleanVerdicts = false; this.type = 'inline'; };
 searchshield.MySpaceSearchEngine.prototype = new
searchshield.SearchEngine();
 searchshield.MySpaceSearchEngine.prototype.constructor =
searchshield.MySpaceSearchEngine;
 searchshield.MySpaceSearchEngine.prototype.name = "myspace"; // the name
by
which the search engine is known (always lowercase)
 searchshield.MySpaceSearchEngine.prototype.validSearch = function(href)
{ var
```

```
uri; if (typeof(this.search) === 'undefined' || null === this.search) uri
=
searchshield.parseLink(href); else uri = this.search.uri; if(!uri ||
!uri.host)
 return false; if (uri.host.indexOf("www.myspace.com") != -1) return
true;
 return false; }; searchshield.MySpaceSearchEngine.prototype.includeLink
=
function(tag) { if (searchshield.DoesURLContain(tag.href,
this.search.uri.host)
) return false; if (tag.href.match(/\.msplinks\.com/i)) return
searchshield.checkUrl(searchshield.removeHtmlTags(tag.innerHTML)); return
tag.href }; searchshield.MySpaceSearchEngine.prototype.addImage =
function(node,
image, hidden) { var doc = this.search.doc;
this.init_inline_ratings(doc);
 this.show_inline_ratings(doc, node, image); }; /////////////// MYSPACE
SEARCH
ENGINE /////////////// /////////////// AVG TOOLBAR REPORTING
/////////////// var
avgreport = { BLOCK_SEVERITY: 3, scanResult: function (doc, url,
foundUrl, ip) {
 // report scan end only if surf enabled if ('1' !=
searchshield.avgCallFunc(doc, 'GetSurfEnabled')) return; // for now
native is
handling the interstitial var results = searchshield.avgCallFunc(doc,
'MalsiteCheck', url); if ( results == null ) return; var parts =
results.split('::'); // need at least severity if (parts == null) return;
if
(!ip) ip = ""; // use a block severity if foundUrl is given var severity
=
!!foundUrl ? avgreport.BLOCK_SEVERITY : parseInt(parts[0], 10); var
category =
""; var threat = ""; // fill in the category and threat if something was
found
 if (severity > 0) { category = parts[2]; threat = parts[4]; } // else
 // return; var scan_result = searchshield.avgCallFunc(doc,
'ReportScanResult',
url, url, threat, category, ip, severity); var scan_end =
avgreport.scanEnd(doc,
url); return (scan_result && scan_end); }, scanEnd: function (doc, url) {
return
searchshield.avgCallFunc(doc, 'ReportScanEnd', url); },
GetInterstitialIP:
function (interstitial) { // simple regex to pull the IP address var
regex =
/(([1-9][0-9]{0,2})|0)\.(([1-9][0-9]{0,2})|0)\.(([1-9][0-
9]{0,2})|0)\.(([1-9][0-9]{0,2})|0)/;
 var match = regex.exec(interstitial); if (!match) return ''; return
match[0]; }
 } /////////////// AVG TOOLBAR REPORTING /////////////// ///////////////
FLYOVERS /////////////// var avglsflyover = { count: 0, poppedUp: false,
 poppedElement: null, reset: function () { avglsflyover.count = 0;
 avglsflyover.poppedUp = false; avglsflyover.poppedElement = null; },
popup:
function (event, hash, search, flyover) { if (!event) event =
window.event; var
```

```
div = document.getElementById("XPLSS_Flyover"); if (div == null) return;
//
establish target element and get its containing document object // in
case
verdict is inside a frame var eventTarget = event.srcElement; var
frameDoc =
eventTarget.ownerDocument; // if the element is the clock, don't pop over
it if
((eventTarget.src != null) && (eventTarget.src.indexOf("clock.gif") != -
1))
 return; // save the element we popped over avglsflyover.poppedElement =
eventTarget; // if no flyover get it if ((flyover == null) || (flyover ==
"")) {
 flyover = searchshield.avgCallFunc(frameDoc, 'BuildFlyover', hash); if
(!flyover) return; // cleanup flyover, replace any new lines flyover =
flyover.replace(/\r/g, ""); flyover = flyover.replace(/\n/g, ""); //
escape any
single quotes flyover = flyover.replace(/'/g, "&#39;"); } // set the html
in the
layer div.innerHTML = flyover; // needed to prevent the flyover from
hiding
inadvertantly in IE7 if (searchshield.docMode == 7) {
div.style.backgroundColor
= "#fff"; } // there is an unwanted text node that causes vertical
misalignment
of flyover if (div.firstChild.nodeType == 3)
div.removeChild(div.firstChild);
 avglsflyover.poppedUp = true; // reset display count avglsflyover.count
= 0;
 avglsflyover.position(); }, hide: function (event) { var frameDoc =
document ||
top.document; var div = frameDoc.getElementById("XPLSS_Flyover"); if
((div ==
null) || (div.style == null) || (div.style.visibility == "hidden"))
return; var
trans_div = frameDoc.getElementById("XPLSS_Trans"); if ((trans_div ==
null) ||
(trans_div.style == null) || (trans_div.style.visibility == "hidden"))
return;
 // scroll and keydown events will pass a null event by design //
toElement will
be null when mousing out of frameelement containing a verdict if (event
== null
|| event.toElement == null) { return; } if
(trans_div.contains(event.toElement)
|| div.contains(event.toElement)) return; // if the toElement is a
cooresponding
alt image then don't hide // use try/catch because toElement will be null
when
mousing out of frame try { if (!!event.toElement.id &&
!!event.srcElement.id) {
 if
(event.toElement.id.indexOf(event.srcElement.id.substring(0,event.srcElem
ent.id.length-2))i
== 0) return; if (event.srcElement.id == 'XPLSS_Trans' &&
/XPLSS_\d+VU\d/.test(event.toElement.id)) return; } } catch(err){} //
hide and
```

```
move somewhere off screen (negative offsets) div.style.visibility =
"hidden";
 div.style.left = "-2100px"; div.style.top = "-2100px";
 trans_div.style.visibility = "hidden"; trans_div.style.left = "-2100px";
 trans_div.style.top = "-2100px"; return true; }, position: function () {
if
(!avglsflyover.poppedUp || (avglsflyover.poppedElement == null)) return;
var
flyover = document.getElementById("XPLSS_Flyover"); if (flyover == null)
return;
 // relative position of flyover in relation to icon var locateX = 0; //
0=left,
1=right var locateY = 0; // 0=above, 1=below, 2=beside icon // get window
sizes
 var winSize = searchshield.viewPortSize(); var windowX = winSize[0]; var
windowY = winSize[1]; // get the exact size of the flyover var
flyoverSize =
searchshield.elementSize(flyover); var flyoverX = flyoverSize[0]; var
flyoverY =
flyoverSize[1]; var verdictWidth = 0; if (avglsflyover.poppedElement &&
avglsflyover.poppedElement.width) verdictWidth =
avglsflyover.poppedElement.width; // get the bounding rect for image(s)
var
imgRect = searchshield.GetFullBoundingRect(avglsflyover.poppedElement);
// half
width/height of element bounding rect var halfX = (imgRect.right -
imgRect.left)
/ 2; var halfY = (imgRect.bottom- imgRect.top) / 2; // element the mouse
is
over, get the center position var posX =
searchshield.offsetLeft(avglsflyover.poppedElement) + halfX; var posY =
searchshield.offsetTop(avglsflyover.poppedElement) + halfY; // if a
verdict is
inside a frame must get offsets for the frame element var docFrames =
document.frames; if (docFrames) { for (var i=0; i < docFrames.length;
i++) { try
{ var frameElem = docFrames[i].frameElement; if
(frameElem.contentWindow.document.getElementById(avglsflyover.poppedEleme
nt.id))
 { posX += searchshield.offsetLeft(frameElem); posY +=
searchshield.offsetTop(frameElem); break; } } catch(frmErr){} } } var
transXOffset = 0; if (imgRect.mid == undefined) transXOffset = -1 *
halfX; else
 transXOffset = ((imgRect.right + imgRect.left) / 2) - imgRect.mid; //
normalize
pos to 0 -- get amount of scrolling in browser window var scroll =
searchshield.scrollSize(); var pageOffsetX = scroll[0]; var pageOffsetY =
scroll[1]; posX -= pageOffsetX; posY -= pageOffsetY; // setup the offsets
var
offsetX = posX; var offsetY = posY; // calc where to display on page if
((windowX - posX) > posX) { // right offsetX += halfX; locateX = 1; }
else {
 //left offsetX -= (flyoverX + halfX); } if ((windowY - posY) > posY) {
// below
 if (posY < (windowY/4)) { offsetY -= halfY; locateY = 1; } else {
offsetY -=
(flyoverY / 2); locateY = 2; } } else { // above if ((windowY - posY) <
```

```
(windowY/4)) { offsetY -= (flyoverY - halfY); } else { offsetY -=
(flyoverY /
2); locateY = 2; } } // make sure we aren't off the screen if (offsetY <
0)
 offsetY = 0; if ((offsetY + flyoverY) > windowY) offsetY = windowY -
flyoverY;
 // add page offsets back offsetX += pageOffsetX; offsetY += pageOffsetY;
posX
+= pageOffsetX; posY += pageOffsetY; var paddedOffsetX = 0; //provide
space
between icon and flyover var padX = 3; if (locateX == 0) paddedOffsetX =
offsetX
- padX; else paddedOffsetX = offsetX + padX; // set where to put the
flyover
 flyover.style.top = offsetY + "px"; flyover.style.left = paddedOffsetX +
"px";
 // set where to put the transparent layer var trans =
document.getElementById("XPLSS_Trans"); if (trans != null) { var
trans_left = 0;
 var trans_top = 0; var trans_width= 0; var trans_height = 0; //
transparent
layer should overlap verdict image if (locateX == 0) trans_left = posX -
flyoverX - halfX; // left else trans_left = posX - transXOffset -
verdictWidth;
// right trans.style.left = trans_left + "px"; trans.style.top = offsetY
+ "px";
 trans.style.width = flyoverX + verdictWidth + "px"; trans.style.height =
flyoverY + "px"; } avglsflyover.display(); }, display: function () {
 avglsflyover.count++; if (avglsflyover.count == 1) { var flyover =
document.getElementById("XPLSS_Flyover"); if (flyover == null) return; //
show
the flyover, must use a little count to tell, crazy stuff
 flyover.style.visibility = "visible"; flyover.onmouseout = function(){
 avglsflyover.hide(event); }; // show the transparent layer var trans_div
=
document.getElementById("XPLSS_Trans"); if (trans_div == null) return;
 trans_div.style.visibility = "visible"; trans_div.onmouseout =
function(){
 avglsflyover.hide(event); }; avglsflyover.poppedUp = false; } }, show:
function
() { var div = document.getElementById("XPLSS_Flyover"); if (div == null)
 return; div.style.visibility = "visible"; var trans_div =
document.getElementById("XPLSS_Trans"); if (trans_div == null) return;
 trans_div.style.visible = "visible"; } }; var avglsinlineflyover = {
build:
function (riskCategory, riskName, bgColor, borderColor) { // truncate
very long
risk names if (riskName.length > 60) riskName = riskName.slice(0,55) +
String.fromCharCode(8230); var html = ''; html += '<div
class="avgILFO_head"><div></div></div><div class="avgILFO_content">';
html +=
'<img src=linkscanner://LS_Logo_Results.gif /><br />'; html +=
riskCategory +
'<br />'; html += riskName + '<br />'; html += '</div><div
class="avgILFO_foot"><div></div></div>'; return html; }, popup: function
(event,
flyover, nSeverity, blUrl) { //set verdict info var div =
```

```
document.getElementById('XPLSS_InlineFlyover'); if (div == null) return;
//
blUrl is an object for a blacklisted short url // when it is passed get
the
final url if (blUrl != undefined) { var finalUrl =
searchshield.avgCallFunc(document, 'GetFinalUrl', blUrl.sUrl); var
riskUrl =
blUrl.sUrl; if ((finalUrl) && (searchshield.FilterUrl(finalUrl,
searchshield.shortened_urls))) finalUrl =
avglsinlineflyover.getUrlFromQueryString(finalUrl); var riskName =
blUrl.riskNameLabel + finalUrl; flyover =
avglsinlineflyover.build(blUrl.riskCategory, riskName, blUrl.bgColor,
blUrl.borderColor); } div.innerHTML = flyover; div.style.width = "auto";
//reset
width div.style.position = "absolute"; if (searchshield.quirksMode ||
searchshield.docMode <= 7) { var className =
searchshield.inline.color.classname[nSeverity]; var imgBase =
"linkscanner://" +
className + "_inline_border_"; var divWidth =
searchshield.elementSize(div)[0];
 //round up to nearest 10 to avoid intentional wrapping in div var
flyoverWidth
= divWidth + (10 - Math.ceil(10*((divWidth/10) -
Math.floor(divWidth/10)))); var
ilfoDivs = div.getElementsByTagName("div"); if (ilfoDivs &&
ilfoDivs.length ==
5) { //div.style.fontSize = "10px"; div.style.backgroundImage = "url(" +
imgBase
+ "tl.png)"; div.style.backgroundPosition = "0 0";
div.style.backgroundRepeat =
"no-repeat"; div.style.width = flyoverWidth + "px"; div.style.zIndex =
"9999";
 // avgILFO_head ilfoDivs[0].style.backgroundImage = "url(" + imgBase +
"tr.png)"; ilfoDivs[0].style.backgroundPosition = "top right";
 ilfoDivs[0].style.backgroundRepeat = "no-repeat";
ilfoDivs[0].style.width =
flyoverWidth + "px"; ilfoDivs[0].style.height = "5px"; // avgILFO_head
div
 ilfoDivs[1].style.height = "5px"; // avgILFO_content
 ilfoDivs[2].style.backgroundImage = "url(" + imgBase + "r.png)";
 ilfoDivs[2].style.backgroundPosition = "top right";
 ilfoDivs[2].style.backgroundRepeat = "repeat-y";
ilfoDivs[2].style.fontSize =
"10px"; ilfoDivs[2].style.color = "black"; ilfoDivs[2].style.padding =
"0px
10px"; ilfoDivs[2].style.textAlign = "left"; ilfoDivs[2].style.wordWrap =
"break-word"; ilfoDivs[2].style.lineHeight = "130%"; // avgILFO_foot
 ilfoDivs[3].style.backgroundImage = "url(" + imgBase + "bl.png)";
 ilfoDivs[3].style.backgroundPosition = "bottom left";
 ilfoDivs[3].style.backgroundRepeat = "no-repeat";
ilfoDivs[3].style.height =
"5px"; // avgILFO_foot div ilfoDivs[4].style.backgroundImage = "url(" +
imgBase
+ "br.png)"; ilfoDivs[4].style.backgroundPosition = "bottom right";
 ilfoDivs[4].style.backgroundRepeat = "no-repeat";
ilfoDivs[4].style.width =
flyoverWidth + "px"; ilfoDivs[4].style.height = "5px"; } else {
 div.style.fontSize = "10px"; div.style.backgroundColor =
```

```
searchshield.inline.color.background[nSeverity]; div.style.border =
searchshield.inline.color.border[nSeverity] + " solid 3px";
div.style.padding =
"3px 8px"; } } else { // apply updated styles for new flyover content var
sheets
= document.styleSheets; for (var i=0; i < sheets.length; i++) { if
(sheets[i].id
&& sheets[i].id == "avgILFOStyle") { var avgILFOStyle = sheets[i]; break;
} } if
(typeof(avgILFOStyle) !== "undefined") { var className =
searchshield.inline.color.classname[nSeverity]; var divWidth =
searchshield.elementSize(div)[0]; //round up to nearest 10 to avoid
intentional
wrapping in div var flyoverWidth = divWidth + (10 -
Math.ceil(10*((divWidth/10)
- Math.floor(divWidth/10)))); var rules = avgILFOStyle.rules; for (var
i=0; i <
rules.length; i++) { var bgImg = rules[i].style.backgroundImage; var
selText =
rules[i].selectorText.toLowerCase(); if (bgImg) { var bgImgStyle =
bgImg.replace(/default/, className); rules[i].style.backgroundImage =
bgImgStyle; } if (((selText == ".avgilfo") || (selText ==
".avgilfo_head") ||
 (selText == ".avgilfo_foot div")) && (rules[i].style.width) &&
(rules[i].style.width == "0px")) { rules[i].style.width = (flyoverWidth)
+ "px";
 } } } } if (!event) event = window.event; var eventTarget =
event.srcElement;
 avglsinlineflyover.position(eventTarget); }, hide: function (event) { if
(!event) event = window.event; var div =
document.getElementById("XPLSS_InlineFlyover"); if (div == null) return;
 div.style.visibility = "hidden"; //invisible div.style.left = "-5000px";
if
(!searchshield.quirksMode) { // reset flyover styles var sheets =
document.styleSheets; for (var i=0; i < sheets.length; i++) { if
(sheets[i].id
&& sheets[i].id == "avgILFOStyle") { var avgILFOStyle = sheets[i]; break;
} } if
(typeof(avgILFOStyle) !== "undefined") { var rules = avgILFOStyle.rules;
for
(var i=0; i < rules.length; i++) { var bgImg =
rules[i].style.backgroundImage;
 var selText = rules[i].selectorText.toLowerCase(); if (bgImg) { var
bgImgStyle
= bgImg.replace(/:\/\/([a-z]+)_/i, '://default_');
 rules[i].style.backgroundImage = bgImgStyle; } if (((selText ==
".avgilfo") ||
 (selText == ".avgilfo_head") || (selText == ".avgilfo_foot div")) &&
 (rules[i].style.width)) { rules[i].style.width = "0px"; } } } } },
position:
function (imageElem) { var flyover =
document.getElementById('XPLSS_InlineFlyover'); if (flyover == null)
return; //
relative position of flyover in relation to icon var locateX = 0; //
0=left,
1=right var locateY = 0; // 0=above, 1=below, 2=beside icon // get window
sizes
 var winSize = searchshield.viewPortSize(); var windowX = winSize[0]; var
```

```
windowY = winSize[1]; // Must know if there is a horizontal scroll bar
for
Firefox // for proper flyover positioning near bottom edge var
scrollYWidth =
winSize[2]; var scrollBarX = winSize[2] > 0 ? true : false; // get the
exact
size of the flyover var flyoverSize = searchshield.elementSize(flyover);
var
flyoverX = flyoverSize[0]; var flyoverY = flyoverSize[1];
flyover.style.width =
flyoverX + "px"; // get the bounding rect for image(s) var imgRect =
imageElem.getBoundingClientRect(); // half width/height (center) of
element
bounding rect var halfX = (imgRect.right - imgRect.left) / 2; var halfY =
(imgRect.bottom- imgRect.top) / 2; // element the mouse is over, get the
center
position var posX = searchshield.offsetLeft(imageElem) + halfX; var posY
=
searchshield.offsetTop(imageElem) + halfY; var pageOffsetX = 0; var
pageOffsetY
= 0; var hasParentFrame = false; // normalize pos to 0 -- get amount of
scrolling in browser window var scroll =
searchshield.scrollSize(imageElem);
 pageOffsetX = scroll[0]; pageOffsetY = scroll[1]; hasParentFrame =
scroll[2];
 posX -= pageOffsetX; posY -= pageOffsetY; //compensate for Firefox 3 if
(posX <
imgRect.left) posX = imgRect.left+halfX; // setup the offsets var offsetX
=
posX; var offsetY = posY; // calc where to display on page if ((windowX -
posX)
> posX) { // right //offsetX += halfX; offsetX = imgRect.right + 3;
locateX = 1;
 } else { //left //offsetX -= (flyoverX + halfX); offsetX = imgRect.left
-
flyoverX - 3; } if ((windowY - posY) > posY) { // below if (posY <
(windowY/4))
 { offsetY -= halfY; locateY = 1; } else { offsetY -= (flyoverY / 2) -
halfY;
 locateY = 2; } } else { // above if ((windowY - posY) < (windowY/4)) {
offsetY
-= (flyoverY - halfY); } else { offsetY -= (flyoverY / 2) + halfY;
locateY = 2;
 } } // make sure we aren't off the screen if (offsetY < 0) offsetY = 0;
if
((offsetY + flyoverY) > windowY) { offsetY = windowY - flyoverY; } else
if
(scrollBarX && ((windowY - (posY + halfY)) < scrollYWidth)) { //verdict
overlaps
the horizontal scrollbar offsetY = windowY - (flyoverY + scrollYWidth); }
// add
page offsets back - if not in frame if (!hasParentFrame) { offsetX +=
pageOffsetX; offsetY += pageOffsetY; } //posX += pageOffsetX; //posY +=
pageOffsetY; //var paddedOffsetX = 0; //provide space between icon and
flyover
 //var padX = 3; //if (locateX == 0) // paddedOffsetX = offsetX - padX;
//else
 // paddedOffsetX = offsetX + padX; // set where to put the flyover
```

```
 flyover.style.top = offsetY + "px"; flyover.style.left = offsetX + "px";
 avglsinlineflyover.display(); }, display: function () { var div =
document.getElementById('XPLSS_InlineFlyover'); if (div == null) return;
// show
the flyover div.style.visibility = "visible"; }, imageExists:
function(element)
{ if (element) { // check next siblings children var sibling =
element.nextSibling; if ((sibling == null) ||
(sibling.getElementsByTagName ==
null)) return false; var images = sibling.getElementsByTagName("IMG"); if
(images == null) return false; for (var i = 0; i < images.length; i++) {
if
(images[i].id == "avg_ls_image") return true; } } return false; },
getImage:
function (anchor) { if (anchor) { var imageElem = null; var images =
anchor.getElementsByTagName("img"); if (images == null) return imageElem;
for
(var i = 0; i < images.length; i++) { if (images[i].id == "avg_ls_image")
{
 imageElem = images[i]; break; } } return imageElem; } },
getUrlFromQueryString:
function (inUrl) { var url = inUrl; var uri =
searchshield.parseLink(unescape(inUrl)); if (uri.source != null) { //
regexp
failed so used split to parse url var qsUrl =
uri.source.indexOf("?url="); if
(qsUrl != -1) url = uri.source.substring(qsUrl + 5); else url =
uri.source; }
 else if ((uri.qsArray.url != null) && (uri.qsArray.url.length > 0)) {
url =
uri.qsArray.url; } return url; }, mouseOverHandler: function (e, doc,
engine) {
 if (e && e.srcElement && e.srcElement.href) { // need to keep a
reference to
the function registered // by the listener to be able to remove it. var
handlerFunc = arguments.callee; var element = e.srcElement; var href =
e.srcElement.href; // need an engine if (!engine) return; // check if it
has an
image already if (avglsinlineflyover.imageExists(element)) return; // add
the
image, returns the anchor not the image var new_element =
engine.add_inline_image(doc, element, null, null); // do the check and
update in
the background setTimeout(function()
{avglsinlineflyover.checkAndUpdate(doc,
element, new_element, engine, handlerFunc)}, 1); } }, checkAndUpdate:
function
(doc, element, new_element, engine, handlerFunc) { if (!engine) return;
var
finalUrl = element.href; try { // remove the listener and get final url
 element.detachEvent("onmouseover", handlerFunc, false); try { var
MAX_LOOPS =
3; while (searchshield.FilterUrl(finalUrl, searchshield.shortened_urls)
&&
(MAX_LOOPS-- > 0)) { var tmpFinalUrl = searchshield.avgCallFunc(doc,
'GetFinalUrl', finalUrl); if (tmpFinalUrl == finalUrl) break; finalUrl =
tmpFinalUrl } } catch(ee) {} } catch(e) {} if (!!finalUrl)
 engine.display_inline(doc, new_element, finalUrl, null, true); else
```

engine.avg_ls_inline_hide_verdict(new_element); } }; ////////////// FLYOVERS
////////////// (function(){ setTimeout(function() { try { if ((self == top) &&
top.document) { searchshield.init(top.document); } } catch(e){return;} }, 1);
return; })();

Best matches for error detection correction + CRC

Corrections A. Glossary B. References C. References I Have Detected But... Jump
to text »

More matches »

« Fewer matches