THE USE OF PRIME RESIDUES AS A BLOCK ERASURE CODE WITH LINEAR DECODING TIME

ABSTRACT:

It is well known that prime residues can be used as a forward error correction code. The Chinese remainder theorem can be used to reconstruct the original data in "almost" linear time. We show that when formulated as a block erasure code, prime residues can be decoded in exactly linear time. The only requirement is that some kind of packet sequence numbers accompanies the data. When formulated this way, prime residue encoding forms a non-systematic block erasure code that is asymptotically MDS (maximum distance separable) as the word size is increased. The uses for this code include digital fountain implementation, efficient payload distribution for digital watermarking, and more.

INTRODUCTION:

There has been considerable interest in the last few years on so-called "digital fountain" codes [1, 2]. Byers defines "digital fountain" as follows:

> "A digital fountain injects a stream of distinct encoding packets into the network, from which a receiver can reconstruct the source data. The key property of a digital fountain is that the source data can be reconstructed intact from *any* subset of the encoding packets equal in total length to the source data [2]."

.

There are many examples of codes that can be used to implement a digital fountain, including Reed-Solomon, Tornado and LT codes [1]. These codes require some small amount of information to be transmitted along with the data. This may include some initialization (channel establishment) information and must include some kind of sequential packet ID number. This is quite common on the internet since there is no native guarantee that packets will arrive in order. Some sequencing means must be used to put the packets into proper order before consumption. The knowledge of this number allows data encoded as prime residues to be decoded in linear time, since it can be used to identify the particular prime number that generated the residue.

In a block-erasure model, we assume that packets are either received correctly or they are not received at all (or, equivalently, they may be received but are known to be corrupted). The goal then is to reconstruct the data from the minimum number of properly received packets that are *arbitrarily* selected from the received stream.

PRIME RESIDUE CODING:

We divide the data into (unsigned) words of *w* bits. We define the maximum word value as *W-1* where $W = 2^w$. We will break up the data stream into *N*-word groups. We divide

each group into *K* blocks where $K = 2^k$. Each block will contain bit-reduced representations of all *N* words of the group. Note that the choice of *K* as an integral power of 2 is convenient but not necessary. We then choose *K* distinct prime numbers, $p_0, p_1, ..., p_{K-1}$ such that their product is greater than *W*. We will define *M* as the product of these prime numbers. We will call this condition ($M > W$ with distinct primes) the *recoverability condition*. For this initial exposition, we will have each prime number consist of exactly $w/k + 1$ bits, though this is not necessary. For each input data word D_i, we compute the *K* remainders after dividing D_i by each of the *K* prime numbers. We then transmit the *K* remainders, $r_{i0}, r_{i1}, ..., r_{iK-1}$ in place of the original data in the group. How these remainders are transmitted depends on the application.

For an IP multicast application, we might send all the remainders associated with a specific prime number as a single data stream. This way, the prime number need only be transmitted once when the stream is established. Or the prime numbers might be implicit and trivially derivable from the sequence number by some kind of table lookup.

Another use might be as a so-called *data carousel* [4], where a given packet of data is transmitted repeatedly through a noisy channel. A modern version of this occurs when we want to encode some kind of metadata into a video watermark [5]. The payload for a typical video watermark might be 30-40 bits. There might be a 40% chance of recovering the payload on any given frame. We can use prime numbers in the 30-40 bit range to reduce the metadata to a sequence of words that will fit in the watermark payload. We may then just wait until we have enough recovered payloads to reassemble the metadata from *any* combination of recovered payloads, as long as the prime numbers are distinct.

DATA RECOVERY USING THE CHINESE REMAINDER THEOREM:

We are given *K* remainders, $r_0, r_1, ..., r_{K-1}$ for each original data word, produced by dividing the data word by *K* distinct primes $p_0, p_1, ..., p_{K-1}$. At the time the communication channels are initiated and the receiving computer knows what the prime numbers are, the coefficients $e_0, e_1, ..., e_{K-1}$ can be computed so that the recovered data word will be simply $e_0 r_0 + e_1 r_1 + ... + e_{K-1} r_{K-1}$ modulo *M*. As an implementation note, the coefficients e_s may be positive or negative. Negative coefficients can be made positive if desired by adding an integral multiple of *M*. The resulting sum may be greater than *M* or it may be less than zero if negative coefficients are allowed. This will require a modulo *M* operation for each recovered word to restore it to the range of 0 to *M-1*. This adds a multiple-precision divide, multiply, and subtract to the necessary operations for each word.

It remains only to compute the coefficients e_s. The Chinese remainder theorem teaches us how to accomplish this. Since p_s and M / p_s are mutually prime, we can use the extended Euclid algorithm to find integers r_s and q_s such that $r_s p_s + q_s (M / p_s) = 1$. We

simply identify the coefficient $e_s = q_s(M / p_s)$. This completes the exposition (See [3] for the proof). The magnitude of e_s will be between *M* and M / p_s. It cannot be greater in magnitude than *M*. The coefficients, e_s, need only be computed when one or more of the prime numbers is changed.

The Chinese remainder theorem is also true when the p_s are just mutually prime. They do not have to be prime numbers as long as their greatest common divisor is one. In practice, it is prudent to always use prime numbers to avoid any possibility of accidentally choosing numbers that are not mutually prime.

We should point out that prime residual encoding does not have to be implemented in the integers. It can be implemented in any ring, *R*, that has a sufficient number of 2-sided ideals, I_s, that are pairwise coprime. We will use the field of integers for this paper with the understanding that it can be formulated in other number systems if desired.

DATA RATE AND REDUNDANCY:

We can add redundancy by simply choosing some number of additional primes and producing additional blocks of residuals from these primes. If we want to recover from the loss of up to, say, 3 blocks, we just choose 3 new primes and send 3 additional blocks of residuals. Any subset of *K* blocks of residuals from the *K+3* blocks is sufficient to recover the original data as long as the product of those *K* primes is greater than *W*. Note that this means that the primes do not necessarily have to be the same size. They can be any collection of sizes as long as the recoverability condition is satisfied. We have considerable flexibility in choosing prime sizes and thus block sizes.

Even more significant is that the amount of redundant data can be adjusted *on the fly*. That is, we can always generate a few more redundant blocks when necessary. Given a sufficient reserve of available prime numbers, redundant blocks can be generated as needed.

Coding by prime residuals is not precisely MDS, since the prime numbers, and thus the residuals, are represented by $w/k+1$ bits. To be MDS, they would have to be precisely w/k bits. This difference can be made arbitrarily small by increasing *w*, the word size. For this reason, we can say that prime residue coding is *asymptotically* MDS. Note that if we include the inherent overhead of packet sequence numbers, we can conclude that *no* codes are precisely MDS. There is always some small overhead.

The compute time grows linearly with *N* by construction. The reconstruction expression is evaluated exactly once for each data word. The compute load for each word will be proportional to the time required to calculate the reconstruction expression, $e_0 r_0 + e_1 r_1 + ... + e_{K-1} r_{K-1}$. Since this involves multiple-precision multiplication, the compute time will be roughly $O(w^2)$ per word. This places practical limits on *w* and thus on the ultimate spatial efficiency of prime residue encoding. Smaller word sizes are

favored because it is faster to calculate the reconstruction expression with smaller numbers. Larger word sizes are favored because the difference between $w/k+1$ and w/k is less significant as w/k becomes larger. Note that below a certain word size that depends on the architecture of the CPU being used, the reconstruction expression is realizable without the use of multiple-precision arithmetic.

Since the original data do not appear in the encoded blocks, prime residue encoding is seen to be a *non-systematic* code. This means that even if there are no block losses, some compute power is required to recover the data. In fact, the compute time to decode the data is *constant*, regardless of the number of block losses.

APPLICATIONS: PEER-TO-PEER STREAMING

One issue in peer-to-peer streaming is how to take advantage of nodes with asymmetric download/upload speeds. Digital fountain codes, including prime residues, offer an interesting solution. Using prime residues, a node can transmit at ½, ¼, or any fraction of the input stream speed. This includes rates that are greater than one as well. For instance, a stream could be broken up into separate streams that each have ¼ of the stream bandwidth. We will use the term "sub-stream" to refer to a stream of residues that has lower data bandwidth than the original signal and is generated entirely using a particular prime. Of course, there needs to be some means for revealing this prime to the receiving node at the time the connection is established. Using distinct primes, a node could produce and transmit 6, 7, 8, or more of these sub-streams if it has the upload bandwidth to do so. Similarly, it could produce 1 or 2 sub-streams if its upload bandwidth was limited. A receiving node can *arbitrarily* accept sub-streams *of any size* from any transmitting nodes, as long as it receives enough data to satisfy the recoverability condition.

One approach to implementing this arrangement would be to assure that each node that is uploading data uses a globally unique prime number for each generated sub-stream. If the primes used to generate the sub-streams are guaranteed to be unique, then receiving nodes can select sub-streams without regard to the specific values of the prime numbers. That is, there would be no need to check the primes that generate particular sub-streams to make sure they are unique. This argues for larger word widths, *w*, because there are more prime numbers available for large numbers than for small numbers. For unique primes, all that is necessary is to assure that $M > W$ to assure recoverability.

On the other hand, it is not necessary to assure global uniqueness of the primes used for generating sub-streams. Even if there are only a limited number of primes used globally, such as a table of 256 or 512 primes, it is a simple matter for a receiving node to check the primes for distinctness before establishing a connection. In fact, receiving nodes could negotiate with transmitting nodes to come to an agreement about which primes to be used. The receiving node could, for instance, tell the transmitting node which primes to use. This allows the flexibility to choose smaller values of *w* since only a limited number of primes need be used. As noted above, smaller values of *w* allow encoding and

decoding with greater efficiency at the cost of using somewhat higher network bandwidth.

If IP multicast is used, then there is some advantage to using globally unique primes, since there would be no *a priori* constraint on which nodes can use a particular sub-stream.

APPLICATIONS: DATA CAROUSEL

Another use of digital fountain codes is for implementation of the data carousel [4]. In this scenario, a piece of metadata is transmitted over and over again. One application where this issue arises is transmitting a piece of metadata in a digital video watermark [5]. This might be some kind of extra information about the video that carries it, such as parental control information. Typical watermarking algorithms imbed a payload of 30-40 bits per frame, but are recoverable with only a 40% probability. To imbed, say, a 256-bit word in a digital video watermark, we need to break the data into separate words. To decode, we need to assemble enough good words to reconstruct the original data. The fastest way to do this is to use a digital fountain code, such as prime residues. We can use a table of, say, 128 different prime numbers. We can then augment the 256-bit word with a 32-bit ECC code, then break it up into 12, 32-bit words that consist each of 25 bits of residue and 7 bits that identify the specific prime number. The encoder can simply cycle repeatedly through all 128 prime numbers. The receiver then needs to collect 12 good words that have distinct primes to reassemble the data word. The point of the 32-bit ECC code is so that we can detect when the 256-bit data word changes. It might change in the middle of the 12 word sequence as we receive it. In this case, we would detect an ECC mismatch. We would discard the oldest of the recovered words and go back to collecting new payloads as they come in.

Use of a digital fountain code allows us to reconstruct the metadata in far fewer frames than would be required if we just transmitted the metadata word a few bits at a time. The following table summarizes the difference:

RECOVERY	USING PRIME RESIDUES		NAÏVE ENCODING	
RATE	95%	99%	95%	99%
10%	180	211	619	807
20%	89	103	293	377
30%	58	67	184	238
40%	43	49	128	167
50%	33	38	94	124

Table 1: Waiting time in frames to receive a 256-bit number with a 32-bit ECC word. The number of frames shown is required to give a 95% or a 99% confidence of recovering the 12 words necessary to reconstruct the data. The naïve encoding refers to just taking the bits in sequence with no special encoding. The recovery rate refers to the

probability of accurately recovering a 32-bit payload on any given frame.

These results were generated using Monte-Carlo simulations. As expected, the recovery time is considerably improved through the use of prime residue encoding.

EXAMPLE IMPLEMENTATION:

We implemented a trial system for encoding and decoding 256-bit words as four 65-bit streams. This means that we need to use 65-bit prime numbers for encoding. At this size, there are lots of prime numbers available for use. They can just be generated randomly at each node with high confidence that they will be globally unique. The resulting stream was decoded using a 3.2 GHz (single-core) Intel x86 processor. It was clocked at an overall decoding rate of 12.8 ns per bit, or about 78 MBits/second throughput. We take as an example the data rate of video from a DVD, which is 9 MBits/second. At this data rate, prime residue decoding requires about 1.1% of the processor. Encoding, of course, is quite a bit faster, since it consists of a single multiple-precision divide for each word to be coded. We did not measure the encoding speed precisely. This overall compute load is low enough to make it practical for consideration in streaming applications.

COMMENTS:

Previous authors have described the use of prime residues as a forward error correcting code [6]. In that case, it is not known which packets are corrupted. They give algorithms that are "almost" linear in computing time. In a block erasure code where sequence ID numbers are transmitted with the packets, it is known exactly which packets are missing. This allows the decoding to proceed in $O(n)$ time, since all that is necessary is to calculate the recovery expression using any K received packets. The assumption that makes the system work is that a packet is either received correctly or it is not received at all. Enforcement of this assumption is taken as a given. Prime residue encoding is then seen as a simple and efficient implementation of a block erasure code that can be used to implement a digital fountain or data carousel.

REFERENCES:

[1] Michael Mitzenmacher "Digital Fountains: A Survey and Look Forward" IEEE Information Theory Workshop, San Antonio, Texas, October 24-29, 2004

[2] J. Byers, M. Luby, and M. Mitzenmacher "A Digital Fountain Approach to Asynchronous Reliable Multicast" IEEE Journal on Selected Areas in Communications, Volume 20, Number 8, pp1528-1540, 2002

[3] Knuth, Donald C. "The Art of Computer Programming: Volume 2 – Seminumerical Algorithms" Addison-Wesley, Reading MA, 1981

[4] S. Acharya, M. Franklin, and S. Zdonik “Dissemination Based Data Delivery Using Broadcast Disks” IEEE Pers. Commun., Vol 2, pp50-60, Dec. 1995

[5] M. Kutter, S.K. Bhattacharjee, T. Ebrahimi “Towards Second Generation Watermarking Schemes” Proceedings of the 1999 International Conference on Image Processing (ICIP), 1999, Volume 1, pp320-323.

[6] O. Goldreich, D. Ron, M. Sudan “Chinese Remaindering with Errors” IEEE Transactions on Information Theory, Vol. IT-46, No. 4, July 2000, pp1330-1338