TOWARDS IMPROVING NETWORK FLOW WATERMARKS USING THE REPEAT-ACCUMULATE CODES

Amir Houmansadr and Nikita Borisov

University of Illinois at Urbana-Champaign
Electrical and Computer Engineering Department
Urbana, IL, USA

ABSTRACT

Network intruders try to hide their identity by relaying their traffic through a number of intermediate hosts, called stepping stones. Network flow watermarks have been used to detect such attacks by inserting a special timing pattern into one flow by means of artificial delays and detecting relayed flows by searching for the same pattern.

We study the application of coding schemes to improve the efficiency of network flow watermarks. In particular, we use the Repeat-Accumulate codes, a class of low complexity, high performance error-correcting codes, to improve the detection performance of a recent flow watermark, the RAINBOW. We show the effectiveness of the improved scheme, C-RAINBOW, through simulation and discuss design tradeoffs.

Index Terms— Repeat-Accumulate codes, network flow watermarking, stepping stone attack, traffic analysis

1. INTRODUCTION

Internet attackers commonly relay their traffic through a number of (usually compromised) hosts in order to hide their identity. Detecting such hosts, called stepping stones, is therefore an important problem in computer security. The detection proceeds by finding correlated flows entering and leaving the network. Traditional approaches have used patterns inherent in traffic flows, such as packet timings, sizes, and counts, to link an incoming flow to an outgoing one [1]. More recently, an active approach called *watermarking* has been considered [2, 3]. In this approach, traffic characteristics of an incoming flow are actively perturbed as they traverse some router to create a distinct pattern, which can later be recognized in outgoing flows. These techniques also have relevance to anonymous communication, as linking two flows can be used to break anonymity.

The existing flow watermarking schemes are based on using a randomly generated watermark sequence, being shared amongst different entities of the watermarking system, and being looked for in the suspected network flows. Previous work has used some simple coding schemes, such as spread-spectrum [2] and repetition codes [4, 5], to improve detection performance; however, we believe the space of coding techniques that can be applied to this work has been largely underexplored. To address this, we investigate the use of Repeat-Accumulate (RA) codes to improve the detection performance of a recent flow watermarking scheme, RAINBOW [3]. We show through simulations that the coding-empowered version of the RAINBOW, namely C-RAINBOW, provides better detection performance. We also explore different design tradeoffs for the C-RAINBOW scheme. We believe that this new framework can also be used to design efficient covert channels over the timing domain of network flows; we leave this as a future research.

The rest of this paper is organized as follows: in Section 2 we design a new flow watermarking scheme, called C-RAINBOW, which uses Repeat-Accumulate codes in order to improve the detection performance. In order to show the effectiveness of coding in improving the detection performance, in Section 3 we simulate and compare the detection performance of C-RAINBOW with its ancestor the RAINBOW scheme. The paper is concluded in Section 4.

2. C-RAINBOW FLOW WATERMARKING SCHEME

2.1. Repeat-Accumulate codes

Repeat-Accumulate (RA) codes are a class of error-correcting codes that despite being simple and low complexity they provide an outstanding good performance [6]. An RA code works as follows: an information block of size n_A is repeated q times and then scrambled by a permutation function of size $q \times n_A$, and then encoded by a rate 1 accumulator. The RA codes are fast, as their encoding time is linear. Also, the rate of an RA code is $1/q$. The RA decoding is performed using the sum-product algorithm through a number of *iterations* [7]. We will discuss the effect of the iteration numbers on the detection performance in the consecutive sections.

2.2. C-RAINBOW scheme

In this section, we describe the design of the C-RAINBOW flow watermarking scheme, which is an improved version of the RAINBOW scheme by using the RA coding. The C-RAINBOW watermark insertion scheme is the same as that of RAINBOW [3], except for the generation of the watermark sequence. In order to generate the watermark sequence, the C-RAINBOW watermarker first generates an n_A bits *actual watermark*, w_A, which is a drawn randomly from $\{0, 1\}$. The actual watermark is, then, coded using an RA encoder $C_{RA}(\cdot)$, resulting in an $n_I = q \times n_A$ bits *inserted watermark*:

$$w_I = C_{RA}(w_A) \tag{1}$$

where q is the redundancy parameter of the coding algorithm. As mentioned before, the RA encoder $C_{RA}(\cdot)$ uses a random permutation function $p(\cdot)$ which is shared with the watermark detector. The generated w_I watermark is then used to watermark a candidate flow using the watermarking scheme described in [3]. Note that in the original RAINBOW scheme all of the n_I inserted watermark bits are generated at random. In both of the schemes, prior to the watermark insertion the watermark bits are converted from the binary format into $\{-a, +a\}$, e.g., 0 gets converted to $-a$, where a is the *watermark amplitude*.

The RAINBOW watermark detection works in a non-blind manner: upon receiving a flow detector subtracts its inter-packet delays (IPD) from that of another flow from a database, and then calculates the normalized correlation between the subtraction result, d, and the inserted watermark sequence [3]. It, then, declares the candidate flow to be watermarked if the calculated correlation metric exceeds some *detection threshold*. The C-RAINBOW detection scheme differs from that of RAINBOW scheme in two ways: first, the subtracted IPDs, d, passes through a soft-limiter block in order to remove the outlier noise components added because of the noisy communications network. Second, instead of correlating the subtracted IPDs with the inserted watermark, w_I, the subtracted IPDs sequence is first passed through an RA decoding algorithm $D_{RA}(\cdot)$ (using the same q and $p(\cdot)$ as in $C_{RA}(\cdot)$), and the resulted sequence is then correlated with the actual watermark, w_A. Similar to the RAINBOW detector, the resulted correlation metric is, then, compared with a detection threshold in order to decide whether the candidate flow is watermarked.

3. SIMULATIONS AND DISCUSSIONS

We simulated the RAINBOW and C-RAINBOW schemes in Matlab and compared their detection performances. In order to observe the performance improvement only caused by using the RA coding, we apply the soft-limiter noise reducer block over the RAINBOW detector as well. In our experiments, the network flows are simulated by modeling them as Poisson processes, and the network jitters are picked up at random from a database of network jitters measured over the Internet. Each experiment is run for 10000 times. In the following, we discuss the effect of different parameters in the detection performance.

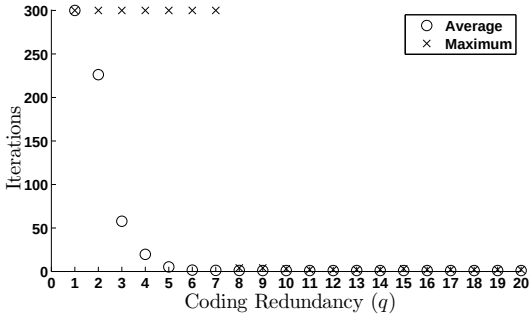


Fig. 1. The average and maximum number of iterations required by the C-RAINBOW detector to detect all of the inserted bits.

3.1. Effect of the redundancy parameter (q)

As mentioned before, the RA decoder works by going through a number of iterations until its output converges to the actual watermark w_A. Since more number of iterations requires more processing and time resources a system designer looks for a detector that converges in fewer number of iterations. Figure 1 illustrates the average and maximum number of iterations required to detect the total number of actual watermark bits for different values of q (a maximum of 300 iterations tried, $a = 10msec$, and $n_A = 100$). As we can see, the number of iterations exponentially decreases with the q parameter, such that for $q \geq 5$ the decoder is able to construct all the actual bits within at most 2 iterations.

Figure 2(a) sketches the empirical mean and standard deviation of the C-RAINBOW detection metric for different values of q, along with those of the RAINBOW detector (10000 runs). As can be seen, the mean of C-RAINBOW detection metric exponentially converges to 1 by increasing q, and for $q > 2$ it outperforms that of the RAINBOW detector. Also, by increasing q the empirical standard deviation of the C-RAINBOW decreases exponentially, such that it equals to 0 for $q > 6$.

We also run similar experiments to evaluate the effect of the q parameter on the false positive rate. In order to make the detection faster, a detector may limit the number of iterations to the maximum numbers required for true detection (Figure 1). Figure 2(b) shows the empirical mean and standard deviation of the detection metrics for the C-RAINBOW and RAINBOW schemes for different values of q. Again, q has no effect on the RAINBOW metric, however, increasing q increases the mean and variance of the C-RAINBOW metric. This, in fact, is because increasing q for a fixed value

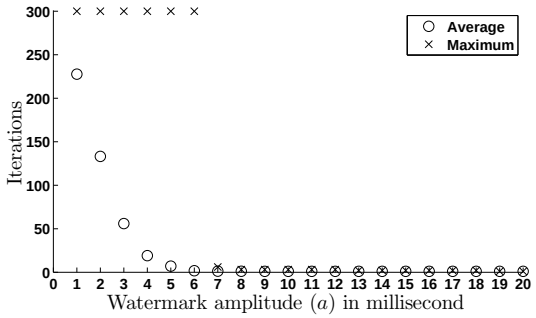


Fig. 3. The average and maximum number of iterations required by the C-RAINBOW detector to detect all of the inserted bits.

of n_I decreases the number of actual watermark bits for the C-RAINBOW scheme, e.g., $n_A = n_I/q$. We can show that the false positive rate has reverse relation with the number of actual watermark bits, e.g., $\binom{n_A}{k}(0.5)^{n_A}$ is the probability of k false positive bits.

3.2. Effect of the watermark amplitude (a)

We performed similar experiments to evaluate the effect of the watermark amplitude a. Figure 3 shows the average and maximum number of iterations needed for the decoder in order to extract all of the watermark bits ($q = 10$, and $n_A = 100$).

Figure 4(a) illustrate the empirical mean and standard deviation of the watermarking schemes for different values of a. As can be seen, increasing a increases the mean value of the both detectors exponentially, however for the C-RAINBOW scheme the standard deviation decreases much faster. This shows that increasing a better improves the C-RAINBOW's performance. Also, for smaller values of a, coding helps in detecting at least some of the bits while the RAINBOW performs very poorly. We also run the same experiments for the false detection. As can be seen from Figure 4(b), the false detection behaviour of the both of the schemes is independent of the watermark amplitude a.

3.3. Effect of the watermark bit count (n_I)

We run similar experiments to measure the effect of the number of inserted watermark bits, n_I. Our simulations show that for $N \geq 50$ the detector is able to extract all of the watermark bits in at most 2 iterations (for $a = 10msec$, and $q = 10$). Figure 5(a) shows the empirical mean and standard deviation of the detection metrics for both of the schemes for different values of n_I. For n_I as small as $n_I = 50$ the C-RAINBOW scheme detects all of the watermark bits in all of the 10000 runs, which performs far better than the RAINBOW scheme. Figure 5(b) shows the mean and standard deviation of the watermarking schemes for the case of false detection. For both of the schemes increasing n_I improves the variance of the false detections, however RAINBOW always outperforms the C-RAINBOW in terms of the false positives, as RAINBOW has q times more actual bits than the C-RAINBOW scheme.

4. CONCLUSIONS

In this paper, we take the first steps towards improving the detection performance of network flow watermarks by using coding algorithms. Specifically, we use the Repeat-Accumulate codes to enhance the detection performance of the recent RAINBOW watermarking scheme. Our simulations confirm that for similar set of watermarking parameters, e.g., the watermark amplitude and the number of inserted bits, the new scheme, C-RAINBOW, performs better detection. We believe that this framework can also be used to design efficient covert timing channels for network flows, that we leave for the future research.

5. REFERENCES

[1] Y. Zhang and V. Paxson, "Detecting stepping stones," in *USENIX Security Symposium*, Steven Bellovin and Greg Rose, Eds., Berkeley, CA, USA, Aug. 2000, pp. 171–184, USENIX Association.

[2] W. Yu, X. Fu, S. Graham, D.Xuan, and W. Zhao, "DSSS-based flow marking technique for invisible traceback," In Pfitzmann and McDaniel [8], pp. 18–32.

[3] Amir Houmansadr, Negar Kiyavash, and Nikita Borisov, "RAINBOW: A robust and invisible non-blind watermark for network flows," in *NDSS*. 2009, The Internet Society.

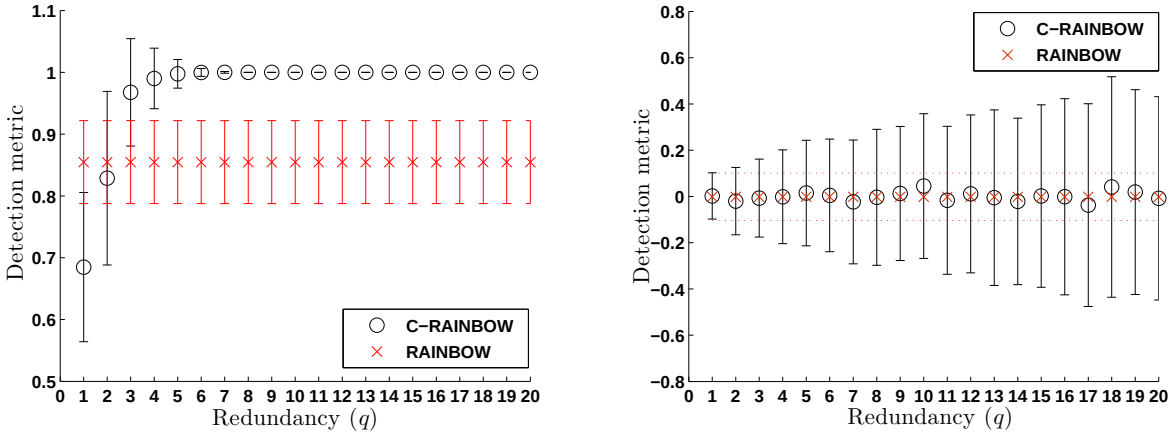
[4] Y. Pyun, Y. Park, X. Wang, D. S. Reeves, and P. Ning, "Tracing traffic through intermediate hosts that repacketize flows," in *IEEE Conference on Computer Communications (INFOCOM)*, George Kesidis, Eytan Modiano, and R. Srikant, Eds., May 2007, pp. 634–642.

[5] X. Wang, S. Chen, and S. Jajodia, "Network flow watermarking attack on low-latency anonymous communication systems," In Pfitzmann and McDaniel [8], pp. 116–130.

[6] Dariush Divsalar, Hui Jin, and Robert J. McEliece, "Coding theorems for "turbo–like" codes," in *Proceedings 36th Annual Allerton Conference on Communication, Control, and Computing*, Monticello, IL, USA, Sept. 1998, pp. 201–210.

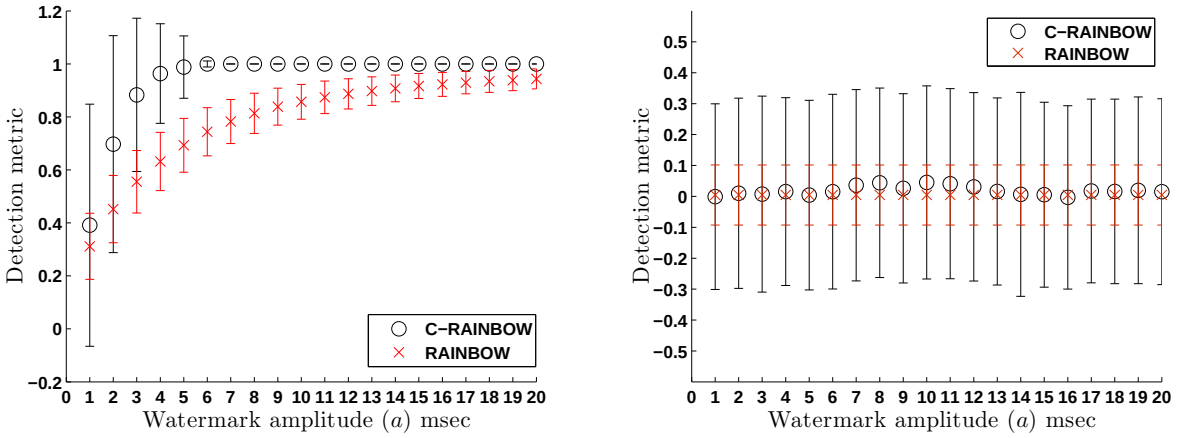
[7] Frank R. Kschischang, Brendan J. Frey, and Hans-Andrea Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, 2001.

[8] Birgit Pfitzmann and Patrick McDaniel, Eds., *IEEE Symposium on Security and Privacy*, May 2007.



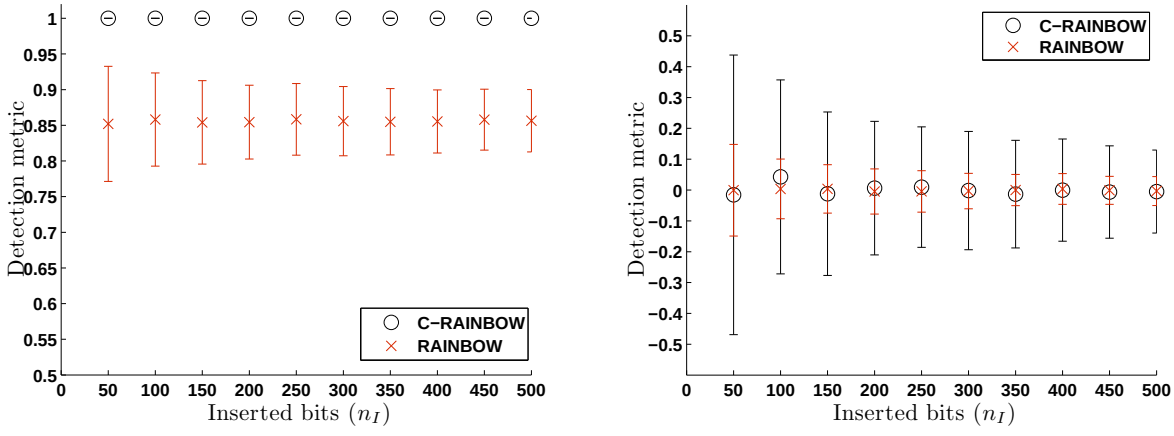
(a) True detection (b) False detection

Fig. 2. Empirical mean and standard deviation of the detection metrics for different q ($a = 10msec$, $n_I = 100$).



(a) True detection (b) False detection

Fig. 4. Empirical mean and standard deviation of the detection metrics for different a ($q = 10$, $n_I = 100$).



(a) True detection (b) False detection

Fig. 5. Empirical mean and standard deviation of the detection metrics for different n_I ($q = 10$, $a = 10msec$).