Decoding a VCR Controller Code

Ken Shirriff[1]
Curt Welch[2]
Andrew Kinsman[3]

ADDRESSES: (1) 6406 Hillegass, Oakland, CA 94618
Email: shirriff@sprite.Berkeley.EDU
(2) 3126 Windwood Farms Dr., Oakton, VA 22124
Email: curt@kcwc.com
(3) 5441 Holtz Rd., Palmyra, NY 14522
Email: kinsman@ssd.Kodak.Com

ABSTRACT

The VCR Plus+ is a remote control for programming video cassette recorders. It uses an encrypted 1 to 8 digit number that encodes the channel, start time, length, and day of the month of the television show to be recorded. This paper describes a procedure for decoding the 1 to 6 digit codes.

Keywords: Cryptography, VCR

November 6, 1993

Decoding a VCR Controller Code

Ken Shirriff[1]
Curt Welch[2]
Andrew Kinsman[3]

ADDRESSES: (1) 6406 Hillegass, Oakland, CA 94618
Email: shirriff@sprite.Berkeley.EDU
(2) 3126 Windwood Farms Dr., Oakton, VA 22124
Email: curt@kcwc.com
(3) 5441 Holtz Rd., Palmyra, NY 14522
Email: kinsman@ssd.Kodak.Com

INTRODUCTION

The VCR Plus+ is a video cassette recorder controller designed to simplify VCR programming. To record a show, you just enter a code number into the VCR Plus+ controller, and the VCR Plus+ looks after controlling the VCR. The 1 to 8 digit code numbers, called PlusCodes, are printed in many TV listings. Details of the coding method, invented by Henry Yuen and Daniel Kwoh, have not been revealed [1]. However, the PlusCode system has several cryptographic flaws that allow the code to be broken. This article explains how 1 to 6 digit codes can be decoded; we were unable to decode the 7 and 8 digit codes.

The code numbers are assigned so common, prime-time shows usually have short, 1 to 4 digit codes, while less common shows have longer numbers. This Huffman-style encoding scheme [2] minimizes the average length of the code numbers used. Each code number contains the day of the month, channel, start time, and duration of the show to be recorded. The start time and duration are in steps of 5 minutes.

After the user enters codes into the VCR Plus+, the VCR Plus+ remembers these codes up to a month and signals the VCR at the right time. The VCR Plus+ controls the VCR by using the infra-red remote control signals. The VCR Plus+ knows how to control common VCRs, and is told by the user what kind of VCR is being used. When it is time for the show to start, the VCR Plus+ signals the VCR to change to the proper channel and to start recording. At the end of the show, the VCR Plus+ signals the VCR to stop recording. The VCR Plus+ uses an internal user-programmed table to convert the encoded channel number to the VCR's channel number, in case channels are numbered differently in the local area. The VCR Plus+ has an internal clock with the current year, date, and time; it uses this to keep track of when to start recording.

Note: VCR Plus+ and PlusCode are trademarks of the Gemstar Development Corporation. Gemstar claims copyright and patent protection on the PlusCode numbers and software to create them. The decoding process in this paper is described for cryptographic research purposes only.

The numbers are decoded using a fairly complex technique, which is shown in Figure 1. In overview, the entered code number is multiplied by a fixed key to generate an intermediate number. The bottom three digits are divided by 32. The quotient gives the day of the month for the code. The remaining top digits are processed to generate an offset and new top digits. The new top number is converted to binary and the bits are separated into bits used for the channel and bits for the table index. The offset, the remainder from the bottom three digits, and the day multiplied by the month are all added to produce a sum. The sum is converted to binary and the bits are separated into channel bits and table bits. All the channel bits are combined to yield the channel of the show. The table bits are combined and are used to index into a scrambled table of start times and durations to yield the starting time and duration of the show.

DETAILS OF THE DECODING METHOD

There are two basic concepts used in the decoding; we call them no-carry multiplication and length looping. No-carry multiplication is ordinary long multiplication with all carries discarded. We will denote this operation by (×). For example, 123 (×) 456 =

628 (+) 5050 (+) 48200 = 43878, where (+) indicates addition with carries discarded. It can be verified that no-carry multiplication is commutative and associative. No-carry multiplication is very similar to convolution of vectors modulo 10.

Length looping is used to generate mappings that preserve the length of the input. For example, suppose we have a one-to-one mapping M4 on numbers up to 4 digits long, so M4 converts each input number from 0 to 9999 to a new, unique number from 0 to 9999. Now suppose we want to make M4 into a new mapping that operates on numbers that are exactly 4 digits long; that is, it bijectively maps numbers from 1000 to 9999 onto 1000 to 9999. The `length_loop` procedure used to generate the new mapping is given in Figure 2. The concept is to apply mapping M4 to the 4 digit input number. If the result is too short due to a leading zero, repeat by applying M4 to the result. This loop is continued until the result is the proper length. The `length_loop` procedure works analogously for inputs of any length i, using mapping Mi.

It can be seen that length looping results in a new one-to-one mapping. Suppose the mapping isn't one-to-one, so there are two inputs x and y that yield the same result. Since M4 is one-to-one we can step backwards from the common result and show that either x=y or the loop would have terminated earlier. It can also be seen that the length loop must terminate: if it doesn't, some value must appear twice. However, since M4 is one-to-one we can work backwards and show the initial value must appear twice. However, that would terminate the loop unless the initial value has a leading zero, which is prohibited.

We will now explain the decoding method of Figure 1 in more detail. In Step 1 of the decoding procedure, the number is no-carry multiplied by the decoding key, the result is truncated to the original length, and length looping is used to prevent leading zeros. This algorithm is shown as the `key_mult` routine in Figure 2. The previous discussion on length looping shows that Step 1 is a one-to-one mapping from the input code to a new code of the same length. For an i digit input, the base mapping Mi consists of no-carry multiplication by 68150631 followed by truncation to i digits. Each mapping Mi can be shown to be one-to-one since the key 61850631 has inverse 9371 under 8 digit no-carry multiplication. (The encoding process uses no-carry multiplication by 9371.)

Steps 2 and 3 determines the date from the multiplied number. The bottom three digits from Step 1 are split off and used in Step 2. One is subtracted from the bottom digits and the result is divided by 32. Adding one to the quotient yields the day of the month. The remainder is used in Step 5.

After the bottom three digits are split off for Step 2, the remaining top digits are processed in Step 4. Step 4 computes the modified top number and the offset; details of this step are given in the `offset` and `map_top` procedures of Figure 2. The modified top number is computed from the original top number by a one-to-one length-preserving mapping that uses the day and year. (Only the last two digits of the year are used.) The base mapping is done by the procedure `map_top` in Figure 2. The `offset` routine in Figure 2 applies length looping to the `map_top` mapping to yield the new, mapped top. Inside the length loop in Step 4, the offset is also computed. The previous proofs about length looping can be used to prove the resulting top number mapping is one-to-one and preserves the length.

The number formed from the new mapped top digits is converted to binary, and the bits are labeled $t_8c_5t_7c_4t_6t_5t_4c_3t_3c_2$, where t bits are used for the table index and c bits are used for the channel. If the input code is 1 to 3 digits long, Step 4 is not used; the mapped top number and offset are both 0.

In Step 5, the remainder from Step 2, the month plus one times the day, and the offset from Step 4 are added modulo 32. The result is converted to a 5 bit binary number and the bits are denoted $t_2c_1t_1c_0t_0$.

In Step 6, the channel bits from Steps 4 and 5 are combined into a 6 bit binary number. Adding 1 to the channel bits yields the channel corresponding to the code.

Finally, in Step 7, the table bits from Steps 4 and 5 are combined into a 9 bit binary number. The resulting table number is used as the index into a table holding combinations of start times and durations for shows. The entries in this table are in a shuffled order. The most common times and durations tend to be near the beginning of the table, allowing common shows, such as half-hour prime time shows, to be encoded with short codes. The first part of the table consists of 192 entries, combining the 48 half-hour start times and the 4 durations of 30, 60, 90, and 120 minutes. This part of the table starts out with entry 0 as 6:30 pm - 30 minutes, entry 1 as 4:00 pm - 30 minutes, entry 2 as 7:30 pm - 30 minutes, and ends with entry 191 as 11:00 am - 90 minutes. Following these 192 scrambled entries are the entries for longer durations in sequential, decreasing order. These entries start with 48 entries of 150 minutes duration, counting down: 11:30 pm, 11:00 pm, 10:30 pm, down to 12:00 am. These entries are followed by the 180, 210, 240, 270, and 300 minute durations in the same counting down order.

EXAMPLES OF THE DECODING METHOD

For an example of the decoding method, consider the code number 3316 in May, 1991. No-carry multiplying 3316 and 68150631 and truncating to 4 digits yields 8296, which is separated into bottom digits 296 and top digit 8. Dividing 296-1 by 32 yields quotient 9 and remainder 7. Incrementing the quotient yields day = 10. The offset loop with day 10 and year 91 gives offset = 8, and mapped top = 8. Then (7+ 10 $\times$ (5+1) + 8) mod 32 = 11. Converting 11 to binary, $i_2 c_1 i_1 c_0 i_0$ = 01011. From the mapped top digits $t_1 c_3 t_0 c_2$ = 1000 binary. Extracting the channel bits yields 0011 binary, and adding 1 yields channel 4. The table bits are 10001 binary or 17. Entry 17 of the table has start time 9:00 pm, and duration of 2 hours. Therefore, PlusCode 3316 in May, 1991 corresponds to a 2 hour show on channel 4 at 9:00 pm on May 10th.

For another example, consider code number 21362 in March, 1992. No-carry multiplying yields 36322. Dividing 321 by 32 yields quotient 10 for day 11 and remainder 1. The offset loop gives offset = 1 and mapped top 28 or 1000010 binary. Adding (1 + 11 $\times$

(3+1) + 1) mod 32 = 14 or 01110 binary in Step 5. The binary numbers yield channel bits 10111 giving channel 24 and table bits 0000010 for table entry 2, which is 30 minutes at 7:30 pm.

HOW WE DETERMINED THE CODE

To determine the coding method, we examined the 1 digit codes and the associated date/channel/time in various months. From these codes, we figured out Steps 2, 3, 5, and 6. We also figured out the first 8 entries in the table for Step 7. Note that 1 to 3 digit codes don't use Step 4 and 1 digit codes are unaffected by Step 1. It is a serious flaw in the PlusCode system from a cryptographic standpoint that so much could be simply determined from the 1 digit codes. The main problem is that the encoding for 1 digit codes is very simple, yet it reveals most of the steps used: the complex Steps 1 and 4 drop out for the 1 digit codes, and only 8 table entries are used in Step 7.

Next we examined a number of 2 and 3 digit codes. The information we got from the 1 digit codes revealed what 3 digit codes must be going into Step 2. From this, we determined there must be an initial processing step and constructed a table showing what must be happening in this first step. Eventually we discovered that no-carry multiplication by 631 would explain the first step, with a few exceptions. The exceptions turned out to have a leading zero, and we thus discovered the length looping process.

Once we had determined these facts, it was simple to determine the entire eight digit multiplication key, digit by digit. The day could be determined from the last 3 digits of any 1 to 8 digit code, as long as length looping didn't occur. For each input length from 4 to 8, we tried codes with the 9 possible leading digits and noted which one decoded to an unexpected day. We knew that the key multiplication of the exceptional code must have yielded a leading digit of 0, so simple algebra revealed what the corresponding digit of the key must have been. In this manner, we quickly obtained the entire 8 digit key. It is another cryptographic flaw of the PlusCodes that the key can be obtained so easily.

It was significantly more difficult to decode the 4 digit codes: these codes use the complex Step 4 and use more table entries. By examining 4 digit codes, we first discovered that the bottom 3 digits are processed separately and an offset depending on the top digit was added in Step 5. Next, we determined the new table entries by holding the code constant and varying the month; this stepped through the table indices from Step 5. We examined the 9 possible top digits to see how they affected the channel and table, and figured out the formula for the mapped top digit. We also figured out how the bottom 4 bits of the mapped top digit were separated into channel and table digits.

The offset was the hardest thing to determine in the 4 digit codes. We figured out that the offsets were a function of the year, the day modulo 10, and the thousands digit. However, even after determining the whole offset table for 1991, the formula was still mysterious. We determining the offset table for other years and noticed a simple pattern from one year to the next. Studying this pattern revealed that the offset was computed by summing `map_top` over multiple years.

The 5 and 6 digit codes were harder than the 4 digit codes. Since we could pick codes that would use any desired table entry, we quickly stepped through the start and duration table and determined its contents. It is another cryptographic flaw with the PlusCode system that it is so easy to read out the table. Next, we could determine the modified top digit formula and bit order by examining a sequence of top digits and seeing what changed. The 5 and 6 digit offset formula was very hard to determine. However, the problem was that we were computing the offset and the modified top digits separately. After combining both these functions into a single loop, the extension of the offset formula to 5 and 6 digits was straightforward.

We haven’t determined how the 7 and 8 digit codes work. The decoding method seems to change significantly. For instance, the offset and modified top formulas do not depend on the day as they do for shorter codes. The offset apparently does not depend on the year at all, while the modified top depends on the exact year, not the year modulo 16.

ACKNOWLEDGEMENTS

We would like to thank the participants in the cryptography discussion on the Usenet computer network for their comments. We would especially like to thank the Electronic Frontier Foundation for their legal assistance.

REFERENCES

1. Churbuck, D., Success Formula, *Forbes*, May 27, 1991, p. 334.

2. Huffman, D. A., A Method for the Construction of Minimum-Redundancy Codes, *Proceedings of the IRE 40*, 9 (September 1952), 1098-1101.

BIOGRAPHICAL SKETCH

Ken Shirriff is currently working towards a doctorate degree in computer science at the University of California, Berkeley.

Curt Welch is an independent consultant, currently working for the David Taylor Research Center.

Andy Kinsman is an computer engineer at Kodak.