Эссе по курсу «Защита информации», кафедра радиотехники, Московский физико-технический институт (ГУ МФТИ), http://www.re.mipt.ru/infsec

CRC-алгоритмы обнаружения ошибок в транзакциях USB пакетов

Выполнил:

Студент: *Карташов Валерий Михайлович*
Группа: *112*

Эссе по курсу «Защита информации», кафедра радиотехники, Московский физико-технический институт (ГУ МФТИ), http://www.re.mipt.ru/infsec

Введение

Для защиты всех полей пакета (кроме PID – packet identifier) спецификация USB-протокола передачи данных предусматривает использование алгоритма контроля с помощью циклического избыточного кода CRC (Cyclical Redundancy Check). Это способ контроля целостности данных при их передаче и хранении. При его помощи вычисляется контрольная сумма пакета данных, которая передается вместе с ним. Принимающее устройство повторно вычисляет контрольную сумму пакета данных. Несовпадение рассчитанной и принятой контрольной суммы расценивается как ошибка передачи данных, при этом, как правило (в зависимости от типа передаваемого пакета), принимающее устройство производит запрос повторной передачи ошибочного пакета. В этой статье сначала будет обсуждаться математическая основа CRC на интуитивном уровне, а затем последует объяснение его специфической реализации, принятой в USB.

Математическая подоплека

Использование CRC базируется на хорошо известном принципе целочисленного деления. Результатом деления двух чисел является их частное и остаток (который может равняться 0). Если вычесть остаток из делимого и поделить это число на делитель, полученный остаток всегда будет равен 0. Пусть, например, 629 делится на 25. В остатке получаем 4. Если делимое (629) и остаток (4) передаются от источника к приёмнику, при получении данных можно отследить их целостность путём пересчёта остатка и проверки его соответствия переданному. В качестве альтернативы, приёмник может подсчитать разницу между переданными делимым и остатком, а затем поделить это на заранее известный делитель в надежде получить нулевой остаток (в случае отсутствия потери данных и ошибок при передаче).

Принцип целочисленного деления можно также применить и к делению полиномов. (Можно интуитивно понять это, принимая во внимание, что цифры, которые образуют целое, могут быть рассмотрены как коэффициенты полинома с основанием 10, например, 629 = 6*10^2 + 2*10^1 + 9*10^0, где «^» - операция возведения в степень.) Для представления коэффициентов (делимого) полинома может быть рассмотрен бинарный битовый поток (структура из «единичек» и «ноликов»). При делении этого полинома на полином генератора (делитель), который является, по сути, другим бинарным битовым потоком, результатом будет полиномиальный остаток или CRC. Арифметика особенно упрощается, если рассматриваемые «нолики» и «единички» являются элементами конечного поля (поля Галоа второго порядка или GF(2)). Такую арифметику иногда называют арифметикой по модулю 2. Для глубокого понимания процесса вычисления CRC достаточно усвоить, что сложение и вычитание в этом поле сводится к простой операции XOR. Это основа техники, используемой в USB пакетах в генерируемом и контрольном CRC.

CRC полезны тем, что они могут обнаружить все одинарные и двойные ошибки и много составных (или пакетных) ошибок с малым числом бит. Так, например, для всех сообщений, битовая длина которых меньше 32768, CRC-16 (см. далее) может обнаружить все ошибки, состоящие из одного, двух, трёх или любого нечётного числа бит, все пакеты

ошибок с битовой длиной не более 16, 99.997% (1 – 2^(-15)) 17-битных пакетных ошибок и 99.998% (1 – 2^(-16)) 18-битных (и более длинных) пакетных ошибок. (Пакетная ошибка битовой длины b – это произвольная битовая строка, содержащая ровно b бит (все равны 1), начинающаяся и заканчивающаяся 1.) Аналогично случаю целочисленного деления, можно легко найти другие строки данных d'(x), дающие при делении на полином генератора тот же самый остаток, что и d(x). Для этого достаточно просто прибавить к d(x) кратное делителя g(x), либо вставить в d(x) дополнительные блоки, представляющие кратное g(x). Однако при обмене пакетами по USB интерфейсу проблемы подмены шифротекста подслушивающей стороной не стоит, поскольку данные не передаются в широковещательном режиме (либо в эфир) и спецификацией исключено постороннее вмешательство. Вся ответственность за управление и отслеживание корректности передачи лежит на хост-контроллере интерфейса.

.Протоколы связи часто используют два CRC в пакете: один для защиты заголовка последнего и другой для защиты его части данных. Заголовок пакета в USB – поле PID. Поскольку это поле всего-навсего 4-битное, оно защищается полем 4-битной проверки, полученной простой побитовой инверсией поля PID. Это обеспечивает соответствующую защиту от однобитовых и множественных ошибок, не требуя логики CRC генерации и контроля. Порция «данных» USB пакета, которая более протяжённая, защищается обыкновенным полем CRC. В пакетах-признаках (token-пакетах) область, защищённая CRC, всего лишь 11 бит, поэтому 5-битное поле CRC вполне обеспечивает соответствующую защиту, а также выравнивает пакет до байтовой границы. Пакеты данных могут содержать вплоть до 1023 байт, поэтому для защиты данных используется более длинное 16 битное поле CRC с соответствующей модификацией алгоритма.

Реализация CRC в USB

В спецификации USB описаны два генератора полиномов: один для пакетов-признаков и другой для пакетов данных. Полином генератора для token'ов – это x^5 + x^2 + x^0, тогда как для пакетов данных – это x^16 + x^15 + x^2 + x^0. Поскольку остаток всегда меньшей степени, чем полином генератора, CRC признака имеет 5-битовый шаблон, а CRC данных – 16-битовый.

Интуитивным способом построения CRC по входному шаблону могло бы быть простое деление последнего на полином генератора. При делении может использоваться подход, имеющий место при целочисленном делении. Если старшая значимая цифра/бит (MSB) делимого равна 1, делитель вычитается из делимого и получается новое делимое. В арифметике по модулю 2 это вычитание сводится к простому побитовому XOR. Тот же шаг последовательно повторяется для следующего MSB, причём эта операция проводится для всех бит вплоть до последней значимой цифры (LSB). Остаток в этой точке – это остаток от деления входного шаблона, помноженного на x^d, на полином генератора степени d (d равен 5 для CRC пакета-признака и 16 для CRC пакета данных).

Реализация, предусмотренная в USB-спецификации, отличается от вышеупомянутого интуитивного подхода тремя особенностями, которые математически несущественны.

1. Реализация стартует со сдвигового регистра, заполненного «единичками». Иначе, идущие в самом начале пакета «нолики» не защищались бы сгенерированным CRC. Математически это эквивалентно прибавлению к делимому масштабированной константы. Масштабирование – это функция от числа бит во входном шаблоне. На принимающей стороне сдвиговый регистр точно так же заполняется «единичками». Это гарантирует, что на стороне приёмника к делимому прибавляется та же самая масштабированная константа (в предположении, что при передаче не было потеряно ни одного бита). Поэтому, если при передаче не было искажено ни одного бита, с обоих концов канала связи будет сгенерирован один и тот же остаток. В алгебраической форме, результатом масштабирования является делимое D(x) = x^32*F(x) + x^k*L(x), где F(x) – полином (k-1)-ой степени, представляющий k бит потока данных, L(x) – полином (d-1)-ой степени со всеми коэффициентами, равными 1, и d – степень полинома генератора.

2. В реализации с выгодой используется коммутативность и ассоциативность операции XOR. В интуитивном подходе новое делимое получалось вычитанием делителя из делимого. В реализации же это вычитание из делимого делается побитово путём наращения и сдвига остатка.

3. Перед добавлением к входному шаблону в качестве контрольной суммы, остаток побитово инвертируется. Без этой модификации замыкающие нули в конце пакета нельзя было бы обнаружить в качестве ошибок передачи данных. Математически это эквивалентно прибавлению известной константы к остатку. К тому же это математически несущественно для работы CRC. В алгебраической форме: CRC = L(x) + R(x), где R(x) – это остаток, полученный делением D(x) на полином генератора G(x).

Контроль CRC на приёмнике производится точно так же, как генерация CRC по входному шаблону, который теперь состоит из начального входного шаблона со следующим за ним инвертированным остатком. Математически этот новый полином должен полностью делиться полиномом генератора, за исключением остатка, связанного с известной константой, обсуждаемой в пункте 3 (см. выше). (Это может быть интуитивно понятно при осознании, что прибавление остатка к LSB делимого эквивалентно его вычитанию из старого делимого). В алгебраической форме перемещённые и принятые данные – это M(x) = x^32*F(x) + CRC. Когда по этому шаблону M(x) генерируется CRC, остаток R’(x) x^32*L(x)/G(x) и может быть получен из предыдущих равенств и некоторых свойств арифметики по модулю 2. R’(x) – однозначно определяемый полином (т.е. коэффициенты всегда те же самые), поскольку L(x) и G(x) определяются однозначно. R’(x) называется остатком или остаточным полиномом.

Для полинома генератора пакета-признака остаток равен 01100 (или x^4 + x^3); для CRC полинома пакета данных остаток равен 1000000000001101 (или x^15 +x^3 +x^2 + x^0).

Реальное вычисление CRC

Итак, для генерации и проверки сдвиговые регистры изначально полностью заполняются единичками. Далее для каждого бита данных, посылаемого или принимаемого, старший бит текущего остатка (в соответствии со спецификацией) XOR'ится с этим битом и затем остаток сдвигается влево на один бит и младший бит заполняется ноликом. Если результат этого XOR'а равен 1, тогда остаток XOR'ится с полиномом генератора. Когда послан последний бит проверяемого поля, CRC инвертируется в генераторе и посылается проверяющей стороне, начиная со старшего бита (MSb). Когда последний бит CRC принят целевым устройством и не возникло никаких ошибок, остаток должен равняться поиномиальному остатку (см. выше).

Следует отметить, что требования битовой вставки должны выполняться и для CRC. Под этим подразумевается необходимость вставки нуля в конце CRC, если все шесть предшествующих бит равнялись единице. Аналогично, приёмник должен подсчитывать CRC после исключения незначимых нулевых бит битовой вставки.

Пример вычисления CRC

Для закрепления понимания реализации CRC в USB приведём несколько примеров.

Рассмотрим генерацию SOF(start of frame) token'а с временной меткой 47(hex) = 00001000111(bin). Для генерации CRC необходимо использовать CRC5 (с полиномом генератора 100101) от значения номера фрейма.

00001000111

0 |
11111|→ 11110| 0 |
00101|→ 11011|→10110| 0 |
00101|→10011|→00110| 0 | 1 |
00101|→00011|→00110|→01100| 0 |
00101|→01001|→
0 |
10010|→00100| 0 | 1
00101|→00001|→00010|→00100| 1 |
00101|→00001|→00010| 1 |
00101|→00111|→01110|
00101|→01011→

→ инвертируем полученное значение → 10100.

Итак, в итоге CRC5(00001000111) = 10100. Метка времени (номер фрейма), CRC и EOP (End Of Packet) могут быть добавлены к Sync (синхронизация), PID и PID check полям, формируя при этом NRZ (Non Return to Zero) пакет:

00000001101001010000100011110100XX1;

В этом примере NRZ sync-поле – это 00000001, а конец пакета фиксируется как XX1, где X – SE0 (single ended 0) режим шины, когда оба провода дифференциальной пары USB шины находятся в низкоуровневом режиме. Поля PID и PID check – соответственно 0101 и 1010.

Заметим, что вставка бит и NRZI (Non Return to Zero Invert) преобразование выполняются непосредственно перед передачей пакета на USB шину (см. пп.7.1.5 и 7.1.6 спецификации). Поскольку эти операции не влияют на порядок бит при CRC генерации и проверке, они в этой статье не описываются.

Понятно, что алгоритм, основанный на битовых вычислениях, очень медленен и неэффективен. Было бы намного лучше проводить вычисления с целыми байтами. Но в этом случае мы сможем иметь дело лишь с полиномами, имеющими степень, кратную 8 (то есть величине байта). В настоящее время существует три основных алгоритма получения CRC, которые, однако, имеют общую черту - итерационность, то есть последовательное накопление CRC для каждого байта информационного поля. В качестве заключения обсудим их вкратце.

Алгоритмы подсчёта CRC

1) С использованием таблиц

В этом алгоритме присутствует предварительный просчет CRC с помещением результатов в соответствующую таблицу. Индексирование таблицы происходит байтами информационного сообщения, т.е. в ней содержится 256 ячеек, причём длина каждой ячейки совпадает с длиной рассматриваемого CRC. Таким образом, в памяти компьютера под предварительные данные требуется 256 умножить на длину CRC бит.
Например, для CRC16 под таблицу потребуется 1 кБ памяти. Таблица – своя для каждого полинома. Этот алгоритм – самый быстрый из существующих. Применяется он, как правило, если длина CRC совпадает с разрядностью процессора и в компьютере имеется достаточное количество памяти на хранение таблицы. Например, активное использование этих алгоритмов имеет место в протоколах Ethernet.

2) «Простой» итерационный алгоритм

В этом алгоритме информация обрабатывается как битовый поток (см. пример выше), требуется меньшее количество ресурса памяти, т.к. таблицы отсутствуют, но его выполнение занимает времени где-то в 8 раз больше, чем в предыдущем алгоритме. Его применение идёт, как правило, только когда надо получить CRC для нескольких делящих полиномов (генераторов), потому что при этом программа занимает намного меньше памяти. С аппаратной точки зрения простота алгоритма состоит в потребности только в одном сдвиговом регистре. Поэтому, например, этот алгоритм реализуется аппаратно в модемных устройствах.

3)Ускоренный алгоритм с использованием особенностей полинома («Fast»)

Специфичность данного алгоритма заключается в учёте при его составлении полиномиальных коэффициентов (речь идёт о делящем полиноме), т.е. для каждого полинома генератора составляется свой собственный алгоритм. Потребление ресурса *Эссе*

памяти в нём такое же, как и в предыдущем алгоритме, однако выполняется он в 2-3 раза быстрее. Его удобно использовать в обособленных функционально (например, от процессора) системах с ограничениями в производительности, например, в микроконтроллерах на одном чипе. Именно этот алгоритм обычно используется при проектировании USB host контроллера по технологии ПЛИС.

Более подробно с теорией и алгоритмами получения CRC можно ознакомиться в [1].

Список литературы

[1] Ross N.Williams «Элементарное руководство по CRC алгоритмам обнаружения ошибок» / «A painless guide to CRC error detection algorithms», версия 3.0, 19 августа 1993 года.
http://www.rusdoc.ru/material/raznoe/crc.zip

[2] CYCLIC REDUNDANCY CHECKS IN USB
http://www.usb.org/developers/whitepapers/crcdes.pdf

[3] Handbook of Applied Cryptography, by A. Menezes, P. Van Oorschot, and S. Vanstone, CRC Press, 1996.
http://www.cacr.math.uwaterloo.ca/hac