Міністерство освіти та науки України
Запорізька державна інженерна академія

Є.Я. Швець
Є.М. Кісельов

ЗАСОБИ ЗАХИСТУ ІНФОРМАЦІЇ У МЕРЕЖАХ ЕОМ

методичні вказівки до курсового проектування

для студентів ЗДІА
спеціальності 7.090804 «Фізична та біомедична електроніка»

м. Запоріжжя
2005 р.

Міністерство освіти та науки України
Запорізька державна інженерна академія

ЗАСОБИ ЗАХИСТУ ІНФОРМАЦІЇ У МЕРЕЖАХ ЕОМ

методичні вказівки до курсового проектування

для студентів ЗДІА
спеціальності 7.090804 «Фізична та біомедична електроніка»

Рекомендовано до видання
на засіданні кафедри ФБМЕ
протокол №1 від 31.08.05

Засоби захисту інформації у мережах ЕОМ. Методичні вказівки до курсового проектування для студентів ЗДІА спеціальності 7.09.0804 «Фізична та біомедична електроніка». /Укладачі: Є.Я. Швець, Є.М. Кісельов. – Запоріжжя: Вид-во ЗДІА, 2005.- 34 с.

Розглянуті методи та алгоритми виявлення помилок у процесах передачі інформації в мережах ЕОМ.

Укладачі:

Є.Я. Швець, к.т.н., професор
Є.М. Кісельов, ст. викладач

Відповідальний за випуск: ***завідуючий кафедри ФБМЕ***
к.т.н., професор Є. Я. Швець

ЗМІСТ

1 ТЕОРЕТИЧНІ ВІДОМОСТІ

1. При передачі даних по каналах зв'язку існує вірогідність появи помилок. Це зв'язано, перш за все, з тим, що канал передачі інформації звичайно піддається дії різних спотворень, викликаних шумом, який може мати як природну дію, так і створюватися використовуваним обладнанням. Не можна також виключати умисне спотворення даних при несанкціонованому доступі до ліній передачі і передавального обладнання. Процес захисту від помилок включає виявлення і виправлення помилок.

2. Існує два основні підходи до виправлення помилок:

а) перший базується на використанні різних методів локалізації і виправлення помилок в приймаючому пристрої (реалізується достатньо рідко);

б) другий використовує методи визначення лише факту появи помилок, "виправлення" яких здійснюється за рахунок повторної передачі повідомлення (реалізується часто).

3. Існують різні методи виявлення помилок, більшість з яких перетворить повідомлення, доповнюючи його надмірною інформацією (методи контрольних сум). Методи CRC (Cyclic Redundancy Codes) відносяться до методів, які залишають без зміни початковий текст повідомлення, додаючи до повідомлення контрольну суму.

4. Метод контролю парності полягає в тому, що для щодо невеликого блоку біт даних обчислюється контрольна сума, яка передається і зберігається разом з даними у вигляді біта парності. При прочитуванні даних контрольна сума обчислюється ще раз і порівнюється з бітом парності. Якщо вони співпадають, то дані вважаються істинними, інакше генерується повідомлення про помилку парності, яке приводить до зупинки комп'ютера.

Існують два основні підходи для цього методу:

а) коли встановлений *контроль по парності (Even),* то біт парності приймає таке значення, щоб разом з ним кількість двійкових одиниць «1» (Mark) в блоці даних, що захищається, була парною.

б) коли встановлений *контроль по непарності (Odd),* то біт парності приймає таке значення, щоб разом з ним кількість двійкових одиниць «1» (Mark) в блоці даних, що захищається, була непарною.

Цей метод використовується тільки для щодо невеликих блоків інформації, наприклад, байтів, що зберігаються в оперативній пам'яті комп'ютерів, оскільки має наступний недолік, вірогідність якого зростає із зростанням розміру блоку інформації, що захищається: метод ніяк не застрахований від подвійних помилок (одночасна зміна значень двох біт не впливає на біт парності). Також до втрати інформації може привести помилка в біті парності.

5. Найбільш простим способом перевірки цілісності даних, передаваних в цифровому представленні, є метод контрольних сум**.** Під контрольною сумою розуміється деяке значення, розраховане шляхом складання всіх чисел з вхідних даних. Оскільки часто сума всіх чисел перевищує максимально допустиме значення, наперед задане для цієї величини (розміром поля контрольної суми), то контрольну суму звичайно розраховують як залишок від ділення підсумкової суми на максимально можливе значення контрольної суми, збільшене на одиницю (такий алгоритм часто називають доповненням контрольної суми до одиниці):

$$Checksum = Total /(MaxVal + 1),$$

де Total - підсумкова сума, розрахована за вхідними даними

MaxVal - максимально допустиме значення контрольної суми, задане наперед.

Наприклад, нехай блок інформації, що захищається, є наступною послідовністю величин завдовжки 10 байт: 36 211 163 4 109 192 58 247 47 92 і розмір поля, відведеного для контрольної суми, складає один байт (тобто

поля, відведеного для контрольної суми, складає один байт (тобто максимальне число, яке можна в нього занести, рівне 255). Сума даних блоку рівна 1159, а залишок від ділення суми на 255+1=256 (контрольна сума) дорівнює 135. Якщо контрольна сума, розрахована відправником інформації, дорівнювала, скажімо, 246, а після отримання вона має значення 135, означає, інформація піддалася зміні.

Недолік методу контрольних сум полягає в тому, що неспівпадання значень цих сум служить вірним доказом, що даний документ піддався зміні, рівність порівнюваних значень ще не дає гарантії, що інформація залишилася незмінною. Можна довільним чином змінити порядок чергування чисел в документі, а контрольна сума при цьому збереже колишнє значення. І що ще гірше - можна змінити окремі числа в документі і підігнати інші так, щоб забезпечити колишнє значення контрольної суми. При використанні для контрольних сум 8-розрядної змінної вірогідність того, що контрольні суми двох абсолютно випадково вибраних послідовностей даних будуть однакові, рівна 1/256. При збільшенні довжини змінної під контрольну суму до 16 або 32 розрядів, вірогідність збігів зменшується, проте цей механізм все одно дуже чутливий до можливих помилок, щоб забезпечити високий ступінь довіри до представлених даних.

6. Більш досконалий спосіб цифрової ідентифікації блоків даних - це обчислення її циклічного надмірного коду (Cyclic Redundancy Code - CRC). Цей метод широко використовується при захисті інформації в комп'ютерних мережах. Наприклад, останнє поле в кадрі Ethernet - це чотирьохбайтне поле контрольної послідовності кадру (Frame Check Sequence, FCS). Значення цього поля обчислюється на основі вмісту заголовка і даних (разом із заповнювачем, але без урахування преамбули і обмежувача) за допомогою 32-розрядного циклічного надмірного коду (CRC-32) по наступній формулі (у двійковій системі числення):

$$\text{контрольна послідовність} = \text{MOD}(\text{дані}/\text{поліном}),$$

де MOD - операція розрахунку залишку від ділення;

дані - бітова послідовність блоку даних, що захищається, поліном - так званий утворюючий поліном, відомий і відправнику, і одержувачу.

Утворюючим поліномом називається мінімальний поліном з сімейства поліноміальний простих поліномів (тобто що не діляться поліноміальний без остачі на будь-яке ціле число, окрім 2 і самого себе). Такий поліном має властивості лінійності (сума по модулю 2 (XOR) будь-яких двох поліномів сімейства призводить до отримання полінома з цього ж сімейства) і циклічності (циклічний зсув будь-якого полінома сімейства на будь-яку кількість розрядів призводить до здобуття полінома цього ж сімейства). Утворюючий поліном володіє властивістю "розмазувати" зміну в деякому біті масиву даних на можливо більше число біт контрольної суми і тим самим підвищувати вірогідність детектування помилки. Оборотність операції XOR дозволяє використовувати один і той же алгоритм при обчисленні суми і відправником, і одержувачем (див. далі), крім того, відносно просто алгоритму реалізовується електронними схемами.

Хоча арифметичне ділення дуже схоже на схему розрахунку контрольної суми, званої CRC, сам же алгоритм CRC декілька складніший, і, щоб зрозуміти його, необхідно розглянути теорію цілих чисел. Всі CRC алгоритми засновані на поліноміальних обчисленнях, і для будь-якого алгоритму CRC можна вказати, який поліном він використовує. Замість представлення дільника, ділимого (повідомлення), приватного і залишку у вигляді позитивних цілих чисел, можна представити їх у вигляді поліномів з двійковими коефіцієнтами або у вигляді рядка біт, кожний з яких є коефіцієнтом полінома. Наприклад, десяткове число 23 в шістнадцятиричній системі числення має вигляд 17, а в двійковому – 10111, що співпадає з поліномом:

1x4 + 0x3 + 1x2 + 1x1 + 1x0

або, спрощено:

$$\mathbf{x^4 + x^2 + x^1 + x^0}$$

І повідомлення, і дільник можуть бути представлені у вигляді поліномів, з якими можна виконувати будь-які арифметичні дії. Припустимо, що необхідно перемножити, наприклад, 1101 і 1011. Це можна виконати, як множення поліномів:

$$\mathbf{(x^3 + x^2 + x^0)(x^3 + x^1 + x^0) = (x^6 + x^4 + x^3 + x^5 + x^3 + x^2 + x^3 + x^1 + x^0) =}$$
$$\mathbf{= x^6 + x^5 + x^4 + 3x3 + x^2 + x^1 + x^0}$$

Тепер для отримання правильної відповіді необхідно вказати, що X рівний 2, і виконати перенесення біта від члена $\mathbf{3x^3}$. В результаті одержимо:

$$\mathbf{x^7 + x^3 + x^2 + x^1 + x^0}$$

Такий спосіб обчислень схожий на звичайну арифметику, з тією лише різницею, що підстава у передбачається, а не строго задано. Т.ч., якщо вважати, що "X" не відомий, то неможливо виконати перенесення, оскільки не відомо, що $\mathbf{3x^3}$ – це те ж саме, що і $\mathbf{x^4 + x^3}$, оскільки не відомо, що $\mathbf{X = 2}$. У поліноміальній арифметиці зв'язку між коефіцієнтами невизначені, і, тому, коефіцієнти при кожному члені полінома стають такими, що строго типізуються — коефіцієнт при $\mathbf{x^2}$ має інший тип, чим при $\mathbf{x^3}$. Якщо коефіцієнти кожного члена полінома цілком ізольовані один від одного, то можна використовувати будь-які види поліноміальної арифметики, міняючи правила, по яких коефіцієнти працюють. Одна з таких схем використовується в теорії CRC: коефіцієнти складаються по модулю 2 без перенесення – тобто коефіцієнти можуть мати значення лише 0

або 1, перенесення не враховується. Це називається "поліноміальна арифметика по модулю 2". Повертаючись до попереднього прикладу:

$$(x^3 + x^2 + x^0)(x^3 + x^1 + x^0) = (x^6 + x^4 + x^3 + x^5 + x^3 + x^2 + x^3 + x^1 + x^0) =$$
$$= x^6 + x^5 + x^4 + 3x3 + x^2 + x^1 + x^0$$

За правилами звичайної арифметики, коефіцієнт члена **3x3** розподіляється по інших членах полінома, використовуючи механізм перенесення і припускаючи, що **X = 2**. У "поліноміальній арифметиці по модулю 2" не відомо, чому рівний "X", перенесень тут не існує, а всі коефіцієнти розраховуються по модулю 2. В результаті одержуємо:

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

Існує схожість між поліноміальною арифметикою і арифметикою високої точності (multiple precision arithmetic), коли основа b замінюється на x. Основна відмінність полягає в тому, що коефіцієнт uk члена xk в поліноміальній арифметиці вважається малою величиною, або величиною, не пов'язаною з найближчими коефіцієнтами $xk_{\ 1}$ (і x_{k+1}), тому перенесення від одного члена до іншого в ній відсутній. Фактично, поліноміальна арифметика по модулю b багато в чому аналогічна арифметиці високої точності по основі b за винятком придушення всіх перенесень. Т.ч., поліноміальна арифметика по модулю 2 – це фактично двійкова арифметика по модулю 2 без урахування перенесень. Хоча поліноми мають зручні математичні засоби для аналізу CRC алгоритмів і алгоритмів корекції помилок, для спрощення вони будуть замінені на безпосередні арифметичні дії в системі, з якою вони ізоморфні, а саме в системі двійкової арифметики без перенесень.

Дії, що виконуються під час обчислення CRC, є арифметичними операціями без урахування перенесень. Це називається поліноміальною арифметикою.

Складання двох чисел в CRC арифметиці повністю аналогічно звичайному арифметичному дії за винятком відсутності перенесень з розряду в розряд. Це означає, що кожна пара бітів визначає результат свого розряду незалежно від результатів інших пар. Наприклад:

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

Для кожної пари бітів можливі 4 варіанти:

```
0+0=0
0+1=1
1+0=1
1+1=0
```

Те ж саме справедливо і для віднімання:

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

коли є також 4 можливих комбінації:

```
0-0=0
0-1=1
1-0=1
1-1=0
```

Фактично, як операція складання, так і операція віднімання в CRC арифметиці ідентичні операції що "Виключає АБО" (eXclusive OR – XOR), що дозволяє замінити 2 операції першого рівня (складання і віднімання) однією дією,

яка, одночасно, виявляється інверсною самому собі. Згрупувавши складання і віднімання в одну єдину дію, CRC арифметика виключає з поля своєї уваги всі величини, лежачі за межами самого старшого свого біта. Хоча ясно, що значення 1010 більше, ніж 10, це виявляється не так, коли говорять, що 1010 повинне бути більше, ніж 1001. Спробуємо одержати з 1010 значення 1001, віднявши або додавши до нього одну і ту ж величину:

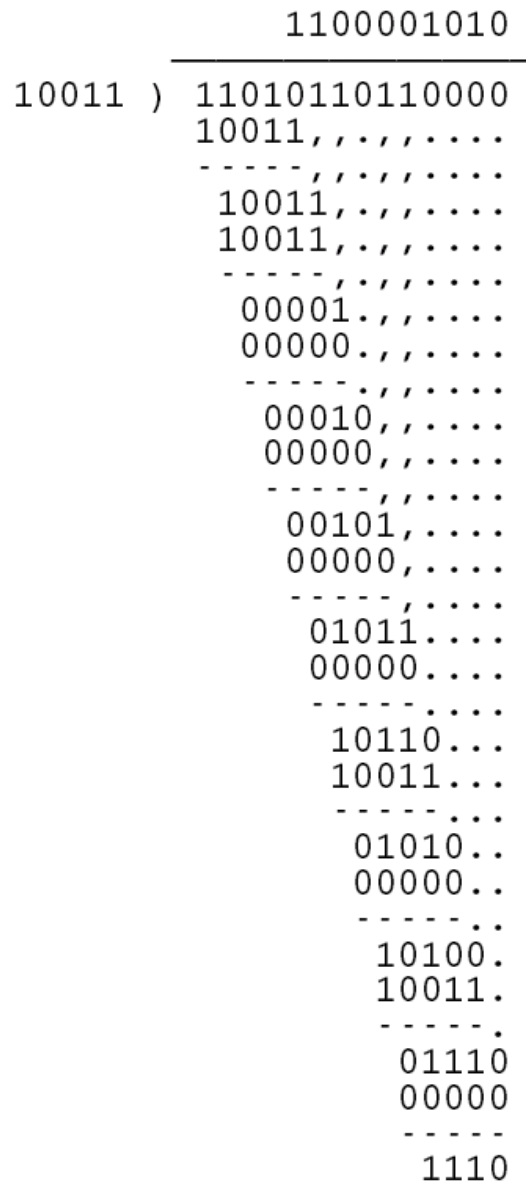
```
1001 = 1010 + 0011
1001 = 1010 - 0011
```

Ця властивість виключає уявлення про порядок.

Множення, як і в звичайній арифметиці, вважається сумою значень першого співмножника, зсунутих відповідно до значення другого співмножника.

```
  x 1101
    1011
    ----
    1101
   1101.
  0000..
 1101...
 -------
 1111111
```

Ділення дещо складніше, так потрібно знати, коли "одне число перетворюється на інше ". Щоб зробити це, потрібно буде ввести слабке поняття розмірності: число X більше або рівне Y, якщо позиція самого старшого одиничного біта числа X більш значуща (або відповідає) позиції самого старшого одиничного біта числа Y. Нижче приведений приклад ділення :



Якщо дії складання і віднімання в CRC - арифметиці – це одне і те ж, то у такому разі **A+0=A** і **A-0=A**. Якщо число A одержане множенням числа B, то в CRC арифметиці це означає, що існує можливість сконструювати число A з нуля, застосовуючи операцію XOR до B, зсунутому на різну кількість позицій. Наприклад, якщо A рівне 0111010110, а B – 11, то ми може сконструювати A з B наступним способом:

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

Проте, якби A було б рівне 0111010111, то не вдалося б скласти його за допомогою різних зсувів числа 11. Тому в CRC арифметиці воно не ділиться на B. Таким чином, CRC арифметика зводиться головним чином до операції що "Виключає АБО" деякого значення при різних величинах зсуву.

Передане повідомлення T є добутком полінома. Зверніть увагу, що

1) останні W біт повідомлення – це залишок від ділення доповненого нулями початкового повідомлення на вибраний поліном;

2) складання рівносильне відніманню, тому збільшення залишку доповнює значення повідомлення до наступного повного добутку.

Якщо повідомлення при передачі було пошкоджено, то одержують повідомлення $T \oplus E$, $T \oplus E$, де E – це вектор помилки, а $\oplus$ – це CRC складання (або операція XOR). Одержавши повідомлення, приймач ділить $T \oplus E$, $T \oplus E$ на G. Оскільки T mod G = 0, (TE) mod G = E mod G. Отже, якість полінома визначатиметься набором добутків G, оскільки у разі, коли E також є твором G, така помилка виявлена не буде. Отже, завдання полягає в тому, щоб знайти такі класи G, добутки яких будуть якомога менше схожі на шуми в каналі передачі (які і викликають пошкодження повідомлення). Розглянемо, які типи шумів в каналі передачі ми можемо чекати.

Однобітові помилки. Помилка такого роду означає, що E=1000...0000. Можна гарантувати, що помилки цього класу завжди буде розпізнані за умови, що в G принаймні 2 біти встановлені в "1". Будь-який добуток G може бути сконструйований операціями зрушення і складання, і, в теж час, неможливо набути значення з 1 одиничним бітом, зрушуючи і складаючи величину, що має більше 1 одиничного біта, оскільки в результаті завжди присутній, принаймні, 2 біта.

Двобітові помилки. Для виявлення будь-яких помилок вигляду 100...000100...000 (тобто коли E містить принаймні 2 одиничних біта) необхідно вибрати таке G, які б не мало множників 11, 101, 1001, 10001, і так далі.

Помилки з непарною кількістю біт. Можливо визначити будь-які пошкодження, коли E має непарне число біт, вибравши поліном G таким, щоб він мав парну кількість біт. Це визначається наступними властивостями:

1) CRC множення є простою операцією XOR постійного регістрового значення з різними зсувами;

2) XOR – це операція перемикання бітів;

3) якщо застосовується в регістрі операцію XOR до величини з парним числом бітів, парність кількості одиничні бітів в регістрі залишиться незмінною. Наприклад, почнемо з E=111 і спробуємо скинути все 3 біта в "0" послідовним виконанням операції XOR з величиною 11 і одним з 2 варіантів зрушень (тобто, "E=E XOR 011" і "E=E XOR 110"). Це аналогічно завданню про перевертання стаканів, коли за одну дію можна перевернути одночасно будь-які два стакани. Більшість популярних CRC поліномів містять парну кількість одиничних бітів.

Пакетні помилки. Пакетна помилка виглядає як E=000...000111...11110000...00, тобто E складається з нулів за винятком групи одиниць десь у середині. Цю величину можна перетворити в E=(10000...00)(1111111...111), де є z - нулів в лівій частині і n - одиниць в правій. Для виявлення цих помилок необхідно встановити молодший біт G в 1. При цьому необхідно, щоб ліва частина не була множником G. При цьому завжди, поки G ширше за праву частину, помилка завжди буде розпізнана. Вірогідність пакетної помилки з шириною, більшою, ніж W, рівна (0.5) W. На цьому ми завершимо розділ, присвячений мистецтву вибору поліномів.

У Ethernet утворюючим поліномом (CRC-32) служить багаточлен:

$$x^{32}+x^{26}+x^{23}+x^{22}+x^{l6}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x+l,$$

що визначає бітову послідовність 10000010011000001000111010110111. Цей код дозволяє виявити ~ 99,9999% всіх помилок в повідомленнях завдовжки до 64 байт. Таким чином, вірогідність того, що приймаюча станція сприйме зіпсований кадр як цілий, практично рівна нулю. Алгоритм контролю CRC вже протягом тривалого часу широко використовується в системах мережевих адаптерів, контролерів жорсткого диска і інших пристроїв для перевірки ідентичності вхідної і вихідної інформації. Також цей механізм застосовується в багатьох комунікаційних програмах і архіваторах.

Так само, як і для контрольних сум, CRC коду не потрібно багато місця (звичайно їх довжина складає 16 або 32 розряду); проте в порівнянні з ними, надійність виявлення невеликих змін вхідної інформації тепер значно вище. Якщо в деякому величезному блоці даних лише один розряд став іншим, то і контрольний параметр CRC з 100-процентною вірогідністю також матиме інше значення. Якщо ж змінять два розряди, то вірогідність виявлення помилки при довжині параметра CRC в 16-розрядів, складає більше 99,99%. На відміну від контрольних сум метод CRC зможе розпізнати перестановку двох байт, додавання 1 до одного байт і відніманню 1 з іншого і інші методи фальсифікації.

Послідовність кодування CRC:

1. Початкова бітова послідовність даних зсовується вліво на m розрядів, де m - найбільший ступінь утворюючого полінома (для CRC-32 рівна 32 біта).

2. Виконується побітове ділення (шляхом застосування операції XOR) одержаної зсунутої інформаційної послідовності на утворюючий поліном g(x) до неподільного залишку (довжина залишку не повинна перевищувати m біт).

3. Зсунута послідовність об'єднується із залишком, в результаті виходить інформаційна послідовність з CRC кодом.

4. Перевірка цієї послідовності одержувачем здійснюється діленням інформаційної послідовності з CRC кодом на той же утворюючий поліном. У ви-

падку, якщо залишок від ділення залишок рівний 0, вважається, що помилок в одержаних послідовності і CRC коді немає, якщо ж залишок не рівний нулю, отже, спотворені або інформаційна послідовність, або CRC код і повинна бути виконана повторна передача.

2 ПРИКЛАД РОЗРАХУНКУ CRC КОДУ

1. Нехай початкова послідовність 10110101, а утворюючий поліном виберемо рівним 1011 (x^3+x+1).

2. Виконуємо зсув послідовності на максимальний ступінь утворюючого полінома - 3 біта: 10110101000.

3. Виконуємо ділення (побітове XOR) на утворюючий поліном:

```
10110101000 | 1011
1011        | 100001
000001010
    1011
    000100
```

4. Залишком (кодом CRC-3) є 100. Об'єднуємо початкову послідовність і CRC-3 код. Одержуємо інформаційну послідовність з кодом, готову до передачі: 10110101100.

5. Одержувач, одержавши послідовність, виконує її ділення (побітове XOR) на той же утворюючий поліном:

```
10110101100 | 1011
1011        | 100001
000001011
    1011
    000000
```

Оскільки залишок від ділення рівний нулю, можна зробити висновок про те, що при передачі інформація не піддалася зміні.

3 АЛГОРИТМИ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ CRC

3.1 Пряма реалізація CRC

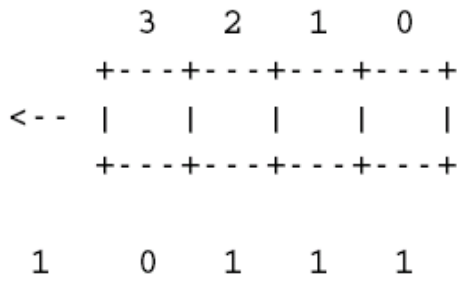
Розглянемо абсолютно примітивну малоефективну реалізацію алгоритму. Потім поступово перетворимо програму до тих пір, поки не одержимо компактний табличний код. Але спочатку нам необхідно створити програму, що виконує CRC ділення. Існують принаймні 2 причини, по якій не можна використовувати звичайні машинні інструкції ділення, що є на комп'ютерах.

1) необхідно виконувати операцію за правилами CRC арифметики.

2) ділиме може мати розмір в десятки мегабайтів, а сучасні процесори не мають таких регістрів.

Т.ч., при виконанні операції CRC ділення необхідне повідомлення по частинах записувати в регістр. На цьому етапі розробки потрібно поводитися з даними абсолютно точно. У всіх подальших прикладах як повідомлення розглядається потік байтів по 8 бітів кожен, з яких сьомий біт вважатиметься самим значущим бітом. Потік бітів, одержаний з цих байтів, передаватиметься, починаючи з 7 біта, і закінчуючи 0 бітом першого байта, потім другого і так далі.

Для прикладу візьмемо поліном 4-й ступені (W=4), і хай цим поліномом буде 10111. У такому разі для виконання ділення нам буде потрібно 4 бітовий регістр



Ділення виконується наступним способом:

Завантажимо регістр нульовими бітами
Доповнимо хвостову частину повідомлення W нульовими бітами
While (поки що є необроблені біти)
Begin
Зрушимо регістр на 1 біт вліво і помістимо черговий ще не оброблений біт з повідомлення в 0 позицію регістра.
If (з регістра був висунутий біт із значенням "1")
Регістр = Регістр XOR Поліном
End
Тепер в регістрі міститься залишок

(***Зауваження***: На практиці умова IF може бути протестована за змістом старшого біта регістра до виконання операції зрушення.)

Назвемо таку реалізацію "простій". Фактично алгоритм можна представити операціями віднімання полінома в різних ступенях (що досягалося операціями зрушення) з початкового повідомлення до тих пір, поки в результаті не залишилося нічого, окрім залишку.

3.2 Реалізація табличного алгоритму

Розглянутий вище "простий" алгоритм є хорошою відправною крапкою, оскільки він безпосередньо пов'язаний з теорією, і тому, що він дійсно простий. Проте, унаслідок того, що він працює на рівні бітів, його достатньо важко закодувати, а ефективність його роботи низка, весь цикл виконується по кожному біту. Для збільшення швидкості роботи необхідно знайти спосіб примусити алгоритм працювати з блоками, більшими ніж біт. Такими блоками можуть бути півбайти (4 біта), байти, (8 біт), слова (16 біт) і довгі слова (32 біта), і навіть довші фрагменти. Півбайти не розглядаються, оскільки вони не вирівняні на границю байта. Для збільшення швидкості роботи програми

ницю байта. Для збільшення швидкості роботи програми необхідно, щоб інформаційні блоки були вирівняні на границю байта, і, фактично, більшість табличних алгоритмів оперують за один цикл саме з байтом. Для простоти давайте перейдемо від 4 бітового полінома до 32 бітовому. Регістр виглядатиме точно також, як і раніше, з тією лише різницею, що осередки тепер представляють не біти, а байти, а поліном складатиметься з 33 бітів (один одиничний біт, що мається на увазі, і 32 "активних" біта)

Спочатку використовуємо "простий" алгоритм. Припустимо, що алгоритм повністю перероблений на роботу з байтами, а старші 8 біт 32 бітові регістри (3 байт) мають значення:

```
t7 t6 t5 t4 t3 t2 t1 t0
```

Під час чергового циклу "простого" алгоритму значення "t7" визначає, чи буде виконана операція що "Виключає АБО" (XOR) полінома зі всім вмістом регістра. Якщо опиниться, що t7=**1,** то операція виконується, інакше вона буде пропущена. Припустимо також, що старші 8 біт полінома мають значення g7 **g6.g0,** тоді до наступного циклу старший байт матиме наступний вміст:

```
      t6 t5 t4 t3 t2 t1 t0 ??
⊕
t7 * (g7 g6 g5 g4 g3 g2 g1 g0)
```

"Новий" старший біт (який визначатиме, що відбудеться в наступному циклі) тепер має значення **t6 ⊕ t7*g7**. Зверніть увагу, що з інформаційної точки зору все, що потрібний для обчислення значення "нового" старшого біта, міститься в 2 старших бітах початкового старшого байта. Точно також, "наступний" старший біт може бути наперед обчислений, виходячи їх 3 старших бітів **t7, t6** і **t5**. У загальному випадку величину старшого біта регістра на до циклі

можна обчислити з до - старших бітів регістра. Допустимо, що старші 8 біт регістра використовуються для розрахунку значення старшого біта регістра на 8-му циклі. Припустимо, що наступні 8 циклів виконуються, використовуючи передобчислені значення (які можна записати в однобайтовий регістр і висувати з нього послідовно по 1 біту). Тоді можна відмітити, що:

1) Величина старшого байта регістра тепер не матиме ніякого значення. Не матиме значення, скільки разів і з яким зрушенням полінома буде виконана операція XOR, оскільки всі біти так чи інакше будуть висунуті з правої частини регістра за подальші 8 циклів.

2) Біти регістра, що все залишилися, опиняться зрушені вліво на 1 позицію, а найправіший байт регістра буде пересунутий в позицію наступного байта ліво його.

3) Регістр опиниться піддадуть серії операцій XOR відповідно до значень бітів передобчисленого керівника байта.

Тепер розглянемо вплив на регістр операцій XOR з постійною величиною при різних зрушеннях. Наприклад:

```
0100010
...0110
..0110.
0110...
-------
0011000
```

З цього прикладу можна зробити наступний висновок: при виконанні операції XOR регістра з постійною величиною при різних її зрушеннях завжди існуватиме деяке значення, яке при застосуванні операції що "виключає АБО" з початковою величиною регістра дасть той же самий результат.

Т.ч., одержуємо:

While (повідомлення повністю не оброблене)

Begin

Перевіримо старший байт регістра

Розрахуємо на його основі контрольний байт

Грунтуючись на значенні контрольного байта, виконаємо операцію XOR нашого полінома з різними зсувами

Зрушимо регістр на один байт вліво, додавши справа новий байт з повідомлення

Виконаємо операцію XOR вмісту регістра з підсумовуваним поліномом

End

Результат вийшов не краще, ніж у разі "простого" алгоритму. Але велику частину обчислень можна провести наперед, а результати зібрати в таблицю. Тоді одержимо:

While (повідомлення повністю не оброблене)

Begin

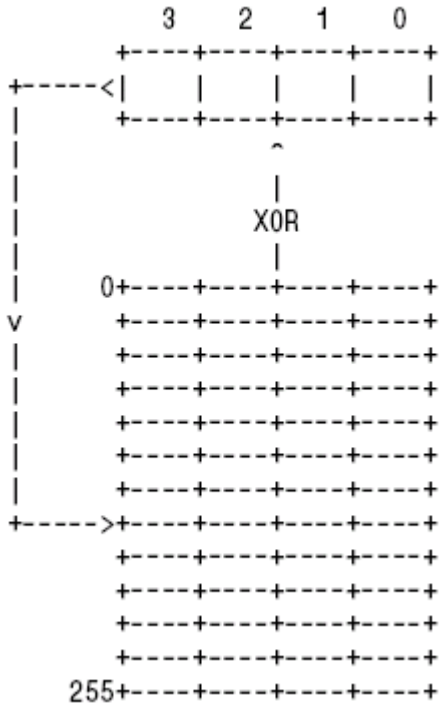
Top = Старший_байт(Register);

Register = (Register << 24) | Новый_байт_сообщения***;***

Register = Register XOR Рассчитанная_таблица[Top];

End

Це дуже ефективний алгоритм, що вимагає тільки операцій зрушення, OR, XOR, а також операції пошуку в таблиці. Графічно він виглядає так.

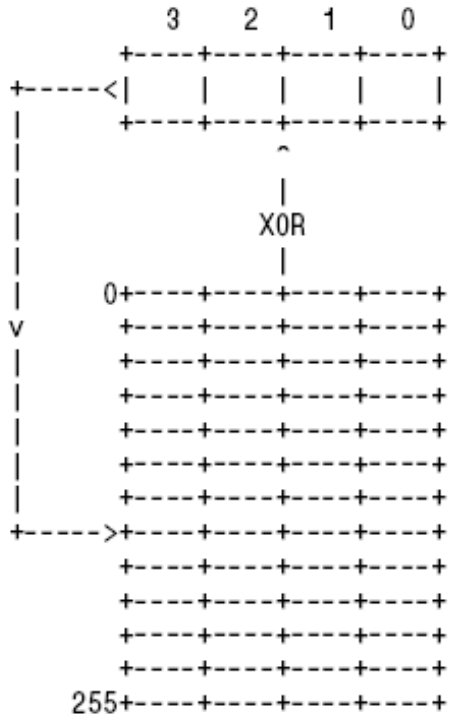


Це дуже чіткий і ефективний цикл, хоча він здається таким, що не вписується в теорію CRC. Умовимося називати його "Табличним" алгоритмом.

3.3 Перетворений табличний алгоритм

Розглянутий вище цикл працює з вирівняним повідомленням, тобто для його використання нам необхідно доповнити повідомлення W/8 нульовими байтами перш, ніж передавати покажчик алгоритму. Залежно від обстановки це може і не скласти великої проблеми, проте, якщо оброблюваний блок даних передається нам іншою програмою, таке доповнення може виявитися проблематичним. Одним з варіантів є просте додавання після основного циклу коду, оброблювального нульові байти.

Проте, пожертвувавши для ясності простотою алгоритму, тепер можна перетворити все так, щоб уникнути і необхідності доповнення повідомлення нульовими байтами, і обробки кожного додаткового нульового байта окремо. Для пояснення цієї можливості повернемося до вже розглянутої нами діаграми.



Обговоримо спочатку деякі складові частини оброблюваного повідомлення. *Хвіст*: W/8 додаткових нульових байта, які з'являються на останньому етапі роботи алгоритму, щоб бути вставленими в регістр у міру завершення обробки самого повідомлення. Проте, вони (нулі) не надають якого або впливу на регістр, оскільки

1) операція XOR з нульовим вмістом не міняє значення другого операнда;

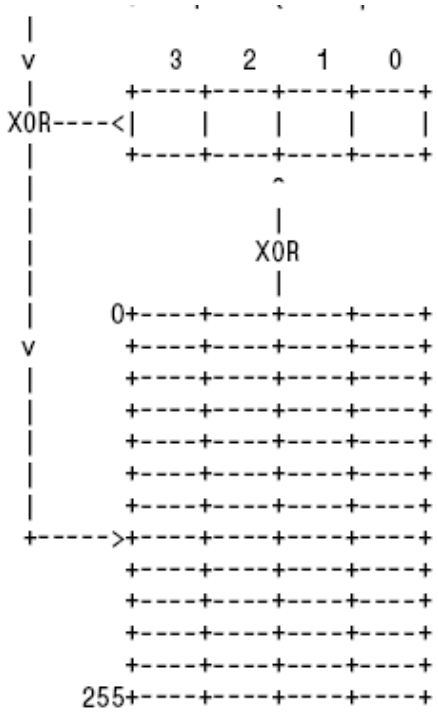
2) ці 4 нульових байта ніколи не висуваються з лівої частини регістра, де їх вміст міг би надати хоч яке - та дія.

Отже, єдине призначення цих нульових байтів полягає в тому, щоб продовжити роботу алгоритму W/8 циклів і дозволити кінцевим байтам оброблюваного повідомлення пройти через весь регістр.

Заголовок: Якщо початковий вміст регістра рівний нулю, то початкові 4 цикли всього лише завантажить перші 4 байти повідомлення в регістр, зрушуючи їх справа наліво. Це відбувається тому, що перші 32 контрольних біта будуть рівні нулю, отже в результаті операції XOR вміст регістра мінятися не буде. У разі ж, коли в регістрі спочатку міститься ненульове значення, в процесі

перших 4 циклів все також відбуватиметься заповнення регістра 4 байтами повідомлення з подальшою операцією що "Виключає АБО" з деяким постійним значенням, яке цілком залежить від початкового вмісту регістра.

Ці факти в сукупності з асоціативною властивістю операції що "Виключає АБО" **(A xor B) xor C = A xor (B xor C)** означають, що байтам оброблюваного повідомлення немає ніякої необхідності продиратися через W/8 байтів регістра. Натомість вони можуть бути безпосередньо скомбіновані операції XOR із старшим байтом регістра з використанням результату як табличний індекс. Т.ч., одержимо наступну модифікацію алгоритму.



Зауваження: Початкове значення регістра в даному випадку повинне бути тим же самим, що і початкове значення регістра попереднього алгоритму, пропущене 4 рази через таблицю. Таблиця ж така, що якщо попередній алгоритм використовував "0", то і новий алгоритм також використовуватиме це ж значення.

Ці алгоритми ідентичні, і вони дають ідентичні результати.

4 ПОРЯДОК ВИКОНАННЯ РОБОТИ

1. Отримати завдання для теоретичної частини, згідно додатку.

2. Запишіть перші десять символів свого прізвища (доповнивши при необхідності її першими символами імені) в шістнадцятирічній і двійковій системі обчислення відповідно до ASCII таблиці і занесіть одержану інформацію в наступну таблицю:

Символ	**Шістнадцятирічний код**	**Двійковий код**	**Біт парності**

У четвертий стовпчик занесіть біт парності для кожного символу так, щоб якщо Ваше прізвище значиться в журналі групи під парним номером, сума біт символу була парною, а якщо Ваше прізвище значиться в журналі групи під непарним номером, сума біт символу була непарною.

3. Запишіть своє прізвище і ім'я парами символів і представте ці пари в шістнадцятирічній системі. Якщо кількість символів непарне доповнити останній символ байтом 00_H. Розрахуйте контрольну суму методом 16-річного доповнення до одиниці.

4. Запишіть перші п'ять символів свого прізвища (доповнивши при необхідності її першими символами імені) у вигляді бітової послідовності і розрахуйте контрольну суму методом CRC з використанням утворюючого полінома CCITT-CRC-16: $\mathbf{X^{16} + X^{12} + X^5 + X}0$

5. Складіть алгоритм, програмний код реалізації розрахунку та перевірки CRC згідно завдання п. 4.

5 ВИМОГИ ДО ЗМІСТУ РОБОТИ

1. Титульний аркуш
2. Завдання до курсового проекту
3. Реферат
4. Зміст
5. Теоретична частина
6. Розрахункова частина
7. Програмна реалізація
8. Висновки
9. Перелік використаних джерел
10. Додатки (програмний код, результати виконання розрахунку CRC за допомогою складеної програми)
11. Графічна частина: аркуш формату А1 – графічне зображення алгоритму розробленого програмного забезпечення, аркуш формату А3 – зображення вказується керівником.

ПЕРЕЛІК РЕКОМЕНДОВАНИХ ДЖЕРЕЛ

1. Дж. Л. Месси. Введение в современную криптологию. // ТИИЭР, т.76, №5, Май 88 – М, Мир, 1988, с.24-42.
2. У. Диффи. Первые десять лет криптографии с открытым ключом. // ТИИЭР, т.76, №5, Май 88 – М, Мир, 1988, с.54-74.
3. А. В. Спесивцев и др. Защита информации в персональных компьютерах. – М., Радио и связь. 1992, с.140-149.
4. В. Жельников. Криптография от папируса до компьютера. – М., ABF, 1996.
5. Вирт Н. Алгоритмы + структуры данных = программы. М.: Мир, 1985.
6. Галлагер Р. Теория информации и надежная связь. М.: Советское радио, 1974.
7. Дэвис Д., Барбер Д., Прайс У., Соломонидес С. Вычислительные сети и сетевые протоколы. М.:Мир, 1982.
8. Кнут Д. Искусство программирования для ЭВМ. Т. 3. Сортировка и поиск. М.: Мир, 1978.
9. Баричев С. Криптография без секретов. М.: "ДИАЛОГ-МИФИ", - 1995.
10. Вакка Дж. Секреты безопасности в Internet. – К.: Диалектика, 1997.
11. http://www.citforum.ru/security/
12. Кулаков Ю.А., Луцкий Г.М. Компьютерные сети.-К. Юниор, 1998.-384с.
13. ТаненбаумЭ. Компьютерные сети. - СПб.: Питер, 2002. - 848 с.: ил.
14. Williams Ross N. Элементарное руководство по CRC-алгоритмам обнаружения ошибок ftp ://www. internode .net. au/clients/rocksoft/papers/crc_v3. txt
15. Романец Ю.В., Тимофеев П.А., Шаньгин В.Ф. Защита информации в компьютерных системах и сетях/Под ред. В.Ф.Шаньгина.-М.: Радио и связь, 1999.-328с.
16. Мельников В.В. Защита информации в компьютерных системах. -М.: Финансы и статистика; Электроинформ, 1997.-368с.

17. Ведев Д Л. Защита данных в компьютерных сетях, http://www.osp.ru.

18. Решения компании Cisco Systems по безопасности компьютерных систем - Cisco SAFE http ://www.cisco.com/warp/public/779/largeent/issues/security/safe.htm 1

ДОДАТОК

ПЕРЕЛІК ЗАВДАНЬ ДЛЯ ТЕОРЕТИЧНОЇ ЧАСТИНИ

1. Шифр «скітала»
2. Шифруючи таблиці
3. Шифрування подвійною перестановкою
4. Магічні квадрати
5. Полібіанській квадрат
6. Система шифрування Цезаря
7. Афінна система підстановок Цезаря
8. Система Цезаря з ключовим словом
9. Що шифрують таблиці Трісемуса
10. Біграммний шифр Плейфера
11. Шифр Гронсфельда
12. Система шифрування Віжінера
13. Шифр „подвійний квадрат” Уїтстона
14. Одноразова система шифрування
15. Шифрування методом Вернама
16. Що роторні шифрують машини
17. Метод гаммування
18. Стандарт DES
19. Скремблери
20. Мережі Фейштеля
21. Блочний шифр TEA
22. Конкурс AES
23. Шифр MARS
24. Шифр RC5
25. Шифр RC6
26. Шифр Serpent

27. Шифр BlowFish
28. Шифр TwoFish
29. Шифр Rijndael
30. Алгоритми створення ланцюжків
31. Методи рандомізації повідомлень
32. Генератори випадкових і псевдовипадкових послідовностей
33. Алгоритм Хаффмана
34. Алгоритм Лемпеля-Зіва
35. Хешування паролів
36. Транспортне кодування
37. Алгоритм RSA
38. Технології цифрових підписів
39. Контрольні суми
40. Механізм розповсюдження відкритих ключів
41. Алгоритм Діффі-Хеллмана
42. Мережеві компоненти, що атакуються
43. Рівні мережевих атак згідно моделі OSI
44. Антивірусний захист
45. Методи резервного копіювання
46. Забезпечення мережевої безпеки
47. Помилки у ПЗ, що приводять до атак на інформацію
48. Основні положення по розробці ПЗ
49. Класифікація інформаційних об'єктів
50. Вимоги по роботі з інформацією
51. Політика ролей
52. Створення комплексної системи безпеки
53. Методи забезпечення безвідмовності