Chương 3

MÃ HÓA CHỐNG NHIỄU

Trong chương này chúng ta sẽ nghiên cứu vấn đề mã hóa và giải mã kênh truyền, tức là mã hóa các tin nguồn thành các từ mã trước khi truyền đi để chống nhiễu do kênh truyền, và sau đó giả mã các xâu nhận được để xác định chính xác từ mã nào đã được gửi đi.

3.1 CÁC KHÁI NIỆM CƠ BẢN

3.1.1. Các khái niệm về kênh truyền

Trong các phần trước, chúng ta đã giới thiệu các loại mã dùng trong mã hóa, với mục tiêu phát hiện lỗi và sửa lỗi, tức là chúng ta chỉ quan tâm đến số các lỗi chúng ta có thể phát hiện và sửa chữa mà không quan tâm đến việc làm thế nào, hoặc nơi mà các lỗi đó xảy ra. Tuy nhiên, mỗi kênh truyền có thể có khả năng xảy ra lỗi với các xác suất khác nhau. Shannon đã đề xuất mô hình truyền tin có xem xét đến xác suất xảy ra lỗi trên kênh truyền gọi là mô hình xác suất. (*Probabilistic Model*). Trong các phần sau của giáo trình này, chúng ta sử dụng mô hình xác suất trong việc nghiên cứu các mã chống nhiễu.

Về mặt vật lý, một kênh truyền là một môi trường truyền tin, có thể gửi và nhận tin, có thể có nhiễu gây lỗi cho việc truyền và nhận tin. Về mặt toán học, ta có một số định nghĩa hình thức cho kênh truyền theo mô hình xác suất như sau:

Định nghĩa 3.1

Một kênh truyền bao gồm một bảng ký hiệu $A = \{a_1, a_2, \dots, a_q\}$ và một tập các xác suất kênh truyền (xác suất P(nhận được a_j | khi đã gửi a_i), có dạng :

$$\{ P(a_j \text{ received} \mid a_i \text{ was sent}) \} \tag{3.1}$$

Các xác suất trên phải thỏa mãn điều kiện:

$$\sum_{j=1}^{q} P(a_j \text{ received} \mid a_i \text{ was sent}) = 1, \quad \forall a_i \in A. \tag{3.2}$$

Thí dụ 3.1 Một kênh truyền với bảng ký hiệu nhị phân A = {0, 1}sẽ bao gồm tập 4 xác suất sau:

p_1 = P(nhận được ‘0’ | khi đã gửi ‘0’)
p_2 = P(nhận được ‘1’ | khi đã gửi ‘0’)
p_3 = P(nhận được ‘0’ | khi đã gửi ‘1’)
p_4 = P(nhận được ‘1’ | khi đã gửi ‘1’)

các xác suất này thỏa mãn điều kiện (3.2): $p_1 + p_2 = 1$ và $p_3 + p_4 = 1$

Định nghĩa 3.2

Một kênh truyền gọi là kênh không nhớ nếu với mọi xâu $x = x_1 x \dots x_n$ và $c = c_1 c_2 \dots c_n$ đều thỏa mãn:

$$P(x \text{ received} \mid c \text{ was sent}) = \prod_{i=1}^{n} P(x_i \text{ received} \mid c_i \text{ was sent})$$

Chú ý rằng trong kênh không nhớ thì mọi lỗi là độc lập với nhau, tức là lỗi xảy ra khi gửi/nhận một bit (chẳng hạn gửi 0 nhận được 1) là độc lập với lỗi xảy ra ở bit khác.

Đặc biệt, với bảng mã nhị phân A = {0 , 1}, nếu có một kênh truyền mà trong đó mọi xác suất lỗi là như nhau P(nhận 0 | khi gửi 1), hoặc P(nhận 1 | khi gửi 0) với cùng xác suất $p < \frac{1}{2}$, thì kênh truyền đó gọi là kênh truyền đối xứng. Các kênh đối xứng là một mô hình xác xuất được sử dụng nhiều trong lý thuyết mã hóa.

Chú ý rằng nếu xác suất lỗi là p thì *xác suất nhận đúng là 1 – p.*

Ta có định nghĩa sau:

Định nghĩa 3.3

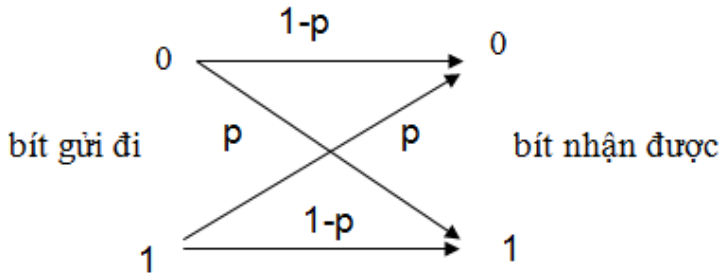
Một kênh truyền nhị phân đối xứng là một kênh trong đó tồn tại số $p < \frac{1}{2}$, *sao cho:*

$$P(\text{'1' received} \mid \text{'0' was sent}) = P(\text{'0' received} \mid \text{'1' was sent}) = p$$

$$P(\text{'1' received} \mid \text{'1' was sent}) = P(\text{'0' received} \mid \text{'0' was sent}) = 1 - p$$

- Số *p* như trên được gọi là *'xác suất chéo'* của kênh truyền nhị phân đối xứng.

Hình dưới đây mô tả một kênh nhị phân đối xứng, với xác suất chéo là p (với $p < \frac{1}{2}$). Xác suất mắc lỗi cùng là *p* khi gửi đi '0' nhận được 1, hay khi gửi đi '1' nhận được '0'. Xác suất nhận đúng sẽ là *1 – p*.



Hình 3.1. Kênh đối xứng nhị phân.

Thí dụ 3.2. Giả sử có một xâu bit được gửi đi trên kênh nhị phân đối xứng, xác suất nhận sai là 0.01. Vậy xác suất để nhận được xâu bit '10011000' là bao nhiêu khi xâu bit gửi đi là '00000000'?

Giải:

Do xác suất nhận sai là 0.01 nên xác suất nhận đúng là 1 – 0.01 = 0.99. Để nhận được xâu bit 10011000 khi xâu bit 00000000 được gửi đi thì bit đầu tiên, bit thứ 4 và bit 5 nhận được sai, còn các bit còn lại nhận được đúng. Do việc gặp lỗi ở các bit là độc lập với nhau, theo quy tắc nhân xác suất, ta tính được xác suất để xảy ra trường hợp này là:

$$(0.99)^5(0.01)^3 = 0.0000009509900499$$ ◄

3.1.2. Quy tắc giải mã người láng giềng gần nhất

Bài toán giải mã: Giả sử một từ mã *c* trong bộ mã C được gửi đi, ta nhận được xâu bit *x*. Từ xâu *x*, ta cần khôi phục được từ mã *c*. Đó là công việc chủ yếu của quá trình giải mã. Nếu việc truyền tin không có lỗi, khi đó ***x*** sẽ giống *c*. Nhưng nếu việc truyền tin có lỗi, ví dụ đường truyền có tạp âm (có nhiễu-*noise*), thì *x* sẽ không giống *c*. Vậy có thể sửa lỗi như thế nào, nói cách khác, có thể khôi phục được *c* như thế nào từ xâu nhận được *x*?

Một cách giải quyết là tính khoảng cách Hamming giữa *x* và mỗi một trong các từ mã của C. Khi đó, để giải mã *x*, ta lấy từ mã có khoảng cách Hamming nhỏ nhất đối với *x* và coi đó là từ mã *c* đã gửi đi. Phương pháp giải mã này được gọi là *Quy tắc giải mã người láng giềng gần nhất*.

Định nghĩa 3.4.

Cho C là mã với độ dài n trên bảng chữ cái A. Quy tắc giải mã 'người láng giềng gần nhất' (nearest neighbor decoding) phát biểu như sau: $\forall x \in A$, *x được giải mã thành* $c_x \in C$, *nếu* c_x *là từ mã gần nhất với x. (theo khoảng cách Hamming)*

- Tức là, nếu ký hiệu *D(x)* là giải mã của từ *x*, thì $D(x) = c_x$, với c_x thỏa điều kiện:

$$d(x, c_x) = min\ \{d(x, c), \forall\ c \in C\}$$

Thí dụ 3.3. Sử dụng phương pháp giải mã *người láng giềng gần nhất* để xác định từ mã nào được gửi đi từ mã C = {00000,01000, 10011} nếu xâu nhận được là 01100.

Giải:

Trước tiên, chúng ta tìm khoảng cách giữa 01100 với mỗi từ mã. Ta có:

d(00000,01100) = 2,

d(01000,01100) = 1,

d(10011,01100) = 5.

Vì từ mã gần nhất đối với 01100 là 01000, ta kết luận 01000 là từ mã đã được gửi đi. ◄

Chú ý:

Với các xâu nhị phân cùng độ dài, khoảng cách Hamming có thể tính bằng công thức:

$$d(x, y) = w(x + y)$$

trong đó *w(x)* là trọng số của xâu bit *x*, là số ký hiệu khác 0 trong xâu bit *x*, phép cộng được hiểu là phép cộng hai xâu bit.

Next T3 14/3 **3.1.3. Quy tắc giải mã hợp lẽ cực đại**

Trong mô hình xác suất, các quy tắc giải mã cũng dựa trên một xác suất. Quy tắc giải mã hợp lẽ cực đại cho phép giải mã xâu nhận được bởi từ mã phù hợp nhất với từ mã đã được gửi. Giả sử một từ mã *c* trong mã C được gửi đi, nhận được xâu bit *x*, do kênh truyền có nhiễu, xâu bit *x* không trùng với từ mã nào của bộ mã C. Ta cần giải mã xâu *x* thành $c_x \in C$, sao cho xác suất để nhận được *x* khi gửi c_x là lớn nhất, so với xác suất để nhận được *x* khi gửi một từ mã bất kỳ *c* trong C. Tức là việc giải mã *x* thành c_x là hợp lý nhất (hợp lẽ cực đại). Ta có định nghĩa sau:

Định nghĩa 3.5.

Cho C là mã với độ dài n trên bảng chữ cái A. Quy tắc giải mã hợp lẽ cực đại (maximum liklihood decoding) như sau: $\forall x \in A^n$, *x được giải mã thành* $c_x \in C$, *khi mà:*

$$P(x: received \,|\, c_x \ was\ sent) = max_{c \in C} \{ P(x \ received \,|\, c \ was\ sent) \}$$

- Tức là, nếu ký hiệu *D(x)* là giải mã của từ *x*, thì $D(x) = c_x$, với c_x là từ mã có khả năng nhất sẽ biến thành x, so với các từ mã khác khi được gửi đi.

Bây giờ chúng ta sẽ chỉ ra rằng *Phương pháp giải mã người láng giềng gần nhất* sẽ đem lại một từ mã giống nhiều nhất với từ mã được gửi đi và như vậy đó cũng là phép giải mã hợp lẽ cực đại.

Định lý 3.1

Trong một kênh truyền nhị phân đối xứng với xác suất chéo $p < \frac{1}{2}$, *phương pháp giải mã người làng giềng gần nhất là tương đương với phương pháp giải mã hợp lẽ cực đại.*

Chúng minh: Cho C là một bộ mã nhị phân, giả sử *c* là 1 từ mã bất kỳ trong C được gửi đi và ta nhận được xâu *x* sai khác *i* lỗi so với *c*, tức là d(*c, x*) = *i*. Do *x* có *i* bit nhận sai và *n- i* bit nhận đúng, và xác suất nhận sai là *p* (xác suất nhận đúng là *1 – p*) nên ta có:

$$P(x \ received \mid c \ was\ sent) = p^i (1-p)^{n-i}$$

Vì $p < \frac{1}{2}$, nên ta có $\frac{1-p}{p} > 1$, do đó:

$$p^i (1-p)^{n-i} = p^{i+1} (1-p)^{n-i-1} . \frac{1-p}{p} > p^{i+1} (1-p)^{n-i-1}$$

vậy:
$$p^i (1-p)^{n-i} > p^{i+1} (1-p)^{n-(i+1)}$$

Với *i = 0, 1, 2, …n-1* ta có:

$$p^0 (1-p)^n > p(1-p)^{n-1} > \ldots > p^n (1-p)^0$$

Từ công thức trên ta thấy khi *i* càng nhỏ thì xác suất (1) càng lớn. Phuong pháp giải mã người láng giềng gần nhất chọn một từ mã c_x gần với x nhất tức là $d(c_x , x) = i$ là nhỏ nhất nên xác suất (1) là lớn nhất. Tức là, quy tắc giải mã '*người láng giềng gần nhất'* cho phép giải mã *x* thành từ mã c_x với một xác suất lớn nhất , phù hợp với quy tắc giải mã hợp lẽ cực đại.

3.2 PHÁT HIỆN LỖI VÀ SỬA LỖI

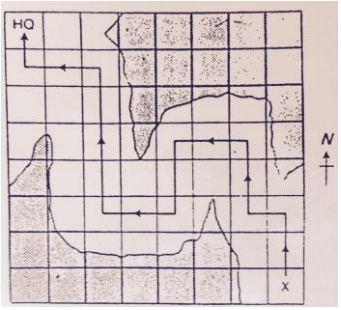
Phát hiện lỗi và sửa lỗi là công việc chủ yếu của quá trình giải mã. Trước hết chúng ta trình bày nguyên lý chung để phát hiện lỗi và sửa lỗi trong truyền tin.

3.2.1. Nguyên lý phát hiện lỗi và sửa lỗi

Giả sử một đầu kênh truyền gửi đi một từ mã $c \in C$, và bên kia nhận được một xâu *x*. Nguyên lý phát hiện lỗi và sửa lỗi được hiểu như sau:

- *Nguyên lý phát hiện lỗi:* Kiểm tra xem xâu nhận được có phải là một từ mã trong C hay không, nếu không phải từ mã thì xâu nhận được là sai, có lỗi.
- *Nguyên lý sửa lỗi:* Khi phát hiện xâu *x* nhận được là sai, tìm một từ mã c_x có khoảng cách Hamming gần nhất với *x* (theo quy tắc giải mã người láng giềng gần nhất) để thay cho xâu *x* bị lỗi, tức là đã sửa được lỗi, và coi c_x là từ mã *c* đã được gửi đi. (c_x là giải mã của xâu *x*)

Như vậy nguyên lý phát hiện lỗi và sửa lỗi đều dựa trên khoảng cách Hamming. Thí dụ dưới đây cho thấy khoảng cách Hamming cực tiểu d(C) có liên quan đến việc phát hiện lỗi và sửa lỗi.



Thí dụ 3.4 Giả sử HQ và X có tấm bản đồ mạng lưới như hình vẽ bên. Vì lý do bí mật, chỉ có HQ biết các chỉ dẫn mỗi bước đi cho X để X có thể đến được HQ. HQ cần gửi cho X các thông điệp N, S, E, W để theo đó X có thể đi được đến HQ (dãy thông điệp cần chuyển là: NNWNNWWSSWWNNNNWWN). Rõ ràng là độ tin cậy (chính xác) của việc truyền/nhận thông tin là quan trọng hơn tốc độ. Chúng ta có thể mã hóa các thông điệp bằng mã nhị phân:

$$C_1 = \begin{cases} 00 = N \\ 01 = W \\ 10 = E \\ 11 = S \end{cases}$$

Ta có khoảng cách Hamming cực tiểu của bộ mã $d(C_1) = 1$. Nếu một từ mã gửi đi mà bị mắc 1 hoặc 2 lỗi thì nó lại biến thành từ mã khác, do đó mã này không thể phát hiện cũng như không thể sửa được 1 lỗi nào.

Bây giờ ta thêm một bit kiểm tra chẵn lẻ vào các từ của mã C_1 (theo nguyên tắc: bit thêm vào là 0 hay 1 sao cho tổng các bit '1' trong mọi từ mã luôn chẵn. Ta nhận được mã C_2 như sau:

$$C_2 = \begin{cases} 000 \\ 011 \\ 101 \\ 110 \end{cases}$$

Dễ thấy $d(C_2) = 2$, mã này có thể phát hiện 1 lỗi, nhưng không thể sửa được lỗi nào.

Thật vậy, giả sử từ mã c = 000 được gửi đi, nếu chỉ có 1 lỗi xảy ra, sẽ nhận được xâu bit có đúng một bit '1', là xâu 001 ; 010 hoặc 100, đều không phải là từ mã hợp lệ trong bộ mã C_2, do đó sẽ phát hiện được xâu x là lỗi. Tương tự như vậy với các từ mã khác được gửi, nếu xâu nhận được có 1 lỗi sẽ làm thay đổi số bit '1' từ chẵn thành lẻ, do đó lỗi bị phát hiện. Tuy nhiên có thể thấy nếu xâu nhận được mắc 2 lỗi thì không thể phát hiện được, bởi vì khi đó tổng số bit '1' vẫn là một số chẵn. Mặt khác, khi 1 từ mã gửi đi mà bị mắc 2 lỗi thì xâu nhận được lại là 1 từ mã khác trong bộ mã C_2, do đó không thể phát hiện là đã mắc lỗi.

Bây giờ ta tạo mã mới từ C_2 bằng cách lặp lại 2 bit đầu của các từ mã vào cuối mỗi từ mã, ta nhận được mã C_3 như sau:

$$C_3 = \begin{cases} 00000 \\ 01101 \\ 10110 \\ 11011 \end{cases}$$

Dễ thấy $d(C_3) = 3$, mã này không những phát hiện được các xâu chứa 2 lỗi mà còn sửa được các xâu chứa 1 lỗi. Chẳng hạn, khi HQ muốn gửi cho X thông điệp 'N', thông điệp này đã được mã bằng mã C_3, tức là HQ gửi đi từ mã c = 00000 nhưng do bit thứ hai bị lỗi, X nhận được xâu 01000. X sẽ biết ngay từ mã đã bị lỗi, và phần lỗi là ở bên trái bit kiểm tra chẵn lẻ (do tổng các bit 1 trong ba bit đầu phải là số chẵn), vì vậy có thể sửa được lỗi này bằng cách lấy 2 bit cuối

thay cho 2 bít đầu. Từ đó *x* có thể được giải mã D(01000) = 00000, từ đó tìm được hướng đi 'N'.

Ta có định nghĩa về sự phát hiện lỗi và sửa lỗi dựa trên khoảng cách Hamming như sau.

Định nghĩa 3.6.

Cho C là mã với độ dài n trên bảng chữ cái A, khi đó:

1. *Mã C gọi là phát hiện được s lỗi, nếu* $\forall c \in C$ *và mọi xâu* $x \neq c$ *ta có: nếu* $d(x, c) \leq s$ *thì* $x \notin C$.
2. *Mã C gọi là sửa được t lỗi, nếu* $\forall c \in C$ *và mọi xâu* $x \neq c$ *ta có: nếu* $d(x, c) \leq t$ *thì x là từ mã gần nhất với c.*

Ta có thể giải thích rõ hơn cho định nghĩa trên như sau:

1. Mã C gọi là phát hiện được *s* lỗi nếu với mọi từ mã *c gửi đi,* mà có *s* bit bị thay đổi ở xâu nhận được x, thì xâu này không phải là một từ mã trong C, vì vậy phát hiện được từ mã gửi đi đã bị lỗi.

2. Mã C gọi là sửa được *t* lỗi nếu khi gửi đi từ mã c, mà có *t* bit bị thay đổi ở xâu nhận được x, thì sẽ thay thế được xâu này với một từ mã gần nó nhất có d(x, c) ≤ *t* (theo quy tắc giải mã người láng giềng gần nhất)

Ta có định lý sau về khả năng phát hiện lỗi và sửa lỗi của một mã, việc chứng minh định lý dành cho sinh viên như một bài tập.

Định lý 3.2

Cho C là mã với độ dài n trên bảng chữ cái A, khi đó:

1. *Mã C phát hiện được k lỗi trong mọi từ mã nếu khoảng cách Hamming* $d(C) \geq k + 1$
2. *Mã C sửa được k lỗi trong mọi từ mã nếu khoảng cách Hamming* $d(C) \geq 2k + 1$

Như vậy, khả năng phát hiện và sửa lỗi của một mã phụ thuộc vào khoảng cách Hamming cực tiểu d(C) của bộ mã đó, và d(C) càng lớn thì khả năng phát hiện lỗi và sửa lỗi của mã càng cao.

Hệ quả 3.1

Nếu mã C có khoảng cách Hamming cực tiểu là d(C) = d, thì C có thể phát hiện được tối đa k = d – 1 lỗi, và có thể sửa được tối đa $k = \lfloor \frac{d-1}{2} \rfloor$ *lỗi.*

($\lfloor x \rfloor$ là số nguyên lớn nhất không vượt quá x, chẳng hạn $\lfloor 3.14 \rfloor = 3$; $\lfloor 5.0 \rfloor = 5$; $\lfloor -3.14 \rfloor = -4$).

Sự chứng minh của hệ quả này suy trực tiếp từ Định lý 3.2. Hệ quả này thuận lợi cho việc tính toán khả năng phát hiện lỗi và sửa lỗi của một mã. Chẳng hạn, với 1 số khoảng cách Hamming của mã C, ta có thể phát hiện và sửa số lỗi như sau:

d(C)	Số lỗi tối đa có thể phát hiện	Số lỗi tối đa có thể sửa được
1	0	0
2	1	0
3	2	1
4	3	1
5	4	2
6	5	2
…	…	…

Thí dụ 3.5 Giả sử C là bộ mã gồm các từ mã:{c_1 = 00000000, c_2 = 11111000, c_3 = 01010111, c_4 = 10101111}.

1. C có thể phát hiện và sửa bao nhiêu lỗi trong một từ mã bị nhận sai?
2. Khi gửi đi từ mã c = 00000000, nếu nhận được một trong các xâu:
   - x_1 = 11011000,
   - x_2 = 11111000,
   - x_3 = 10101000,

   Trường hợp nào thì xâu nhận được bị coi là bị lỗi, tại sao?
3. Nếu gửi đi từ mã c_4 = 10101111, nhận được xâu x = 10100011. Có thể phát hiện lỗi và sửa lỗi không?

Giải :

1. Tính khoảng cách Hamming cực tiểu giữa các từ mã, ta có d(C) = 5. Vậy mã có thể phát hiện tối đa 5 – 1 = 4 lỗi trong một từ mã. Số lỗi tối đa có thể sửa được là $\lfloor \frac{5-1}{2} \rfloor = 2$

Xâu x_1 nhận được từ C, với 4 lỗi (ở bit 1, bit 2, bit 4 và bit 5), mã C phát hiện được x_1 có lỗi và yêu cầu gửi lại. Với xâu x_2, mắc 5 lỗi, do đó không phát hiện được lỗi, người nhận cho rằng đã gửi đi từ mã c_2. Với xâu nhận được x_3, chỉ mắc 3 lỗi nên phát hiện được đây là từ mã bị lỗi, và yêu cầu gửi lại.

2. Do số lỗi là 2, nên có thể sửa được, dùng phương pháp giải mã người láng giềng gần nhất, có thể giải mã được D(x) = c_4, chính là từ mã gửi đi.

Ta sẽ trình bày một số mã phát hiện lỗi đơn giản. Để dễ theo dõi, chúng ta chỉ xem xét các tin nguồn và các mã đều là các xâu nhị phân.

3.2.2 Các mã phát hiện lỗi

Mã chẵn lẻ (Parity code)

Giả sử thông điệp nguồn là các xâu gồm k bit. Mã chẵn lẻ thêm vào thông điệp nguồn một bit thứ k + 1, gọi là bit kiểm tra chẵn lẻ (*parity check bit*) theo cách sau:

- Nếu thông điệp là xâu có chứa một số chẵn các bit '1' ta thêm '0' vào cuối xâu.
- Nếu thông điệp là xâu có chứa một số lẻ các bit '1' ta thêm '1' vào cuối xâu.

Tổng quát, ta mã hóa thông điệp x = $x_1 x_2 .... x_k$ thành $x_1 x_2 .... x_k x_{k+1}$, *trong đó:*

$$x_{k+1} = (x_1 + x_2 + ... + x_k) \bmod 2$$

Mã mới nhận được là một mã (*k+1, M, k*), trong đó *k* là độ dài tin nguồn, M là số tin nguồn, *k+1* là độ dài từ mã của mã chẵn lẻ.

Nhận xét:

Việc thêm bit kiểm tra chẵn lẻ để đảm bảo số các bit 1 trong mỗi từ mã (xâu mở rộng) phải là một số chẵn (khi đó ta nói từ mã này là hợp lệ).

Như vậy, khi bit kiểm tra chẵn lẻ được thêm vào một xâu các bit, nếu có một lỗi duy nhất trong việc truyền tin đi từ một từ mã thì tổng số các bit 1 trong từ mã đó sẽ là một số lẻ. Do đó, có thể phát hiện được lỗi này. Tuy nhiên, nếu có hai lỗi xảy ra thì những lỗi này không thể phát

hiện được vì rằng tổng số các bit 1 trong xâu mở rộng sẽ vẫn còn là một số chẵn. Nói chung, một số lẻ bất kỳ các lỗi có thể được phát hiện, trong khi một số chẵn các lỗi không thể được phát hiện. Máy thu kiểm tra bit chẵn lẻ của thông điệp thu được và nếu nó phát hiện ra một sự bất đồng, nó sẽ yêu cầu máy phát truyền tin lại thông điệp.

Thí dụ 3.6: Giả sử rằng, bit kiểm tra chẵn lẻ được thêm vào một thông điệp trước khi nó được truyền đi. Có thể kết luận gì nếu ta nhận được các xâu bit 1100001 và 1011111 từ những thông điệp được gửi đi.

Giải: Vì xâu nhận được là 1100001 chứa một số lẻ bit 1, nó không thể là từ mã hợp lệ (và do đó phải chứa một số lẻ các lỗi) .

Đối với xâu nhận được là 1011111: xâu này chứa một số chẵn bit 1, đó là từ mã hợp lệ. Do đó hoặc việc truyền tin hoặc không mắc lỗi, hoặc có một số chẵn các lỗi. Nói chung mã chẵn lẻ chỉ có thể khẳng định chắc chắn việc truyền tin bị mắc lỗi khi số bit '1' nhận được là lẻ.

Ma trận sinh của mã chẵn lẻ

Với mã chẵn lẻ, có một ma trận G để sinh ra các từ mã này từ các thông điệp nguồn. Chẳng hạn, với các thông điệp 4 bit $x = x_1 x_2 x_3 x_4$ để mã hóa thành một mã chẵn lẻ (mã phát hiện lỗi), ta thêm bít thứ 5 cho mỗi thông điệp, theo công thức:

$$x_5 = (x_1 + x_2 + x_3 + x_4) \bmod 2$$

Như vậy, ta có công thức mã hóa: $E(x) = x.G$, với G là ma trận cấp 4×5, có dạng:

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

G là ma trận sinh của các mã chẵn lẻ có độ dài 5, với bit cuối cùng là bit kiểm tra chẵn lẻ.
Có thể viết G dưới dạng chuẩn:

$$G = (I_4 \mid A) ,$$

với A là ma trận cột $\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$. Để cho gọn, có thể viết $A = (1\ 1\ 1\ 1)^T$ và hiểu là cột toàn số 1.

Chẳng hạn, với các thông điệp X = {0001 ; 0011, 0111, 1111}, mã hóa nhờ ma trận G trên:

E(0001) = (0001).G = (00011)
E(0011) = (0011).G = (00110)
E(0111) = (0111).G = (01111)
E(1111) = (1111).G = (11110)

Ta nhận được mã chẵn lẻ sau: ⇒

$$C = \begin{Bmatrix} 00011 \\ 00110 \\ 01111 \\ 11110 \end{Bmatrix}$$

Để mã hóa các thông điệp độ dài *k* bit thành mã chẵn lẻ, ta có ma trận sinh G là ma trận:

$$G = (I_k \mid A) ,$$

trong đó A là ma trận cột gồm k phần tử toàn số 1.

Mã sánh hợp

Một cách đơn giản khác để phát hiện lỗi là lặp lại mỗi bit trong thông điệp 2 lần, khi đó chúng ta có thể phát hiện các lỗi trong một từ mã theo cách sau:

Vì các từ mã của các xâu bit này chứa các cặp bit sánh hợp (giống nhau), như vậy, ta có thể phát hiện các lỗi làm thay đổi không quá một bit ở mỗi cặp bit sánh hợp này**.**

Thí dụ 3.7

1. Mã hóa xâu bit 010001 bằng cách lặp lại mỗi bit 2 lần.

Giải: Việc lặp lại mỗi bit 2 lần đem lại từ mã 001100000011.

2. Ta có thể phát hiện lỗi ở bit thứ 2, thứ 6 và thứ 7 của các từ mã có 8 bit như khi nhận được 01001001 do từ mã 00001111 được gửi đi. Mặt khác, ta không thể phát hiện lỗi khi bit 5 và bit 6 cùng thay đổi (chẳng hạn không thể phát hiện các lỗi khi nhận được 00000011, do từ mã 00001111 được gửi đi).

Ma trận sinh của mã sánh hợp

Với mã sánh hợp, có thể xây dựng ma trận G để sinh ra các từ mã này từ các thông điệp nguồn. Chẳng hạn, với các thông điệp 4 bit $x = x_1 x_2 x_3 x_4$ để mã hóa thành một mã sánh hợp, ta lặp lại mỗi bit thêm một lần nữa, do đó ma trận sinh G của mã sánh hợp có dạng:

$$G = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Chẳng hạn, để mã hóa xâu x = (1001), ta có:

$$E(x) = (1001).G = (11000011)$$

Kết luận: Ở trên, chúng ta đã xem xét các mã có thể được dùng để phát hiện sai. Khi các lỗi được phát hiện, tất cả những gì chúng ta có thể làm để có được từ mã chính xác là đề nghị truyền phát lại và hy vọng sẽ không có lỗi nào xuất hiện trong lần thực hiện này. Tuy nhiên, không phải khi nào cũng có thể yêu cầu phát lại, đó là trường hợp các kênh 1 chiều (*one way channel*): 1 đầu chỉ phát và 1 đầu chỉ nhận. Khi đó, ta phải sử dụng những mã mạnh hơn, không chỉ phát hiện sai mà còn sửa được sai. Mục sau, chúng ta sẽ xem xét tới các mã này, gọi là mã sửa lỗi.

3.2.3 Các mã sửa lỗi

Chúng ta đã thấy rằng, khi sự dư thừa được có trong các từ mã, như khi bit kiểm tra chẵn lẻ được thêm vào một xâu các bit, chúng ta có thể phát hiện được những lỗi truyền tin. Chúng ta thậm chí có thể làm tốt hơn nếu đưa vào nhiều dư thừa hơn. Chúng ta không chỉ có thể phát hiện sai mà còn có thể sửa sai. Chính xác hơn, nếu số lỗi xảy ra trong một từ mã được truyền là đủ nhỏ, chúng ta có thể xác định được chính xác từ mã nào đã được truyền đi.

Mã lặp lại (repetition codes)

Chúng ta mã hóa một thông điệp nhị phân $x = x_1 x_2 \ldots x_i \ldots x_k$ bằng cách lặp lại xâu bít cần truyền m lần, tức là:

$$E(x) = x_{11} x_{21} \ldots x_{i1} \ldots x_{k1}\, x_{12} x_{22} \ldots x_{i2} \ldots x_{k2} \ldots\ldots x_{1m} x_{2m} \ldots x_{im} \ldots x_{km}$$

Trong đó, x_{ij} là bit thứ i của thông điệp được lặp lại lần thứ j trong từ mã ($i = 1, 2, \ldots, k$; $j = 1, 2, \ldots, m$). Dùng quy tắc lấy đa số đơn giản, bằng cách so sánh các bit có cùng chỉ số i với nhau ($1 \le i \le k$), nếu số lượng các bit x_{ij} bằng 0 lớn hơn số lượng các x_{ij} bằng 1 thì ta có thể kết luận x_i bằng 0 khi giải mã; trái lại kết luận $x_i = 1$. Trong trường hợp hai số lượng này bằng nhau thì ta chưa thể kết luận về x_i, tuy nhiên có thể lấy số lần lặp lại m là một số lẻ để tránh tình huống này.

Mã được tạo ra theo phương pháp tạo mã như trên gọi là mã lặp lại (Repetition codes).

Mã lặp lại có nhược điểm là độ dài của từ mã trong mã lặp lại sẽ rất lớn nếu k và m lớn.

Thí dụ 3.8. Giả sử ta có các thông điệp {000, 001, 100, 101} (có thể là một mã nào đó, cần được mã hóa thành mã lặp lại)

Ta mã hóa các thông điệp $x = x_1 x_2 x_3$ bằng phương pháp lặp 3 lần: $E(x) = x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9$, ở đây $x_1 = x_4 = x_7$, $x_2 = x_5 = x_8$, $x_3 = x_6 = x_9$ được mã C với 4 từ mã hợp lệ sau:

C = {000000000, 001001001, 100100100, 101101101}

Để giải mã một xâu bit nhận được, mà xâu này có thể chứa các lỗi, ta dùng quy tắc lấy đa số đơn giản.

Chẳng hạn, để xác định x_1, ta xem xét x_1, x_4, x_7. Nếu 2 trong các bit này là 1, ta kết luận $x_1 = 1$. Trái lại, 2 trong 3 bit này là 0, ta kết luận $x_1 = 0$. Nói chung chúng ta xem xét 3 bit tương ứng với mỗi bit ở thông điệp gốc.

Ví dụ, với mã lặp lại 3 lần được dùng nếu ta nhận được xâu x = 101001001. Sử dụng quy tắc lấy đa số đơn giản trên, để xác định x_1, ta xem xét x_1, x_4, x_7 trong xâu x, do có 2 bit '0' và 1 bit '1' nên ta quyết định $x_1 = 0$. Với các bit thứ hai và thứ ba, ta giữ nguyên do các vị trí đều không bị lỗi. D(x) = 001001001 là từ mã thứ hai trong C, từ đó suy ra thông điệp gốc cần gửi đi là 001.

Nhận xét

1. Việc mã hóa bằng các mã lặp lại có nhược điểm là độ dài từ mã sẽ rất lớn nếu k và m lớn, và như vậy sẽ làm giảm tốc độ tuyền tin. Tuy nhiên, việc này lại làm tăng độ tin cậy, do các mã lặp lại có khả năng phát hiện lỗi và sửa lỗi rất tốt. Trong nhiều trường hợp sự tin cậy cần hơn tốc độ.

2. Việc sử dụng các mã chống nhiễu (mã phát hiện lỗi và mã sửa lỗi) làm tăng khoảng cách Hamming cực tiểu của các mã, từ đó làm tăng khả năng phát hiện lỗi và sửa lỗi.

Ma trận sinh của các mã lặp lại:

Chẳng hạn, với các thông điệp 4 bit $x = x_1 x_2 x_3 x_4$ để mã hóa thành một mã lặp lại 3 lần, ta lặp lại xâu x 3 lần, nhận được từ mã:

$$E(x) = x_1 x_2 x_3 x_4 \; x_5 x_6 x_7 x_8 \; x_9 x_{10} x_{11} x_{12}$$

Trong đó: $x_1 = x_5 = x_9$; $x_2 = x_6 = x_{10}$; $x_3 = x_7 = x_{11}$; $x_4 = x_8 = x_{12}$.

Do mỗi xâu x (coi là 1 ma trận hàng) nhân với ma trận đơn vị thì bằng chính xâu đó, do đó để lặp lại xâu x ba lần, ta nhân nó với ma trận G như sau

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Có thể ký hiệu : $G = (I_4 \mid I_4 \mid I_4)$

Để mã hóa thông điệp 4 bit x thành 1 từ mã lặp lại 3 lần, ta có E(x) = x.G.

Chẳng hạn, để mã hóa thông điệp x = 1101 thành từ mã lặp lại 3 lần, ta có:

$$E(x) = (1101).G = 110111011101$$

3.3 PHÁT HIỆN LỖI VÀ SỬA LỖI CHO CÁC MÃ TUYẾN TÍNH

3.3.1 Giải mã cho các mã tuyến tính bằng tập giải mã coset

Vấn đề giải mã và vector lỗi

Giả sử từ mã $\boldsymbol{c} = c_1 c_2 \ldots c_n$ được gửi đi trên một kênh có nhiễu, xâu nhận được là $\boldsymbol{x} = x_1 x_2 \ldots x_n$.

- Xâu $\boldsymbol{e} = e_1 e_2 \ldots e_n$ được gọi là vector lỗi (*error vector*) nếu $\boldsymbol{e} = x - c$ (*)

Nếu xác định được vector lỗi *e*, thì từ xâu *x* ta sẽ xác định được từ mã nào đã được gửi đi, vì từ (*) ta có: $c = x - e$.

- Bộ giải mã cần phải quyết định rằng với xâu nhận được $\boldsymbol{x}$ thì từ mã nào đã được gửi đi, hoặc là xác định được vector lỗi *e* đã xuất hiện trong quá trình truyền tin, từ mã là giải mã của *x* ký hiệu là c_x

Có nhiều cách để xác định c_x, chẳng hạn phương pháp giải mã người láng giềng gần nhất. Một phương pháp hiệu quả hơn cho các mã tuyến tính là phương pháp dùng các tập giải mã (coset).

Tập giải mã coset

Định nghĩa 3.7

Cho mã tuyến tính C[n, k] trên trường GF(q) và một vector tùy ý $\boldsymbol{a} \in F_q^n$ *, khi đó tập giải mã của C (coset of C) được ký hiệu và xác định như sau:*

$$\boldsymbol{a} + \boldsymbol{C} = \{ a + c \mid \forall c \in C \}$$

Từ định nghĩa trên, ta thấy mã C có thể có nhiều coset, mỗi vector $a \in F_q^n$ đều xác định một tập coset của C, tuy nhiên một số tập có thể trùng nhau. Xét bổ đề sau:

Bổ đề 3.1

Giả sử $\boldsymbol{a} + \boldsymbol{C}$ *là một coset của C, và giả sử* $b \in \boldsymbol{a} + \boldsymbol{C}$*, khi đó ta có:*

$$\boldsymbol{b} + \boldsymbol{C} = \boldsymbol{a} + \boldsymbol{C}$$

Chứng minh: Vì $b \in \boldsymbol{a} + \boldsymbol{C}$, nên b = a + x, với x là từ mã nào đó của bộ mã C. Bây giờ giả sử $b + y \in \boldsymbol{a} + \boldsymbol{C}$, khi đó: $b + y = (a + x) + y = a + (x + y) \in \boldsymbol{a} + \boldsymbol{C}$. Vì vậy $\boldsymbol{b} + \boldsymbol{C} \subseteq \boldsymbol{a} + \boldsymbol{C}$. (i)

Mặt khác, nếu $a + z \in \boldsymbol{a} + \boldsymbol{C}$, thì $a + z = (b - x) + z = b + (z - x) \in \boldsymbol{b} + \boldsymbol{C}$. Vậy $\boldsymbol{a} + \boldsymbol{C} \subseteq \boldsymbol{b} + \boldsymbol{C}$ (ii)
Từ (i) và (ii) suy ra $\boldsymbol{b} + \boldsymbol{C} = \boldsymbol{a} + \boldsymbol{C}$

Định lý 3.3

Cho mã tuyến tính C[n, k] trên trường GF(q), khi đó:

(i) Mọi vector tùy ý $a \in F_q^n$ *thuộc về một coset nào đó của C,*

(ii) Mọi coset của C đều chứa đúng q^k *vector,*

(iii) Hai coset bất kỳ của C hoặc trùng nhau hoặc rời nhau.

Việc chứng minh định lý này được trình bày trong tài liệu tham khảo [1].

Thí dụ 3.9

Cho C là mã nhị phân tuyến tính [4, 2] có ma trận sinh:

$$G = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

Dễ thấy C = {0000, 1011, 0101, 1110}, ở đây q = 2, k = 2, n = 4, theo định lý trên, mọi coset của C đều có đúng $q^k = 2^2 = 4$ vector. Mặt khác, F_2^4 có tất cả $2^4 = 16$ vector nhị phân độ dài 4, các vector này phải xắp xếp trong các coset khác nhau của C. Vì vậy, ta chọn 4 vector có trọng số nhỏ nhất trong F_2^4 để tạo nên 4 coset của C

Coset 1: 0000 + C = {0000, 1011, 0101, 1110} (bằng chính C, do 0000 ∈ C)

Coset 2: 1000 + C = {1000, 0011, 1101, 0110}

Coset 3: 0100 + C = {0100, 1111, 0001, 1010}

Coset 4: 0010 + C = {0010, 1001, 0111, 1100}

Chú ý rằng nếu chọn a = 0001 thì coset 0001 + C = {0100, 1111, 0001, 1010} = coset 0100 + C, bởi vì 0001 ∈ 0100 + C (coset 3). Do đó để tìm tất cả các coset của mã C[4, 2] ta chỉ cần chọn 4 vector có trọng số nhỏ nhất với vị trí bit '1' tăng theo thứ tự từ trái qua phải. *Vector có trọng số nhỏ nhất trong mỗi coset của mã tuyến C[n, k] gọi là coset leader của coset đó.*

Nếu một coset có nhiều vector có cùng trọng số nhỏ nhất, ta có thể chọn ngẫu nhiên một vector như vạy làm coset leader. Chẳng hạn trong thí dụ trên, các coset leader của các coset lần lượt là : 0000, 1000, 0100. 0010. Với coset 0100 + C (coset 3), ta cũng có thể chọn vector 0001 làm coset leader thay cho 0100.

Bảng chuẩn của mã tuyến tính

Định nghĩa 3.8

Bảng coset chuẩn của mã tuyến tính C[n, k] là một bảng gồm q^{n-k} *hàng và* q^k *cột chứa tất cả các vector của* F_q^n*, trong đó hàng đầu tiên chứa tất cả các từ mã của C với từ mã O (từ mã gồm các ký hiệu '0')ở bên trái nhất, các hàng khác chứa các coset* $\mathbf{a_i} + \mathbf{C}$*, với các coset leader* a_i *ở bên trái nhất*

Một bảng coset chuẩn của mã tuyến tính C có thể được xây dựng theo các bước sau:

Bước 1. Lập hàng 1: Liệt kê các từ mã của **C**, với từ mã O ở bên trái nhất.

Bước 2. Chọn bất kỳ vector $a_1 \in F_q^n$ có trọng số nhỏ nhất và không thuộc hàng 1 của bảng, tính coset $\mathbf{a_1} + \mathbf{C}$ và viết vào hàng 2 của bảng, a_1 viết dưới từ mã O và $a_1 + x$ dưới từ mã x, $\forall x \in C$.

Bước 3. Chọn vector a_2 không thuộc hàng 1 và hàng 2, a_2 có trọng số nhỏ nhất trong các vector còn lại của F_q^n, tính $\boldsymbol{a_2}$ + **C** và liệt kê vào hàng 3 tương tự như bước 2.

Bước i. Tiếp tục chọn các vector a_i không thuộc các hàng đã tính ở trên, để tính coset thứ i…cho đến khi tính đủ q^{n-k} coset của C và liệt kê vào q^{n-k} hàng của bảng theo cách trên, các dòng này phải khác nhau và sẽ chứa tất cả các vector của F_q^n.

Thí dụ 3.10

Cho mã nhị phân tuyến tính C = {0000, 1011, 0101, 1110} trong thí dụ 3.9, bảng chuẩn của C là:

Codewords of C →				
	0000	1011	0101	1110
	1000	0011	1101	0110
	0100	1111	0001	1010
	0010	1001	0111	1100
	↑			
	coset leader			

Qua thí dụ trên ta thấy, mỗi phần tử trong bảng là tổng của một từ mã ở dòng đầu tiên của cột chứa phần tử đó cộng với coset leader tương ứng ở bên trái nhất của hàng chứa phần tử đó. Chẳng hạn với phần tử 1111, rõ ràng ta có 1111 = 1011 + 0100.

Giải mã bằng bảng coset chuẩn

Bây giờ chúng ta sẽ mô tả việc giải mã dùng bảng coset chuẩn được thực hiện như thế nào.

Giả sử gửi đi một từ mã $c \in$ C và nhận được xâu $x \in F_q^n$, khi đó vị trí của x sẽ được xác định trong bảng, vì bảng chứa tất cả các vector của F_q^n. Theo cách xây dựng bảng, ta có $x = c + e$, với c là từ mã ở dòng đầu tiên của cột chứa x, còn e là coset leader của hàng chứa x. Như vậy, mỗi khi nhận được một xâu x, từ vị trí của xâu nhận được ta xác định được từ mã gửi đi c ờ dòng đầu của cột chứa x, còn vector lỗi chính là coset leader của hàng chứa x.

Chẳng hạn, gửi đi một từ của mã C trong thí dụ trên, nhận được xâu x = 1111, ta sẽ xác định được từ mã gửi đi là c = 1011, còn vector lỗi là e = 0100.

Như vậy: giải mã của một xâu nhận được $x \in F_q^n$ là từ mã nằm ở dòng đầu tiên của cột chứa x trong bảng coset chuẩn. Từ mã là giải mã của x ký hiệu là c_x.

Thí dụ 3.11 *(về một bảng coset chuẩn khi tính gặp 2 hàng giống nhau)*

Cho mã nhị tuyến tính C có ma trận sinh là : G = $\begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}$, xác định bảng chuẩn của mã C.

Giải: Từ ma trận sinh, ta có C = {0000, 1101, 1010, 1111}. Ta lập bảng coset chuẩn theo các bước trong định nghĩa 3.8. Lần lượt chọn các ceset leader a_0 = 0000, a_1 = 1000, a_2 = 0100, a_3 = 0010 để tính các hàng. Chú ý rằng khi tính đến hàng thứ tư là 0010 + C, thì ta được kết quả trùng với hàng thứ 2, bỏ đi một trong hai hàng này Chẳng hạn bỏ đi hàng thứ tư, như vậy bảng mới có 3 hàng khác nhau, ta phải tính thêm hàng mới với coset leader 0001, ta được coset {0001, 1100, 1011, 0110}. Kết quả tính toán minh họa trong hình sau:

Codewords of C →				
	0000	1101	1010	0111
	1000	0101	0010	1111
	0100	1001	1110	0011
	0010	1111	1000	0101
	0001	1100	1011	0110

↑

coset leader

Qua thí dụ trên ta thấy, trong quá trình lập bảng chuẩn, có thể có hai hàng (hai coset) giống nhau, khi đó phải tính thêm các coset mới ứng với các coset leadr mới sao cho có đủ q^{n-k} hàng khác nhau (trong thí dụ này $q^{n-k} = 2^{4-2} = 4$ hàng)

Khi đó phải bỏ đi 1 trong 2 hàng giống nhau, chỉ giữ lại 4 hàng khác nhau (do số hàng của bảng tối đa là $q^{n-k} = 2^{4-2} = 4$ hàng), kết quả được bảng coset chuẩn như sau:

Codewords of C →				
	0000	1101	1010	0111
	1000	0101	0010	1111
	0100	1001	1110	0011
	0001	1100	1011	0110

↑

coset leader

Nhận xét: Nói chung việc giải mã bằng bảng coset chuẩn rất dễ ràng nhưng khá hạn chế:

1. Việc giải mã như trên trong thực tế rất chậm với những mã lớn, và khá tốn kém cho chi phí lưu trữ các bảng chuẩn. Chẳng hạn, với mã C[6, 3], bảng chuẩn là một bảng gồm $2^{6-3} = 8$ hàng và $2^3 = 8$ cột, mã C[7, 4] có dãy chuẩn gồm 8 hàng và 16 cột.
2. Việc giải mã và sửa lỗi chỉ thực hiện được khi xâu nhận được chỉ chứa các lỗi xảy ra ở vị trí các bit '1' của các coset leader, tức là có vector lỗi là các coset leader. Ta xem xét thí dụ sau:

Thí dụ 3.11. Với mã C trong thí dụ trên, khi gửi đi từ mã 0101 (mã hóa từ thông điệp 01):

(a) Nếu xâu nhận được là x = 0001 (lỗi xảy ra ở bit thứ 2 của từ mã, vector lỗi là coset leader 0100) thì có thể sửa được lỗi này, giải mã của x là $c_x = 0101$, từ đó thông điệp gốc là 01.

(b) Nếu xâu nhận được y = 0100 (lỗi xảy ra ở bit thứ 4 của từ mã, vector lỗi e = 0001, không phải là một coset leader) thì sẽ giải mã được $c_y = 0000$, từ đó thông điệp gốc là 00: sai! Trường hợp này không sửa được lỗi.

	Message		Codeword		Channel + noise		Received vector		Decoded word		Received message
(a)	01	→	0101	→	0101	→	0001	→	0101	→	01
(b)	01	→	0101	→	0101	→	0100	→	0000	→	00

Dưới đây ta sẽ nghiên cứu một phương pháp phát hiện lỗi và sửa lỗi tốt hơn, khắc phục được các nhược điểm trên của phương pháp giải mã bằng bảng coset chuẩn. Đó là phương pháp phát hiện lỗi và sửa lỗi bằng cách dùng các ma trận kiểm tra của mã tuyến tính.

3.3.2 Phát hiện lỗi bằng ma trận kiểm tra.

Liên hệ giữa ma trận sinh và ma trận kiểm tra.

Để tiện theo dõi, chúng ta nhắc lại một số kết quả trong chương 2:

Cho C[n, k] là mã nhị phân tuyến tính, khi đó:

(a) Mã đối ngẫu của C là: $C^{\perp} = \{ v \in \{0, 1\}^n \mid v.c = 0, \forall c \in C\}$,

(b) Gọi G là ma trận sinh của C , G có cấp $k \times n$,

(c) Gọi H là ma trận sinh của mã đối ngẫu $C^{\perp}$, H có cấp $(n-k) \times n$

(d) Ma trận H gọi là ma trận kiểm tra của mã C, (tương tự, ma trận G là ma trận kiểm tra của mã $C^{\perp}$). Người ta cũng gọi H là ma trận kiểm tra liên kết với ma trận G.

Ta có một số kết quả về sự liên hệ giữa ma trận sinh G và ma trận kiểm tra H của một mã tuyến tính C như sau:

Đinh lý 3.4

Nếu C là mã tuyến tính trên trường GF(q) có ma trận sinh G, một ma trận H là ma trận kiểm tra của mã C khi và chỉ khi H có các hàng độc lập tuyến tính và thỏa điều kiện:

$$G.H^T = O \qquad (3.3)$$

trong đó: H^T là ma trận chuyển vị của H, O là ma trận không (gồm toàn số không)

Chứng minh:

Trước hết ta chú ý về cấp của các ma trận, với mã C[*n, k*] thì $G = (g_{ij})_{k \times n}$, $H = (h_{ij})_{(n-k) \times n}$ do đó $H^T = (h_{ji})_{n \times (n-k)}$. Phép nhân $G.H^T$ cho kết quả là ma trận cấp $k \times (n-k)$. Do mỗi cột của H^T là một từ mã trong mã đối ngẫu $C^{\perp}$, nên trực giao với mọi hàng của G, vì vậy kết quả mọi hàng của G nhân với mọi cột của H^T đều bằng không. Từ đó suy ra điều phải chứng minh.

Thí dụ 3.12 Cho mã tuyến tính nhị phân C = {000,001,100,101}. Kiểm tra công thức (3.3):

C là mã tuyến tính C[3, 2] nên có ma trận sinh tạo bởi hai từ mã độc lập tuyến tính:

$$G = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

Ta tìm được mã đối ngẫu $C^{\perp}$ = { 000, 010 } nên ma trận kiểm tra của mã C là H = [0 1 0],

Ta có: $G.\ H^T = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} . \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, vậy G và H thỏa mãn định lý 3.4.

***Nhận xét*:** Áp dụng định lý 3.4, ta có thể tìm ma kiểm tra H khi biết ma trận sinh G. Với ma trận sinh G đã cho, ma trận kiểm tra H cần tìm là ma trận thỏa mãn phương trình ma trận $G.H^T = O$.

Thí dụ 3.13 Tìm ma trận kiểm tra H liên kết với ma trận G sau:

$$G = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

Giải: Ở đây G có cấp $k \times n = 2 \times 3$, nên ma trận kiểm tra H sẽ có cấp $(n-k) \times n = 1 \times 3$.

Ma trận kiểm tra H sẽ có cấp 1×3. Ma trận H có cấp 1×3 , đặt $H = (x_1, x_2, x_3)$, ta sẽ xác định các phần tử x_1, x_2, x_3 từ phương trình ma trận:

$$G.\, H^T = O,$$

x_1, x_2, x_3 là nghiệm của hệ phương trình $\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

Hệ trên tương đương với hệ: $\begin{cases} 0.x_1 + 0.x_2 + 1.x_3 = 0 \\ 1..x_1 + 0.x_2 + 0.x_3 = 0 \end{cases}$

Giải ra ta được: $x_1 = 0$ và $x_3 = 0$; x_2 tùy ý, để có nghiệm khác không ta chọn $x_2 = 1$. (cần chọn $n-k$ nghiệm độc lập tuyến tính, ở thí dụ này n-k =1 nên chọn k để có 1 nghiệm khác không).

Vậy $H^T = [\, 0\ 1\ 0\ ]^T$. Từ đó ma trận kiểm tra H liên kết với ma trận G là:

$$H = [\, 0\ 1\ 0\ ]$$

Việc áp dụng định lý 3.4 cho phép tính các ma trận kiểm tra H từ ma trận G cho trước mà không cần tìm mã đối ngẫu của mã C. Tuy nhiên việc tính toán còn khá phúc tạp, nhất là khi k và n lớn.

Định lý dưới đây cho phép tính ma trận kiểm tra rất dễ dàng từ các ma trận sinh dạng chuẩn.

Định lý 3.5

Nếu mã tuyến tính C[n , k] trên trường GF(q) có ma trận sinh dạng chuẩn $G = (I_k \mid A)$ thì ma trận kiểm tra dạng chuẩn của mã C được xác định bởi:

$$H = (-A^T \mid I_{n-k}) \qquad (3.4)$$

với A^T là ma trận chuyển vị của ma trận A trong ma trận sinh dạng chuẩn G.

Để chứng minh định lý này chỉ cần chỉ ra rằng với ma trận H xác định như trên thì H có các hàng độc lập tuyến tính và có $G.\, H^T = O$. Theo định lý 3.4 thì H là ma trận kiểm tra của mã C.

Sinh viên có thể chứng minh định lý này này xem như là bài tập.

- Chú ý rằng với các mã nhị phân tuyến tính thì công thức (3.4) có dạng:

$$H = (A^T \mid I_{n-k}) \qquad (3.4)'$$

Thí dụ 3.14

Cho mã C[4, 2] trên trường F_3, có ma trận sinh $G = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \end{bmatrix}$, dễ thấy ma trận sinh G ở dạng chuẩn. Vậy theo định lý 3.5, ta có ma trận kiểm tra của mã C là :

$H = (-A^T \mid I_2)$, với ma trận $A^T = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}$, ta có $-A^T = \begin{bmatrix} -1 & -1 \\ -1 & -2 \end{bmatrix} = \begin{bmatrix} 2 & 2 \\ 2 & 1 \end{bmatrix}$.

Vậy ma trận kiểm tra của mã C là H = $\begin{bmatrix} 2 & 2 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{bmatrix}$, ma trận kiểm tra H ở dạng chuẩn.

Thí dụ 3.15 Cho mã tuyến tính C[7, 4, 3] có ma trận sinh ở dạng chuẩn G = $\begin{bmatrix} 1\,0\,0\,0\,1\,0\,1 \\ 0\,1\,0\,0\,1\,1\,1 \\ 0\,0\,1\,0\,1\,1\,0 \\ 0\,0\,0\,1\,0\,1\,1 \end{bmatrix}$, tìm ma trận kiểm tra của mã C.

Giải: theo công thức (3.4)'định lý 3.5, do C là mã nhị phân tuyến tính ta có ma trận kiểm tra của mã C ở dạng chuẩn là :

$$H = (A^T \mid I_3)$$

do A = $\begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$ ta có A^T = $\begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix}$

Vậy H = $\begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$ là ma trận kiểm tra của mã C, đã ở dạng chuẩn.

Nhận xét:

1. Từ định lý 3.5, nếu biết ma trận sinh G ở dạng chuẩn của mã tuyến tính nhị phân C[n, k], thì có thể tìm được ma trận kiểm tra H ở dạng chuẩn.

2. Ngược lại, nếu biết ma trận kiểm tra H ở dạng chuẩn thì có thể tìm được ma trận sinh liên kết G ở dạng chuẩn. Cụ thể, nếu H = ($B \mid I_k$) thì G = ($I_{n-k} \mid -B^T$). Tức là, với mã C, nếu biết một trong hai ma trận G và H ở dạng chuẩn thì có thể tìm được ma trận kia. Điều này chứng tỏ 2 ma trận G và H có liên quan chặt chẽ với nhau.

Thí dụ 3.16 Cho mã C[7, 4] có ma trận sinh như sau:

G = $\begin{bmatrix} 1\,1\,1\,1\,1\,1\,1 \\ 1\,0\,0\,0\,1\,0\,1 \\ 1\,1\,0\,0\,0\,1\,0 \\ 0\,1\,1\,0\,0\,0\,1 \end{bmatrix}$. Hãy tìm ma trận kiểm tra của mã C[7, 4].

Bước 1: Trước hết đưa ma trận G về ma trận tương đương ở dạng chuẩn nhờ các phép biến đổi tương đương *chỉ theo hàng*:

G' = $\begin{bmatrix} 1\,0\,0\,0\,1\,0\,1 \\ 0\,1\,0\,0\,1\,1\,1 \\ 0\,0\,1\,0\,1\,1\,0 \\ 0\,0\,0\,1\,0\,1\,1 \end{bmatrix}$

Bước 2 : Do G' là ma trận sinh của mã C, đã ở dạng chuẩn, G' = (I_4 | A), trong đó ma trận A là :

$$A = \begin{bmatrix} 1\,0\,1 \\ 1\,1\,1 \\ 1\,1\,0 \\ 0\,1\,1 \end{bmatrix}$$

Theo định lý 3.4, ma trận kiểm tra của mã C có dạng H = (A^T | I_3), ta có:

$$H = \begin{bmatrix} 1\ 1\ 1\ 0\ 1\ 0\ 0 \\ 0\ 1\ 1\ 1\ 0\ 1\ 0 \\ 1\ 1\ 0\ 1\ 0\ 0\ 1 \end{bmatrix}$$

Phát hiện lỗi bằng ma trận kiểm tra.

Định lý 3.6

Cho H là ma trận kiểm tra của mã nhị phân tuyến tính C[n, k]. Một xâu bit x độ dài n là một từ mã của C khi và chỉ khi:

$$x.H^T = O \qquad (3.5)$$

trong đó: H^T là ma trận chuyển vị của H, O là ma trận không (gồm toàn số không)

Chứng minh:

- Điều kiện cần: Giả sử $x = (x_1\ x_2\ \ldots\ x_n)$ là một từ mã trong C. Do đó x phải trực giao với mọi hàng của ma trận H, hay trực giao với mọi cột của ma trận H^T. Từ đó ta có $x.H^T = O$.

- Điều kiện đủ: Giả sử x thỏa mãn $x.H^T = O$, điều này chứng tỏ x trực giao với tất cả các cột của ma trận H^T, hay x trực giao với các hàng của H, theo kết quả của đại số tuyến tính, x thuộc không gian con trực giao với không gian con sinh bởi H. Do H là ma trận sinh của mã đối ngẫu $C^{\perp}$ nên suy ra $x \in C$, tức là x là một từ mã trong C.

Nhận xét:

1. Từ định lý 3.6, ta có phương pháp phát hiện lỗi như sau:

- Giả sử một từ mã $c \in C$ được gửi đi, ta nhận được xâu x, để xác định xâu x có phải là từ mã hợp lệ hay không, ta kiểm tra điều kiện (3.5): $x.H^T = O$.

Nếu điều kiện trên thỏa mãn (vế phải là một ma trận không) ta kết luận x là một từ mã hợp lệ của C. Nếu vế phải $\neq O$, thì xâu nhận được không phải là một từ mã trong C.

2. Dễ thấy rằng điều kiện $x.H^T = O$ là tương đương với điều kiện:

$$H.x^T = O, \qquad (3.6)$$

(trong đó x^T là ma trận cột chuyển vị của xâu x)

Việc thực hiện tính toán với điều kiện (3.6) là dễ dàng hơn (3.5), nên ta thường kiểm tra điều kiện này , thay cho việc kiểm tra điều kiện (3.5)

Định nghĩa 3.9

Cho mã tuyến tính C[n, k] có ma trận kiểm tra H, với mọi xâu bit x có độ dài n, biểu thức $H.x^T$ gọi là Syndrome của x, (hay vector hội chứng của x), ký hiệu là S(x)

Như vậy, để kiểm tra x có phải là 1 từ mã trong C hay không, ta kiểm tra điều kiện S(x) = O.

Nhận xét:

1. Dễ thấy rằng với hai xâu bit x và y cùng độ dài thì: $S(x + y) = S(x) + S(y)$
2. Với mọi ma trận kiểm tra H, với O là xâu toàn bit '0' thì S(O) = O.

Thí dụ 3.17

Giả sử một mã nhị phân tuyến tính C[7, 4] có ma trận kiểm tra là H = $\begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$

1. Giả sử nhận được xâu x = 0001110. Hỏi x có phải là một từ mã thuộc C hay không?

Giải: Tính S(x) = H.x^T.

Giải: ta tìm Syndrome của x:

$$S(x) = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

S(x) ≠ O, vậy x không phải là một từ mã.

2. Giả sử nhận được xâu x = 0001011. Hỏi x có phải là một từ mã thuộc C hay không?

Giải: Tính S(x) = $H.x^T$

$$S(x) = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

S(x) = O, vậy x là một từ mã.

3.3.3 Sửa lỗi bằng ma trận kiểm tra.

Ma trận kiểm tra không những được dùng để phát hiện lỗi mà nó còn được dùng để sửa lỗi.

Giả sử từ mã $c = (c_1\ c_2\ \dots\ c_n)$ là từ mã gửi đi, xâu nhận được là $x = (x_1\ x_2\ \dots\ x_n)$, gọi e là vector lỗi, với $e = (e_1\ e_2\ \dots\ e_n)$, trong đó: c_i , x_i và $e_i \in \{0, 1\}$

Rõ ràng là nếu $x = c$ thì xâu nhận được không có lỗi, khi đó có thể viết $x = c + e$, với e = O (xâu toàn bit '0').

Trái lại, nếu x có lỗi ở bit thứ j, thì x và c sẽ khác nhau ở bit thứ j, thì có thể viết: $x = c + e$ với e có bit thứ j bằng '1', các bit khác bằng '0'.

Thật vậy: giả sử gửi đi từ mã c = 1110001 ta nhận được x = 1110000, có lỗi xảy ra ở bit thứ 7.

Ta có thể viết : 1110000 = 1110001 + 00000001; hay là $x = c + e$ với e = 0000001

Bây giờ ta giả sử khi từ mã c được gửi đi ta nhận được xâu x, với chỉ có duy nhất 1 lỗi xảy ra, ta sẽ chứng minh rằng khi đó ta có thể sửa được lỗi này, và tìm được chính xác từ mã đã gửi đi.

Bổ đề 3.2

> *Nếu mã tuyến tính C[n, k] có ma trận kiểm tra H không có cột bằng O và không có 2 cột nào giống nhau, thì có thể sửa được một lỗi xảy ra khi gửi đi một từ mã bất kỳ.*

Chứng minh:

Giả sử khi từ mã c được gửi đi ta nhận được xâu x, với chỉ có duy nhất 1 lỗi xảy ra.

Giả sử lỗi xảy ra ở bit thứ j, có thể viết: $x = c + e$, với vector lỗi e chỉ có duy nhất một bit khác không ở vị trí thứ j, e = (00 …010…0) , với bit thứ j bằng '1'

Ta tính Syndrome của x:

$$S(x) = S(c) + S(e),$$

do c là từ mã nên S(c) = O, ta có : $S(x) = S(e) = H.e^T$ = H. [0 0 …1..0]^T = h_j

Trong đó h_j là cột thứ j của ma trận H. Do H không có hai cột giống nhau, nên vị trí j là duy nhất.

Như vậy việc tính S(x) sẽ xác định được vị trí bit bị sai, ứng với cột thứ j của ma trận H, từ đó có thể sửa được lỗi này. (đpcm)

Từ đó, nếu chúng ta nhận được x và giả sử không có quá một lỗi xuất hiện, ta có thể tìm được từ mã được gửi đi là c bằng cách tính Syndrome của x, $S(x) = Hx^t$. Nếu $S(x) = O$, ta biết x là từ mã được gửi . Nếu $S(x) \neq O$, nó sẽ bằng cột thứ j nào đó của H (nếu H không có 2 cột nào giống nhau), từ đó ta biết rằng bit thứ j đã bị thay đổi, và ta có thể thay đổi giá trị bit này để sửa lỗi và nhận được chính xác từ mã đã gửi đi.

Thí dụ 3.15

Cho mã tuyến tính C[7, 4] có ma trận sinh là

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

Dùng ma trận kiểm tra để xác định từ mã nào trong mã C được gửi đi, nếu ta nhận được xâu x = 0011111. Giả sử x mắc nhiều nhất một lỗi xảy ra trên đường truyền.

Giải:

Do G có dạng chuẩn, G = (I_4 | A), với ma trận

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

Theo định lý 3.4, từ G, ta có ma trận kiểm tra H = (A^T | I_3), vậy:

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Tính Syndrome của x:

$$S(x) = H.x^T = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

Syndrome của x khác O, vậy x không phải là một từ mã (*x* có lỗi). Do S(x) trùng với cột thứ 6 của ma trận kiểm tra H, nên ta có j = 6 là vị trí của bit bị lỗi trong xâu x. Từ đó có thể sửa được lỗi này và xác định chính xác từ mã đã gửi đi là c = 0011101. Đó là từ mã thứ tư của bộ mã C.

3.4 MÃ HAMMING

3.4.1 Mã Hamming nhị phân cấp r

Mã Hamming là một loại mã tuyến tính do R. W. Hamming đề xuất được sử dụng khá phổ biến trong các hệ thống truyền tin sử dụng kỹ thuật FEC (Forward Error Corection). Mã này có khả năng sửa sai các từ mã bị sai chỉ 1 lỗi khi truyền tin.

Mã Hamming có thể được định nghĩa trên mọi trường GF(q). Để đơn giản, trong phần này, chúng ta chỉ xét các mã mã Hamming nhị phân cấp *r*.

Do đặc điểm mã hóa và giải mã khá đơn giản, nên mã Hamming được sử dụng khá rộng rãi trong thực tế

Định nghĩa 3.10

Mã Hamming nhị phân cấp r, ký hiệu là Ham(r, 2), là một mã tuyến tính có ma trận kiểm tra H cấp r× (2^r–1), (r là số nguyên dương) có các cột là các xâu nhị phân có giá trị lần lượt là 1, 2, ... 2^r–1.

Nhận xét:

1. Ma trận kiểm tra H của mã Hamming không có cột gồm toàn bit ‘0’và không có 2 cột giống nhau.

2. Mã Ham(r, 2) là một mã tuyến tính nhị phân C[n, k], với các liên hệ giữa *r, n* và *k*:

- $n = 2^r - 1;$
- $k = 2^r - 1 - r;$
- $r = n - k$.
- Ma trận sinh của mã Ham(r, 2) là ma trận G cấp $k \times n$

Như vậy mã Hamming có chức năng mã hóa các thông điệp gồm *k* bit thành các từ mã độ dài *n*, với $n = 2^r - 1$. Cấp của mã Hamming là r, với r là số bit thêm vào để chống nhiễu, còn gọi là các bit kiểm tra.

Thí dụ 3.16

1. Với r = 2, ta có ma trận H = $\begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$ là ma trận của mã Hamming cấp 2, Ham(2, 2)

2. Cho mã nhị phân tuyến tính C có ma trận kiểm tra sau:

$$H = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

C là một mã Hamming cấp 3, Ham(3, 2)

Với $r = 3$, $n = 2^3 - 1 = 7$, $k = n - r = 4$, vậy C là mã tuyến tính C[7, 4].

Bổ đề 3.3

> *Các mã Hamming nhị phân cấp r, Ham(r, 2), đều có khoảng cách cực tiểu bằng 3, vì vậy có thể phát hiện sai hai bit và sửa sai 1 bit.*

Chứng minh:

Ta biết rằng khoảng cách Hamming cực tiểu của mã nhị phân tuyến tính bằng trọng số cực tiểu của một từ mã khác O, tức là: $d(C) = min\{wt(\omega) \mid \omega \neq O, \omega \in C\}$.

Để chứng minh bổ đề , ta sẽ chỉ ra rằng mã Ham(r, 2) có từ mã khác O có trọng số cực tiểu là 3.

Giả sử có từ mã *x* có trọng số nhỏ nhất là bằng 1. Khi đó *x* có dạng:

x = 000 ... 010 . . .00 (bit '1' ở vị trí thứ j)

Gọi hàng thứ i của ma trận H là h_i, có dạng: $h_i = h_{i1}\, h_{i2} \ldots h_{ij} \ldots h_{in}$, $\forall i = 1, 2, ..., r$.

Vì x là từ mã trong C nên x phải trực giao với mọi vector hàng của ma trận kiểm tra H, tức là:

$$x.\, h_i = 0\ ,\ \forall i = 1, 2, ..., r$$

$$\Leftrightarrow \quad h_{ij} = 0\ ,\ \forall i = 1, 2, ..., r$$

Tức là cột thứ *j* của ma trận H chứa toàn bit '0', trái với định nghĩa mã Ham(*r, 2*). Vậy không có từ mã có trọng số 1.

Bây giờ giả sử có từ mã y có trọng số nhỏ nhất là bằng 2. Khi đó y có dạng:

y = 0 0 0 ... 010 . . .010 . . . 00 (bit '1' ở vị trí thứ j và *s*)

do *y* là từ mã trong C nên *x* phải trực giao với mọi vector hàng của ma trận kiểm tra H, tức là:

$$y.\, h_i = 0\ ,\ \forall i = 1, 2, ..., r$$

$$\Leftrightarrow \quad h_{ij} + h_{is} = 0\ ,\ \forall i = 1, 2, ..., r$$

$$\Leftrightarrow \quad h_{ij} = h_{is}\ ,\ \forall i = 1, 2, ..., r$$

Tức là cột thứ j và cột thứ s của ma trận H là giống nhau, trái với định nghĩa mã Ham(r, 2). Vậy không có từ mã có trọng số 2.

Bây giờ ta chỉ ra rằng trong mã Ham(r, 2) có từ mã trọng số 3, là trọng số cực tiểu của các từ mã trong mã này.

Ba cột đầu của H có dạng:

$$\begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{matrix}$$

Với xâu x = 1 1 1 0 ... 00 (chỉ 3 bit đầu là '1', các bit còn lại là '0'), ta luôn có $S(x) = 0$. Vậy x là một từ mã trong Ham(r, 2), với $wt(x) = 3$, đây là từ mã có trọng số nhỏ nhất, vậy d(C) = 3.

Nhận xét:

Từ các kết quả trên, suy ra mã ***Ham(r, 2)*** là một mã tuyến tính $C[2^r - 1\ ,\ 2^r - 1 - r\ ,\ 3]$

3.4.2 Ma trận sinh của mã Hamming

Mã Ham(r, 2) có ma trận kiểm tra là H cấp $r \times (2^r - 1)$. Ta sử dụng các phép biến đổi tương đương chỉ theo hàng để đưa ma trận kiểm tra về dạng chuẩn:

$$H = (B \mid I_r)$$

Từ ma trận kiểm tra H ở dạng chuẩn, dễ dàng xác định được ma trận sinh dạng chuẩn của mã Ham(r, 2) như sau:

$$G = (I_{n-r} \mid B^T)$$

Thí dụ 3.17

1. Tìm ma trận sinh của mã Ham(3, 2)
2. Mã hóa thông điệp $u = 1011$ bằng mã Ham(3, 2)

- Giải 1: Tìm ma trận sinh của mã Ham(3, 2).

Mã Ham(3, 2) có ma trận kiểm tra là:

$$H = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Ta đưa ma trận kiểm tra H về dạng chuẩn nhờ các phép biến đổi tương đương *theo hàng*:

$$H = \begin{bmatrix} 0\ 1\ 1\ 1\ 1\ 0\ 0 \\ 1\ 0\ 1\ 1\ 0\ 1\ 0 \\ 1\ 1\ 0\ 1\ 0\ 0\ 1 \end{bmatrix}$$

Từ đó có ma trận sinh G của mã Hamming nhị phân cấp 3 là:

$$G = \begin{bmatrix} 1\ 0\ 0\ 0\ 0\ 1\ 1 \\ 0\ 1\ 0\ 0\ 1\ 0\ 1 \\ 0\ 0\ 1\ 0\ 1\ 1\ 0 \\ 0\ 0\ 0\ 1\ 1\ 1\ 1 \end{bmatrix}$$

- Giải 2: Do mã Ham(3, 2) có ma trận sinh G, xâu bit *u* được mã hóa thành từ mã nhờ công thức: $w = u.G$. Vậy:

$$w = (1\ 0\ 1\ 1) \cdot \begin{bmatrix} 1\ 0\ 0\ 0\ 0\ 1\ 1 \\ 0\ 1\ 0\ 0\ 1\ 0\ 1 \\ 0\ 0\ 1\ 0\ 1\ 1\ 0 \\ 0\ 0\ 0\ 1\ 1\ 1\ 1 \end{bmatrix} = (1\ 0\ 1\ 1\ 0\ 1\ 0)$$

Chú ý:

1. Một mã Ham(r, 2) có thể có nhiều ma trận sinh khác nhau (các ma trận sinh này đều tương đương), cho nên việc mã hóa cùng 1 thông điệp *u* với các ma trân sinh khác nhau có thể cho các từ mã khác nhau, tuy nhiên các từ mã này đều thuộc bộ mã đã cho.
2. Việc mã hóa và giải mã phải dùng cùng một ma trận sinh.

3.4.3 Phát hiện lỗi và sửa lỗi bằng mã Hamming.

Giả sử khi gửi đi một từ mã của bộ mã Hamming nhị phân cấp r, ta nhận được xâu *x*. Nếu xâu *x* chỉ bị sai tối đa 1 lỗi thì ta có thể phát hiện lỗi và sửa lỗi theo các bước sau:

Bước 1: Tính Syndrome của x: $S(x) = H.x^T$

Bước 2: Nếu có $S(x) = O$, tức là nhận được cột gồm *r* bit '0'. Kết luận *x* chính là từ mã được gửi.

Bước 3: Nếu có $S(x) \neq O$, tức là nhận được 1 cột xâu nhị phân tương ứng với 1 cột thứ *j* của ma trận kiểm tra H, từ đó suy ra bit thứ *j* của xâu *x* bị nhận sai và có thể sửa lỗi bằng cách cộng 1 vào bit đó.

Chú ý rằng, ma trận kiểm tra H có các cột lần lượt là các xâu bit nhị phân có giá trị là *1, 2, ..., n,* nên giá trị các xâu bit chính là vị trí của cột đó. Vì vậy, ở bước 3, nếu giá trị xâu bit $S(x)$ là *j* sẽ cho vị trí *j* của bit bị nhận sai.

Thí dụ 3.18.

Cho C là mã Ham(3, 2), với ma trận kiểm tra $H = \begin{bmatrix} 0\ 0\ 0\ 1\ 1\ 1\ 1 \\ 0\ 1\ 1\ 0\ 0\ 1\ 1 \\ 1\ 0\ 1\ 0\ 1\ 0\ 1 \end{bmatrix}$, khi gửi đi một từ mã của C, nếu xảy ra một trong các trường hợp:

1. Nhận được xâu $x = 1011010$
2. Nhận được xâu $y = 1011011$
3. Nhận được xâu $z = 1101011$

Giả sử các xâu nhận được chỉ sai tối đa 1 lỗi. Tìm từ mã được gửi đi ứng với mỗi trường hợp?

Giải:

1. Khi nhận được xâu x = 1011010, tính $S(x) = H.x^T$:

$$S(x) = \begin{bmatrix} 0&0&0&1&1&1&1 \\ 0&1&1&0&0&1&1 \\ 1&0&1&0&1&0&1 \end{bmatrix} . \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Vậy xâu x nhận được không có lỗi, 1011010 là từ mã được gửi.

2. Khi nhận được xâu y = 1011011, tính $S(y) = H.y^T$:

$$S(y) = \begin{bmatrix} 0&0&0&1&1&1&1 \\ 0&1&1&0&0&1&1 \\ 1&0&1&0&1&0&1 \end{bmatrix} . \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

Xâu bit của S(y) là '1 1 1' có giá trị là 7. Vậy xâu y bị sai ở bit thứ 7, là bit cuối cùng, sửa sai bit này, từ đó ta có từ mã được gửi là: 1011010.

3. Khi nhận được xâu z = 1101011, tính $S(z) = H.z^T$ bằng cách tương tự, ta có xâu bit của S(z) là '110', có giá trị là 6, vậy bit sai là bit thứ 6, sau khi sửa lỗi, ta có từ mã gửi đi là 1101001.

3.4.4 Giới thiệu mã Hamming q-phân cấp r

Trên đây ta đã xét các mã Hamming nhị phân. Trường hợp tổng quát, trên trường GF(q), ta có mã Hamming q-phân cấp r (*q-ary Hamming code*) được định nghĩa như sau:

Định nghĩa 3.11

Mã Hamming q-phân cấp r, ký hiệu Ham(r, q) là một mã tuyến tính có ma trận kiểm tra H có r hàng, và có $(q^r-1)/(q-1)$ cột, không có cột nào bằng O, không có 2 cột giống nhau và giá trị khác không đầu tiên của mỗi cột đều bằng 1.

- Mã Hamming q- phân cấp r là một mã tuyến tính $C[(q^r-1)/(q-1) , (q^r- 1)/(q – 1) – r , 3]$,

Thí dụ 3.19.

1. Mã Ham(2, 3) có ma trận kiểm tra là: $H = \begin{bmatrix} 0&1&1&1 \\ 1&0&1&2 \end{bmatrix}$

2. Mã Ham(3, 3) có ma trận kiểm tra là

$$H = \begin{bmatrix} 0&0&0&0&1&1&1&1&1&1&1&1&1 \\ 0&1&1&1&0&0&0&1&1&1&2&2&2 \\ 1&0&1&2&0&1&2&0&1&2&0&1&2 \end{bmatrix}$$

BÀI TẬP CHƯƠNG 3

1. Giả sử, với mã lặp lại 3 lần được dùng, nếu ta nhận được xâu x = 100110011100111. Có thể kết luận gì với thông điệp được gửi đi?

2. Giả sử có một xâu bit được gửi đi trên kênh đối xứng nhị phân, xác suất nhận sai là 0.001. Vậy xác suất để nhận được xâu bit 100110001 là bao nhiêu khi xâu bit gửi đi là:

a/. 101110011

b/. 110111011

c/. 110111010.

3. Cho mã C = {0000000,0101000, 1000111}

a/. Sử dụng *quy tắc giải mã người láng giềng gần nhất* để xác định từ mã nào được gửi đi nếu nhận được xâu x = 0101001.

b/. Giả sử các xâu bit của C được gửi đi trên kênh đối xứng nhị phân, xác suất nhận sai là 0.01. Vậy xác suất để nhận được xâu bit x trên là bao nhiêu khi xâu bit gửi đi là c_x được xác định bằng phương pháp trên?

4. Có thể phát hiện và sửa bao nhiêu lỗi trong các mã sau:

a/. C = {11101,10011}

b/. C = {111001,100011,00011}

c/. C = {11101101,10001111,10011011, 11011011}

5. Giả sử C là mã {000000001,111110100,010101111, 100001111}. C có thể phát hiện và sửa bao nhiêu lỗi ?

6. Tìm ma trận kiểm tra H của một mã khi ta thêm một bit kiểm tra chẵn lẻ vào một một thông điệp nhị phân có độ dài bằng 5.

7. Tìm ma trận kiểm tra H của một mã lặp lại 3 lần đối với một thông điệp nhị phân có độ dài bằng 4.

8. Cho mã nhị phân tuyến tính C có ma trận sinh G = $\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$

a/. Mã C sẽ mã hoá các thông điệp nào thành các từ mã của C. Tìm tất cả các từ mã của C?
b/. Bảng coset chuẩn của mã C gồm bao nhiêu hàng, bao nhiêu cột. Xây dựng bảng coset chuẩn của mã C.

c/. Gửi đi một từ của bộ mã C, nhận được xâu x = 111, dùng bảng coset chuẩn xác định từ mã nào đã được gửi đi. Cũng hỏi như vậy nếu nhận được xâu y = 001, xác định vector lỗi (nếu có) trong trường hợp này.

9. Cho mã nhị phân tuyến tính C có ma trận sinh G = $\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$

a/. Mã C sẽ mã hoá các thông điệp nào thành các từ mã của C. Tìm tất cả các từ mã của C?
b/. Bảng coset chuẩn của mã C gồm bao nhiêu hàng, bao nhiêu cột. Xây dựng bảng coset chuẩn của mã C.

c/. Gửi đi một từ của mã C, nhận được xâu x = 0111, dùng bảng coset chuẩn xác định từ mã nào đã được gửi đi. Cũng hỏi như vậy nếu nhận được xâu y = 1110, xác định vector lỗi (nếu có) trong trường hợp này.

10. Cho mã nhị phân tuyến tính C có ma trận sinh G = $\begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{bmatrix}$

a/. Mã C sẽ mã hoá các thông điệp nào thành các từ mã của C. Tìm tất cả các từ mã của C?
b/. Bảng coset chuẩn của mã C gồm bao nhiêu hàng, bao nhiêu cột. Tìm 6 hàng đầu tiên của bảng coset chuẩn của mã C.
c/. Gửi đi một từ của mã C, nhận được xâu x = 11111, dùng bảng coset chuẩn xác định từ mã nào đã được gửi đi. Cũng hỏi như vậy nếu nhận được xâu y = 01111, xác định vector lỗi (nếu có) trong trường hợp này.

11. Giả sử rằng mã C có ma trận kiểm tra:

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}$$

a/. Tìm ma trận sinh **G** của mã C.
b/. Những từ mã nào trong bộ mã C được tạo ra bởi ma trận ma trận sinh này?
c/. Mã C với ma trận kiểm tra H trên đây có phải là một mã Hamming không?

12. Tìm 16 từ mã trong mã Hamming nhị phân cấp 3.

13. Tìm ma trận kiểm tra của mã Ham(2, 2). Từ ma trận kiểm tra, viết ma trận sinh của mã này và liệt kê tất cả các từ của mã.

14. Cho ma trận sinh của mã tuyến tính C[7, 4] trên trường F_2:

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

a/. Tìm tất cả các từ mã sinh bởi G.
b/. Khi gửi đi một từ mã của C, ta nhận được xâu *x* = 1011011. Giả sử xâu nhận được sai không quá 1 bit, tìm từ mã đã gửi đi.

15. Cho mã tuyến tính C[7, 3] có ma trận sinh: G = $\begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$

a/. Tìm tất cả các từ mã của mã C. ĐA: C = {0000000; 0010111; 0101011; 0111100; 1001101; 1011010; 1100110; 1110001}

b/. Tìm ma trận kiểm tra của mã C. ĐA: Ma trận kiểm tra H = $\begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$

c/. Mã C có phải là một mã Ham(r, 2) hay không?

d/. Gủi đi 1 từ mã của C, nhận được xâu x = 1110000. Giả sử xâu nhận được sai không quá 1 bit, tìm từ mã đã gửi đi.

e/. Cũng hỏi như câu d, với xâu nhận được y = 1011010

16. Trên trường F_3, cho C là mã Ham(2, 3) có ma trận kiểm tra là: H = $\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 2 \end{bmatrix}$

a/. Hãy tìm ma trận sinh của mã C nói trên.
b/. Viết tất cả các từ mã của mã C.

TÀI LIỆU THAM KHẢO

[1] Raymond Hill, (1993). *"A First Course in Coding Theory"*, Clarendon Press, Oxrord, USA. ISBN : 0-19-853803-0.

[2] Yehuda Lindell, *"Introduction to Coding Theory",* Lecture Notes, Department of Computer Science Bar-Ilan University , Israel (2010).

[3] Tom Richardson, Rudiger Urbanke. *"Modern Coding Theory".* Cambridge University Press (2008)

[4] Đặng Văn Chuyết, Nguyễn Tuấn Anh.(1998) *"Giáo trình Cơ sở lý thuyết truyền tin".* NXB Giáo dục.