**Tuyệt vời! Đây là một thuật toán mã hóa khối tùy chỉnh khá thú vị. Dưới đây là cách giải thích từng bước một cách dễ hiểu hơn, dành cho một người chơi Crypto CTF:**

**Tổng quan về Thuật toán**

**Thuật toán này mã hóa một thông điệp văn bản bằng cách:**

1. **Chuẩn bị: Chuyển chữ cái thành số, đệm thêm ký tự 'x' cho đủ độ dài, rồi chia thành các khối 12 ký tự.**
2. **Biến đổi khối thành ma trận: Mỗi khối 12 ký tự được chuyển thành một ma trận 6x6 chứa các "trit" (giá trị 0, 1, hoặc 2).**
3. **Xáo trộn ma trận: Ma trận trit này được xáo trộn nhiều lần dựa trên các ký tự của khóa bí mật. Mỗi ký tự khóa chọn một phép hoán vị và một phép cộng đặc biệt.**
4. **Trích xuất khối mã hóa: Từ ma trận đã xáo trộn, ta đọc ra 12 ký tự mã hóa mới.**
5. **Hoán vị cuối cùng: Tất cả các khối 12 ký tự mã hóa được ghép lại, sau đó trải qua một phép hoán vị cột cuối cùng dựa trên khóa.**

**Giải mã là thực hiện chính xác các bước ngược lại.**

**Chi tiết các bước MÃ HÓA**

**1. Mở đầu (Chuẩn bị thông điệp)**

* **Ánh xạ ký tự → số:** 
  + **Chuyển mỗi chữ cái thường (a-z) thành một số từ 1 đến 26.**
  + **Công thức: số = ord(chữ cái) - 96. (Ví dụ: 'a' → 1, 'b' → 2, ..., 'z' → 26).**
* **Đệm (Padding):** 
  + **Thêm các ký tự 'x' vào cuối thông điệp gốc cho đến khi tổng độ dài của nó chia hết cho 12.**
* **Chia khối (Blocking):** 
  + **Chia thông điệp đã đệm thành các khối, mỗi khối có đúng 12 ký tự.**

**2. Xây dựng ma trận "trit" 6x6 từ mỗi khối 12 ký tự**

**Mỗi khối 12 ký tự (gọi là L₀L₁…L₁₁) sẽ được chuyển thành một ma trận blockM kích thước 6x6 chứa các "trit".**

* **Chuyển đổi ký tự thành 3 trit:** 
  + **Mỗi ký tự (đã được ánh xạ thành số từ 1-26) được biểu diễn bằng 3 trit (giá trị 0, 1, hoặc 2) trong hệ cơ số 3.**
  + **Nếu value là giá trị số của ký tự:** 
    - **q₀ = value // 9 (phần nguyên khi chia value cho 9)**
    - **q₁ = (value % 9) // 3 (phần nguyên khi chia (phần dư của value chia 9) cho 3)**
    - **q₂ = value % 3 (phần dư khi chia value cho 3)**
  + **Ví dụ: 'm' (value=13) → q₀=1, q₁=1, q₂=1 (vì 13 = 1*9 + 1*3 + 1)**
* **Điền vào ma trận blockM (6x6):** 
  + **Nửa trên ma trận (Hàng 0, 1, 2):** 
    - **Lấy 6 ký tự đầu tiên của khối (L₀ đến L₅).**
    - **Với mỗi ký tự Lᵢ (trong số L₀ đến L₅):** 
      * **Ba trit q₀(Lᵢ), q₁(Lᵢ), q₂(Lᵢ) của nó sẽ điền vào cột i của ma trận.**
      * **blockM[0, i] = q₀(Lᵢ)**
      * **blockM[1, i] = q₁(Lᵢ)**
      * **blockM[2, i] = q₂(Lᵢ)**
  + **Nửa dưới ma trận (Hàng 3, 4, 5):** 
    - **Lấy 6 ký tự tiếp theo của khối (L₆ đến L₁₁).**
    - **Với mỗi ký tự L₆₊ᵢ (trong số L₆ đến L₁₁):** 
      * **Ba trit q₀(L₆₊ᵢ), q₁(L₆₊ᵢ), q₂(L₆₊ᵢ) của nó sẽ điền vào cột i của ma trận (trong nửa dưới).**
      * **blockM[3, i] = q₀(L₆₊ᵢ)**
      * **blockM[4, i] = q₁(L₆₊ᵢ)**
      * **blockM[5, i] = q₂(L₆₊ᵢ)**

**Kết quả: blockM là một ma trận 6x6 chứa đầy các trit.**

**3. "Xáo trộn" ma trận blockM**

**Bước này dùng khóa bí mật để làm rối ma trận blockM. Khóa là một chuỗi ký tự (ví dụ: k₀k₁…kₘ₋₁).**

* **Chuẩn bị từ khóa:** 
  + **Với mỗi ký tự kⱼ trong khóa:** 
    - **keyNum = ord(kⱼ) – 97 (cho ra số từ 0-25, 'a'→0, 'b'→1,...).**
    - **permuteIndex = (keyNum // 5) % 5 (ra số từ 0-4, để chọn 1 trong 5 phép hoán vị A,B,C,D,E).**
    - **addIndex = keyNum % 5 (ra số từ 0-4, để chọn 1 trong 5 quy tắc cộng).**
* **Quá trình xáo trộn lặp lại:** 
  + **Lặp qua từng ký tự của khóa, từ trái sang phải. Với mỗi ký tự khóa:** 
    - **Hoán vị (Permute): Áp dụng phép hoán vị permuteIndex đã chọn lên toàn bộ 36 ô của blockM. (Tức là, các trit trong blockM sẽ đổi chỗ cho nhau theo một trong 5 mẫu A,B,C,D,E).**
    - **Cộng (Add) modulo 3: Áp dụng quy tắc cộng addIndex đã chọn cho các trit trong blockM, tất cả các phép cộng đều là mod 3.** 
      * **addIndex = 0: Cộng 1 (mod 3) vào mọi ô blockM[i,j] nếu (i + j) là số chẵn.**
      * **addIndex = 1: Khối 3x3 dưới-phải += khối 3x3 trên-trái (cộng từng ô tương ứng, mod 3).**
      * **addIndex = 2: Khối 3x3 trên-trái += khối 3x3 dưới-phải (mod 3).**
      * **addIndex = 3: Khối 3x3 dưới-trái += khối 3x3 trên-phải (mod 3).**
      * **addIndex = 4: Khối 3x3 trên-phải += khối 3x3 dưới-trái (mod 3).**

**Sau khi xử lý hết các ký tự trong khóa, blockM đã bị xáo trộn.**

**4. Trích xuất 12 ký tự mã hóa từ blockM**

**Từ ma trận blockM đã xáo trộn, ta đọc ra 12 ký tự mới.**

* **Đọc theo hàng, mỗi hàng 2 ký tự:** 
  + **Với mỗi hàng i (từ 0 đến 5):** 
    1. **Ký tự thứ nhất từ hàng i:** 
       - **Lấy 3 trit đầu tiên của hàng: blockM[i,0], blockM[i,1], blockM[i,2].**
       - **Tính giá trị số: num = 9 \* blockM[i,0] + 3 \* blockM[i,1] + 1 \* blockM[i,2]. (Đây là chuyển từ cơ số 3 về cơ số 10, giá trị có thể từ 0 đến 26).**
       - **Nếu num == 0, ký tự là '0'. Ngược lại, ký tự là chr(num + 96) (tức là 1→'a', 2→'b',...).**
    2. **Ký tự thứ hai từ hàng i:** 
       - **Lấy 3 trit tiếp theo của hàng: blockM[i,3], blockM[i,4], blockM[i,5].**
       - **Tính giá trị số tương tự: num = 9 \* blockM[i,3] + 3 \* blockM[i,4] + 1 \* blockM[i,5].**
       - **Chuyển num thành ký tự ('0' hoặc a-z) như trên.**
* **Kết quả: 6 hàng, mỗi hàng 2 ký tự → tổng cộng 12 ký tự mã hóa cho khối này. Gọi chuỗi này là resultLetters.**

**5. Phép hoán vị cột cuối cùng**

* **Ghép nối: Nối tất cả các chuỗi resultLetters (12 ký tự/khối) từ tất cả các khối lại thành một chuỗi dài R.**
* **Chuẩn bị khóa cho hoán vị cột:** 
  + **keyNums = [ord(k) – 97 cho từng ký tự k của khóa gốc].**
  + **reducedKeyNums: Tạo danh sách mới bằng cách duyệt keyNums từ trái sang phải, chỉ giữ lại giá trị đầu tiên xuất hiện (loại bỏ các giá trị trùng lặp về sau).**
  + **N = độ dài của reducedKeyNums.**
* **Hoán vị cột:** 
  + **Chuẩn bị N "hộp" (hoặc cột) rỗng.**
  + **Phân phối các ký tự của chuỗi R vào N hộp này theo kiểu round-robin (chia lần lượt):** 
    - **R[0] vào hộp 0, R[1] vào hộp 1, ..., R[N-1] vào hộp N-1,**
    - **R[N] vào lại hộp 0, R[N+1] vào hộp 1, ...**
  + **Xuất kết quả: Nối nội dung của các hộp lại với nhau. Thứ tự nối các hộp được quyết định bằng cách sắp xếp các giá trị trong reducedKeyNums theo thứ tự tăng dần. (Ví dụ: nếu reducedKeyNums = [14, 6, 22] thì hộp ứng với 6 sẽ được đọc trước, rồi đến hộp ứng với 14, cuối cùng là hộp ứng với 22).**

**Chuỗi cuối cùng thu được chính là bản mã.**

**Chi tiết các bước GIẢI MÃ**

**Giải mã là thực hiện ngược lại toàn bộ quá trình mã hóa.**

**A) Hoán vị cột ngược (Đảo ngược bước 5)**

* **Tính toán lại từ khóa:** 
  + **Tính keyNums và reducedKeyNums (với N là độ dài) từ khóa bí mật, y như lúc mã hóa.**
* **Xác định kích thước các "hộp":** 
  + **Bản mã có độ dài L. Hộp thứ j (trong số N hộp, j từ 0 đến N-1) sẽ chứa ceil((L – j) / N) ký tự. (ceil(x) là làm tròn x lên số nguyên gần nhất).**
* **Đổ lại vào các hộp:** 
  + **Đọc các ký tự của bản mã. Đổ đầy các hộp theo thứ tự của reducedKeyNums đã được sắp xếp tăng dần.**
  + **Tức là, nếu reducedKeyNums sau khi sắp xếp là s\_0, s\_1, ..., s\_{N-1}, thì ceil((L-s\_0\_idx)/N) ký tự đầu tiên của bản mã sẽ vào hộp ứng với s\_0, v.v. (trong đó s\_0\_idx là chỉ số *ban đầu* của s\_0 trong reducedKeyNums *chưa sắp xếp*).**
* **Tái tạo chuỗi R:** 
  + **Đọc lại các ký tự từ các hộp theo kiểu round-robin (hộp 0, hộp 1, ..., hộp N-1, rồi lặp lại) để lấy lại chuỗi R ban đầu (trước khi bị hoán vị cột).**

**B) Phân tách R thành các khối 12 ký tự, tái tạo blockM (Đảo ngược bước 4)**

* **Chia chuỗi R (đã khôi phục) thành các khối 12 ký tự (C₀…C₁₁).**
* **Với mỗi khối:** 
  + **Tạo ma trận M (6x6) rỗng.**
  + **Nửa cột trái của M (cột 0,1,2):** 
    - **Với i từ 0 đến 5 (tương ứng 6 ký tự đầu C₀ đến C₅ của khối):** 
      * **Lấy ký tự Cᵢ. Chuyển nó về giá trị số v ('0' → 0, 'a'→1, ..., 'z'→26).**
      * **Chuyển v thành 3 trit: q₀ = v // 9, q₁ = (v % 9) // 3, q₂ = v % 3.**
      * **Điền vào M: M[i,0] = q₀, M[i,1] = q₁, M[i,2] = q₂.**
  + **Nửa cột phải của M (cột 3,4,5):** 
    - **Với i từ 6 đến 11 (tương ứng 6 ký tự sau C₆ đến C₁₁ của khối):** 
      * **Lấy ký tự Cᵢ. Chuyển nó về giá trị số v.**
      * **Chuyển v thành 3 trit.**
      * **Điền vào M: M[i-6, 3] = q₀, M[i-6, 4] = q₁, M[i-6, 5] = q₂. (Lưu ý i-6 để chỉ số hàng vẫn là 0-5).**

**C) Đảo ngược quá trình xáo trộn (Đảo ngược bước 3)**

* **Lặp qua từng ký tự của khóa, nhưng theo thứ tự ngược lại (từ cuối về đầu).**
* **Với mỗi ký tự khóa:** 
  1. **Tính keyNum, permuteIndex, addIndex như lúc mã hóa.**
  2. **Áp dụng inverse\_add(M, addIndex): Thực hiện phép trừ (mod 3) tương ứng để đảo ngược phép cộng.** 
     + **addIndex = 0: Trừ 1 (mod 3) từ mọi ô M[i,j] nếu (i + j) chẵn.**
     + **addIndex = 1: Khối 3x3 dưới-phải -= khối 3x3 trên-trái (mod 3).**
     + **addIndex = 2: Khối 3x3 trên-trái -= khối 3x3 dưới-phải (mod 3).**
     + **addIndex = 3: Khối 3x3 dưới-trái -= khối 3x3 trên-phải (mod 3).**
     + **addIndex = 4: Khối 3x3 trên-phải -= khối 3x3 dưới-trái (mod 3).**
  3. **Áp dụng inverse\_permute(M, permuteIndex): Áp dụng phép hoán vị ngược của phép hoán vị permuteIndex đã dùng lúc mã hóa. (Mỗi phép hoán vị A,B,C,D,E sẽ có một phép hoán vị ngược tương ứng).**

**D) Đọc lại 12 chữ cái ban đầu từ blockM (Đảo ngược bước 2)**

* **Sau khi blockM (hay M) đã được "un-scramble":** 
  + **6 ký tự đầu tiên (L₀…L₅):** 
    - **Với mỗi cột i (từ 0 đến 5):** 
      * **Lấy 3 trit từ cột i của nửa trên: M[0,i], M[1,i], M[2,i].**
      * **Tính giá trị số: num = 9 \* M[0,i] + 3 \* M[1,i] + 1 \* M[2,i].**
      * **Chuyển num thành chữ cái (chr(num + 96)). Đây là Lᵢ.**
  + **6 ký tự tiếp theo (L₆…L₁₁):** 
    - **Với mỗi cột i (từ 0 đến 5):** 
      * **Lấy 3 trit từ cột i của nửa dưới: M[3,i], M[4,i], M[5,i].**
      * **Tính giá trị số: num = 9 \* M[3,i] + 3 \* M[4,i] + 1 \* M[5,i].**
      * **Chuyển num thành chữ cái. Đây là L₆₊ᵢ.**
* **Kết quả là 12 chữ cái của khối bản rõ ban đầu.**

**Cuối cùng (Đảo ngược bước 1)**

* **Ghép tất cả các khối 12 chữ cái đã giải mã lại.**
* **Loại bỏ các ký tự 'x' đệm ở cuối để thu được thông điệp gốc.**