# Chuyển đổi từ PBC sang RELIC trong libbswabe

## 1. Thay đổi về thư viện và môi trường biên dịch

Trước (PBC): Sử dụng thư viện PBC để thực hiện các phép toán trên nhóm đường cong elliptic và pairing.  
Sau (RELIC): Thay bằng thư viện RELIC, hỗ trợ nhiều loại đường cong và tối ưu hóa hiệu suất tốt hơn.  
Ý nghĩa: Cải thiện hiệu suất, giảm độ phức tạp, tăng tính linh hoạt.

## 2. Thay đổi kiểu dữ liệu

Trước (PBC): Sử dụng element\_t cho mọi nhóm.  
Sau (RELIC): Sử dụng g1\_t, g2\_t, gt\_t, bn\_t, giúp tối ưu hiệu suất và giảm lỗi runtime.  
Ý nghĩa: Phát hiện lỗi sớm hơn, tối ưu bảo mật và hiệu suất.

## 3. Thay đổi cách sinh số ngẫu nhiên

Trước (PBC): element\_random(beta);  
Sau (RELIC): bn\_rand\_mod(beta, order);  
Ý nghĩa: Kiểm soát tốt hơn phạm vi giá trị, tăng cường bảo mật.

## 4. Thay đổi phép toán trên nhóm G1, G2, GT

Trước (PBC): element\_mul(h, g, beta);  
Sau (RELIC): g1\_mul(h, g, beta);  
Ý nghĩa: Hiệu suất tốt hơn, giảm nguy cơ lỗi.

## 5. Thay đổi phép toán pairing

Trước (PBC): element\_pairing(g\_hat\_alpha, g, g\_alpha);  
Sau (RELIC): pc\_map(g\_hat\_alpha, g, g\_alpha);  
Ý nghĩa: Nhanh hơn, hỗ trợ nhiều loại đường cong hơn.

## 6. Thay đổi ánh xạ chuỗi thành phần tử nhóm

Trước (PBC): element\_from\_hash(h, str, strlen(str));  
Sau (RELIC): g1\_map(h, (uint8\_t\*)str, strlen(str));  
Ý nghĩa: Chính xác hơn, hiệu suất cao hơn.

## 7. Thay đổi cách lưu và đọc dữ liệu

Trước (PBC): element\_to\_bytes(buf, g);  
Sau (RELIC): g1\_write\_bin(buf, size, g, 1);  
Ý nghĩa: Định dạng dữ liệu rõ ràng, tối ưu bộ nhớ.

## 8. Thay đổi quản lý bộ nhớ

Trước (PBC): element\_clear(g);  
Sau (RELIC): g1\_free(g);  
Ý nghĩa: Tiết kiệm bộ nhớ, tránh rò rỉ bộ nhớ.

## Kết luận

Việc chuyển đổi từ PBC sang RELIC giúp tăng tốc, an toàn hơn, linh hoạt hơn, và dễ bảo trì hơn.