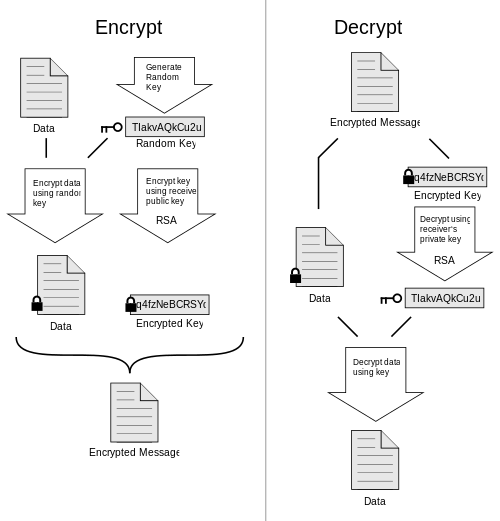
**Demo PGP (Pretty Good Privacy)**

1. Những thành phần bên trong thuật toán PGP
2. Public Key: Khóa công khai, dùng để trao đổi giữa các thành phần. AI cũng có thể nhìn thấy được public key. Dùng để mã hóa dữ liệu trước khi gửi về server.
3. Private Key: Khóa bí mật, dùng để tạo ra public key.
4. PassPhare: kết hợp với Username và các thành phần khác để tạo ra private key và public key. Ngoài ra, còn được dùng trong việc giải mã.
5. Secret Key (aka random Key): Được tạo ra từ PassPhare => Dùng để giải mã dữ liệu.
6. Sơ đồ hoạt động:
7. Sơ đồ ban đầu: truyền 3 tham số để mã hóa dữ liệu: PublicKey, PrivateKey, PassPhare. 
8. Sơ đồ cải tiến: Client gọi API để lấy public key => Dùng public key để mã hóa dữ liệu => chuyển dữ liệu về phía server => Server dùng private key + PassPhare để giải mã dữ liệu.
9. Ví trí Key sau khi được tạo ra:

Vị trí: C:\Users\Thanh\AppData\Local\Temp\Temporary ASP.NET (hoặc tương tự)

* Search PGPPrivateKey.asc và PGPPublicKey.asc
* Được tạo mới mỗi khi run project server.

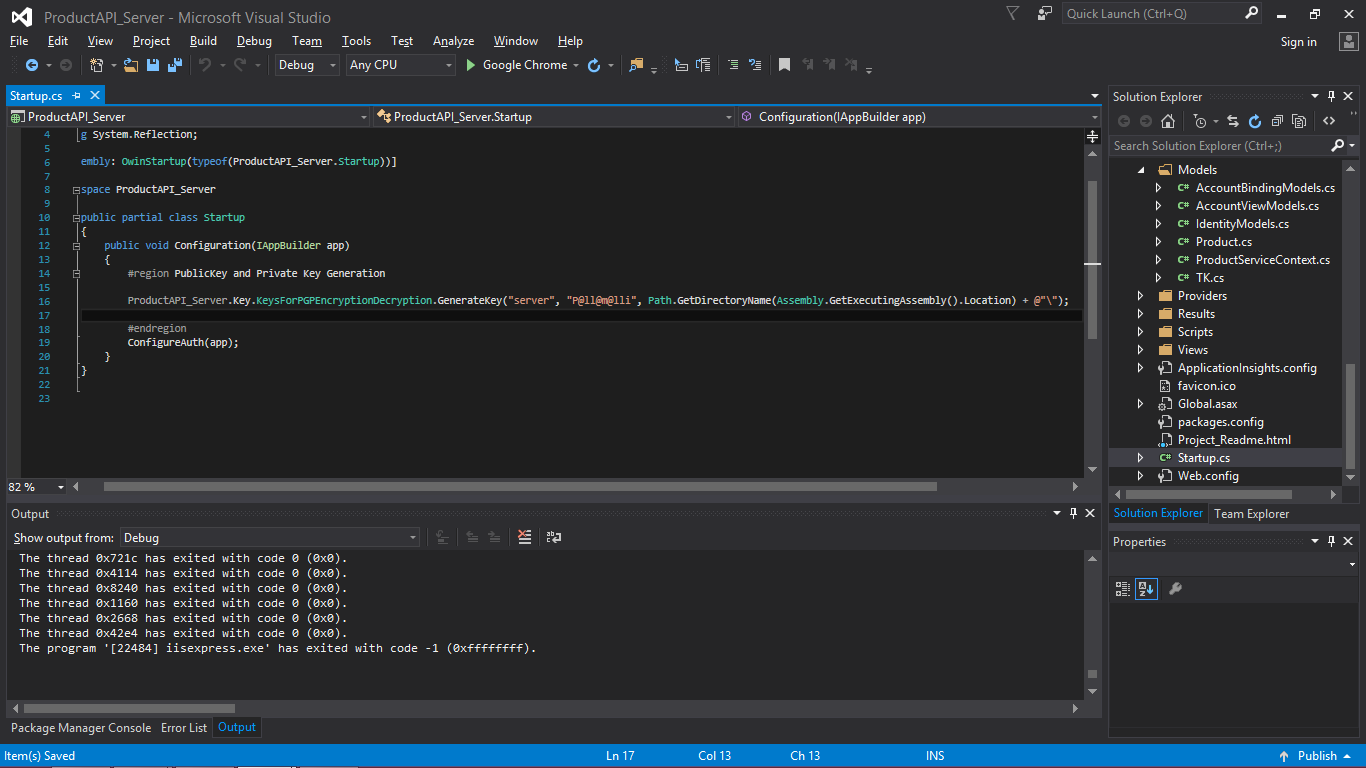
1. Mô tả demo.
2. Bao gồm 3 folder:

* ProductAPI\_Server: web project cung cấp các API để thực thi chương trình
* ProductAPI\_Client: Console app dùng để test các API
* Pretty Good Alg: => Old: PGP Project nguyên mẫu được lấy từ MSDN

=> New/Late: PGP Project đã qua chỉnh sửa + comment các tính năng. (chỉnh từ sơ đồ nguyên mẫu sang sơ đồ cải tiến)

1. Server: Khởi tạo key
2. Khởi tạo key:

* Được gọi ở file Startup.cs, khởi tạo key trước khi bắt đầu chạy app.

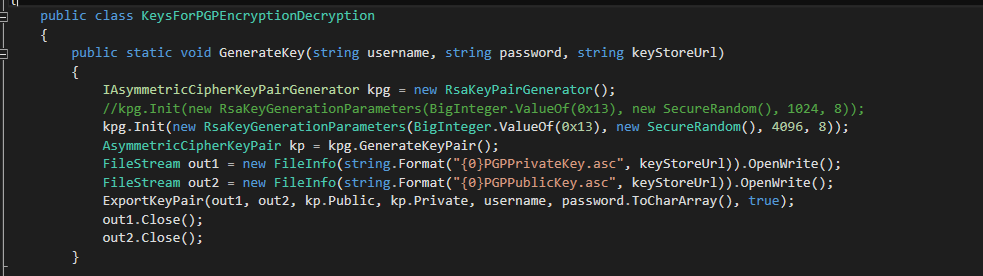


Call command: ProductAPI\_Server.Key.KeysForPGPEncryptionDecryption.GenerateKey("server", "P@ll@m@lli", Path.GetDirectoryName(Assembly.GetExecutingAssembly().Location) + @"\");

1. GenerateKey

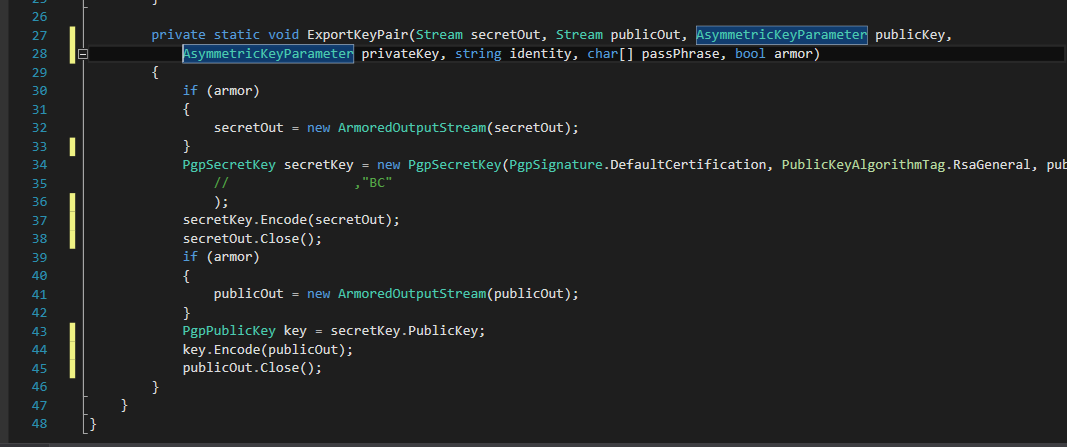
Các tham số được truyền vào: username, password, đường dẫn lưu file.

* Dùng tạo private key và public key dựa vào username và password.
* Gọi hàm lưu chúng vào file.



Lưu ý: số 4096 là độ dài của chuỗi trong file private key. Nên truyền vào tham số la bội số của 1024. Và độ dài của private key cũng ảnh hưởng đến độ dài file của public key.

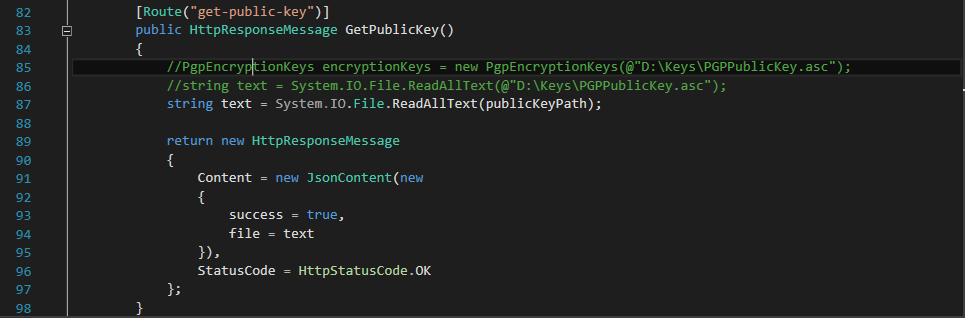
1. ExportKeyPair



* Được gọi từ GenerateKey
* Truyền về các tham số: stream của nơi save private key và public key, publickey và private key sau khi được tạo ra ở GenerateKey, username, password, và tham số phụ (armor?).
* Hàm này dùng để in private key và public key vào file nơi lưu trữ chúng.

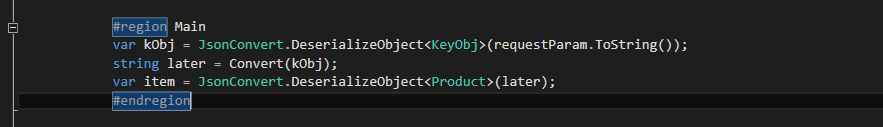
1. Server: API nhận public key

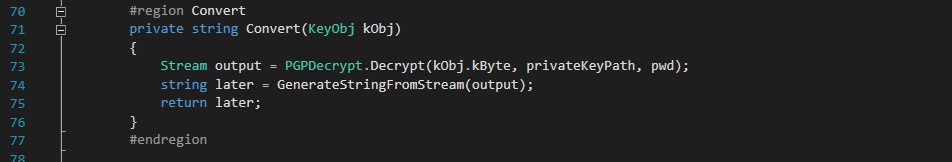
* Đường dẫn: api/acc/get-public-key
* (Năm ở file TksController.cs)



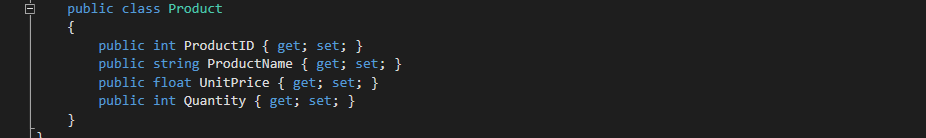
* Trả về chuổi string là thông tin của public key.
* Có thể thêm [Authorized] vào đầu hàm để chỉ có thể gọi được hàm này sau khi đã Authorized.

1. Server: Nhận dữ liệu sau khi mã hóa và giải mã (Nẳm ở file ProductController.cs



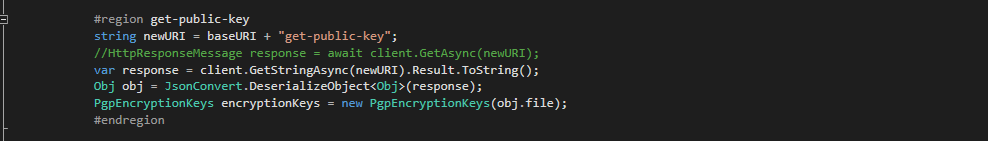




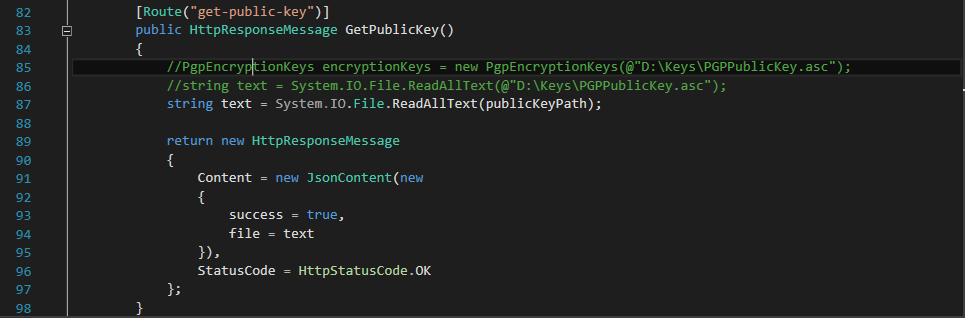


* Dữ liệu được chuyển từ client sang server dưới dạng 1 Json object KeyObj với giá trị byte[] được đặt trong parameter kByte.
* Sau đó phần dât được giải mã sẽ được lưu trữ trong OutputStream output khi đưa đúng private key ứng với public key đã mã hóa và password tạo ra chính private key và public key đó.
* Chuyển từ outputStream sang chuỗi Json để có thể chuyển nó về object ban đầu (quá trình mã hóa: object ban đầu => chuổi json => mã hóa chuổi json đó => Tạo chuổi json mới { kByte: <chuỗi json sau khi mã hóa = chuỗi string mới>} )

1. Client: get public key and encrypt data:

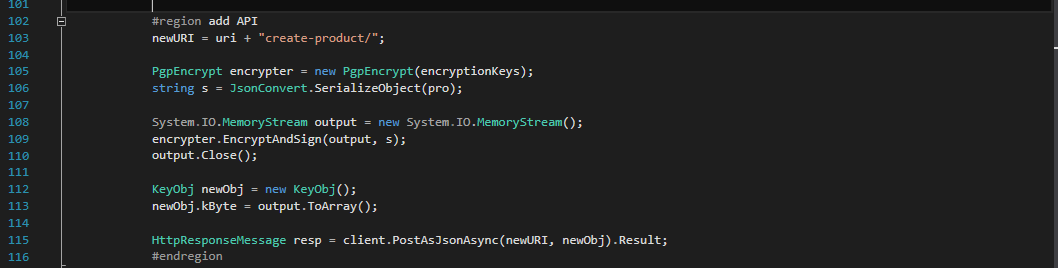


Gọi API để lấy public key dưới dạng một json.





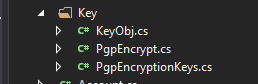
(dùng chuổi string ở trong parameter file đó để tạo public key)





* Chuyển object sang chuỗi json => mã hóa chuỗi json đó => Bỏ vào một object mới => gửi đi

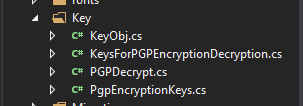
**Lưu ý:**



Object KeyObj: Chứa data trước khi giải mã dưới dạng byte[]

PgpEncrypt: mã hóa dữ liệu

PgpEncryptionKeys: Tạo public từ chuỗi string nhận được từ việc gọi api.



KeysForPGPEncryptionDecryption: Tạo 2 key.

Object KeyObj: Chứa data trước khi giải mã dưới dạng byte[]

PgpEncrypt: mã hóa dữ liệu

PgpEncryptionKeys: Tạo public từ chuỗi string nhận được từ việc gọi api.

**Các thành phần phụ cần lưu ý:**

