**(( گزارش پروژه دوم مبانی امنیت اطلاعات ))**

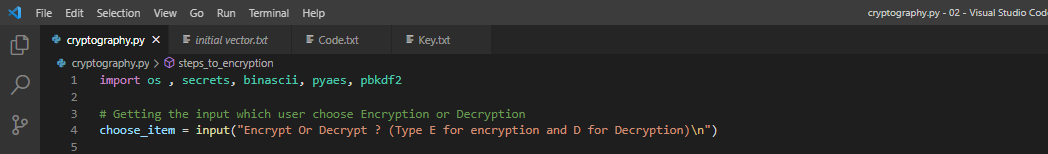
**گردآورنده : عرفان ماجدی 9831099**

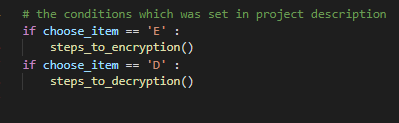
**مبانی امنیت اطلاعات**

**دکتر شهریاری**

**توضیحات کد پروژه )**

**در ابتدا باید کتابخانه های موردنیاز را import کنیم و سپس با توجه به دستور کار پروژه باید یک ورودی از کاربر بگیریم که اگر E وارد کرد همان encryption است یعنی میخواهد عملیات رمزنگاری را انجام دهد و اگر D وارد کرد همان Decryption است به این معنا که میخواهد عملیات رمزگشایی را انجام دهد. حال برای این دو عملیات دو تابع جداگانه تعریف شده است که براساس ورودی کاربر هر کدام از آنها وظایف خود را انجام می دهند و همچنین یک تابع برای ایجاد فایل ها درست کردیم . در ادامه به توضیح این توابع خواهیم پرداخت ولی در ابتدا کد ورودی و شرط رفتن به توابع را باهم می بینیم :**

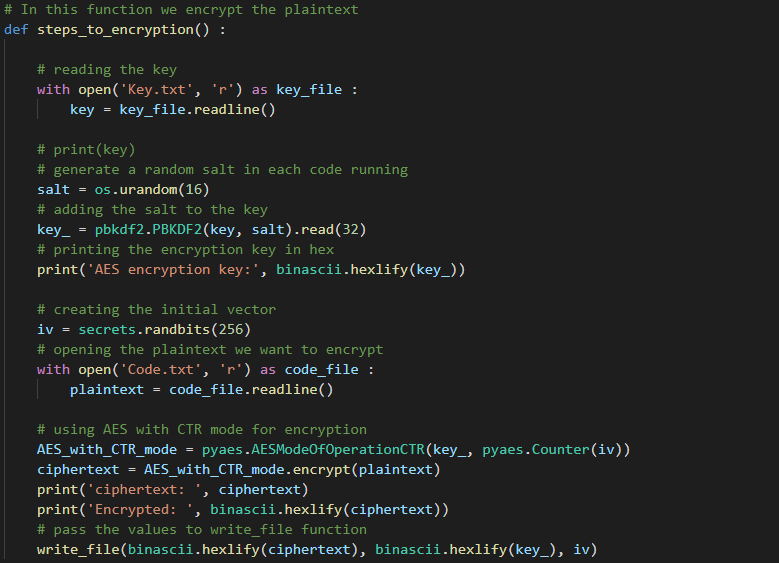
****

****

**همان طور که از کد هم پیداست اگر کاربر E وارد کرد وارد تابع steps\_to\_encryption می شویم و اگر D را به عنوان ورودی وارد کرد وارد تابع steps\_to\_decryption خواهیم شد. حال می خواهیم در ادامه با نحوه کارکرد هرکدام آشنا شویم .**

**تابع steps\_to\_encryption )**

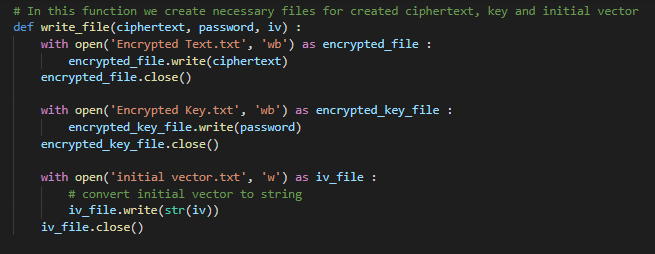
**تابع اولی که میخواهیم درباره ی آن صحبت کنیم در واقع به صورت کلی کاری که میکند این است که plaintext را به ciphertext با استفاده از یک key تبدیل می کند. حال کد این تابع را باهم می بینیم :**

****

**در ابتدای این تابع فایل Key را میخوانیم که محتوای آن کلیدی است که در دستور پروژه قرار داده شده است و میخواهیم از آن در رمزنگاری و رمزگشایی استفاده کنیم. سپس با استفاده از متد urandom در کتابخانه ی OS یک salt به اندازه 16 بایت ایجاد می کنیم و با استفاده از کتابخانه ی pbkdf2 این salt را به key اضافه می کنیم و سپس 32 بایت آن که همان 256 بیت است را می خوانیم و در اخر این کلید جدید که به صورت بایت است را با استفاده از متد hexlify از کتابخانه ی binascii به hex تبدیل می کنیم و به کاربر نشان می دهیم. در مرحله ی بعد initial vector را با استفاده از متد randbits از کتابخانه ی secrets به اندازه 256 بیت تولید می کنیم . حال وقت آن است که فایلی که میخواهیم رمزنگاری کنیم را باز کنیم و آن را بخوانیم و در یک متغیر به نام plaintext ذخیره کنیم. در مرحله ی اخر باید با استفاده از کتابخانه ی pyaes.AESModeOfOperationCTR که شی این کلاس است به آن key\_ و یک شمارنده براساس initial vector بدهیم و سپس آن را در یک متغیر ذخیره می کنیم . حال plaintext را با استفاده از متد encrypt رمزنگاری می کنیم و خود ciphertext و حالت hex آن را به کاربر نمایش می دهیم. در آخرین مرحله چون میخواهیم برای key\_ ، ciphertext و iv فایل های جداگانه بسازیم یک تابع write\_file ایجاد می کنیم و این سه متغیر را به آنها می دهیم. در ادامه با این تابع نیز آشنا می شویم .**

**تابع make\_file )**

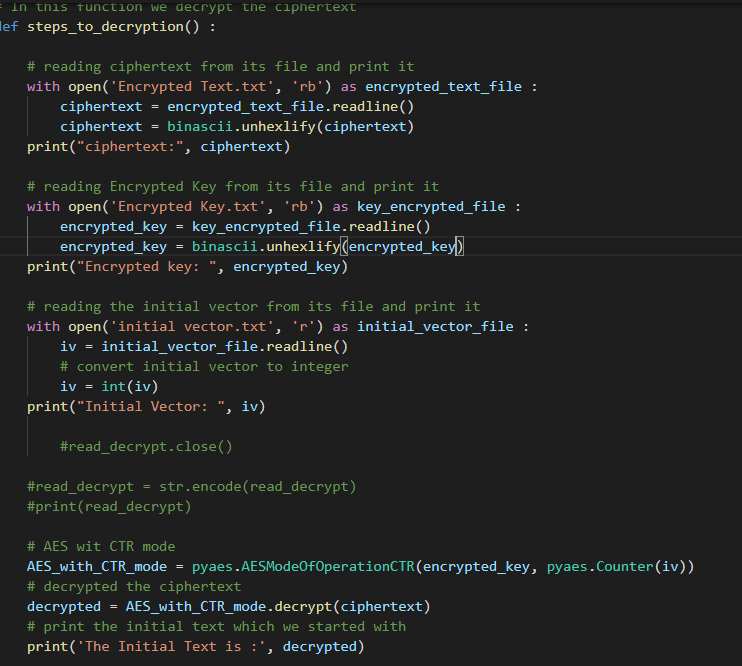
**تابعی که میخواهیم درباره ی آن صحبت کنیم وظیفه اش ایجاد فایل است که در ادامه کد آن را می بینیم :**

****

**این تابع سه ورودی دارد که به ترتیب ciphertext, password, iv است. کاری که انجام می دهیم این است که برای هر ورودی یک فایل ایجاد کرده و در آن فایل این ورودی ها را ذخیره می کنیم و سپس آن را می بندیم . به عنوان مثال برای ورودی iv یک فایل txt به نام initial vector درست کردیم و iv را به صورت string در آن ذخیره کردیم زیرا iv یک عدد صحیح است و سپس فایل تولید شده را بستیم . در اخر به سراغ تابع steps\_to\_decryption می رویم.**

**تابع steps\_to\_decryption )**

**در این تابع کاری که به صورت کلی انجام می دهیم این است که ciphertext را به plaintext ای که در اول برنامه داشتیم تبدیل کنیم. کد این تابع به صورت زیر است :**

****

**در این تابع کاری که در ابتدای امر می کنیم این است که هر سه فایلی که در تابع قبل درست کردیم را باز می کنیم ، می خوانیم و در متغیری ذخیره می کنیم و در آخر هم آن ها را برای اطمینان به کاربر نشان می دهیم . فقط توجه کنید که چون initial vector باید به صورت عدد صحیح استفاده شود ، مقداری که از فایل می خوانیم را به عدد صحیح تبدیل کردیم. سپس مانند قسمت encryption یک AES با mode کاری CTR درست می کنیم ولی در اینجا به جای متد encrypt از متد decrypt استفاده می کنیم و ciphertext را به آن می دهیم و plaintext ایجاد شده را به کاربر نمایش می دهیم. در این جا کار ما در پروژه به اتمام می رسد :)**