

مشروع برمجة وإدارة الشبكات

التشفير من طرف إلى طرف والتشفير من نقطة إلى نقطة

تقديم الطلاب

هنا رياض بوظ

محمد نور مصطفى قاقو

إشراف الدكتور

مهند عيسى

التشفير من طرف إلى طرف والتشفير من نقطة إلى نقطة

ملخص:

يعني التشفير في الأمن الإلكتروني تحويل البيانات من تنسيق قابل للقراءة إلى تنسيق مشفر. لا يمكن قراءة البيانات المشفرة أو معالجتها إلا بعد فك تشفيرها.

ويعد التشفير وحدة البناء الأساسية لأمن البيانات. وهو أبسط الطرق وأهمها لضمان عدم سرقة معلومات نظام الحاسوب أو قراءتها من جانب شخص يريد استخدامها لأغراض ضارة.

يستخدم تشفير البيانات لتأمينها على نطاق واسع من قبل المستخدمين الأفراد والشركات الكبيرة بغرض حماية معلومات المستخدم المرسلة بين المستعرض والخادم. قد تشمل تلك المعلومات أي شيء من بيانات الدفع إلى المعلومات الشخصية. ويتم استخدام برنامج تشفير البيانات، المعروف أيضًا باسم "خوارزمية التشفير" أو "التشفير" فحسب، لتطوير مخطط تشفير لا يمكن اختراقه نظريًا إلا بقوة حوسبة هائلة.

على الرغم من أن حلول التشفير من نقطة إلى نقطة والتشفير من طرف إلى طرف (E2EE) متشابهة، إلا أن هناك فرقًا رئيسيًا. أي أن حلول E2EE لا تفي بمعايير مجلس PCI، ويرجع ذلك في الغالب إلى وجود أنظمة أخرى بين نقطة الاهتمام ونقطة المعالجة، مما يزيد من فرص الاختراق أو الاختراق. في المقابل، تنقل حلول P2PE البيانات مباشرة، مع عدم وجود أنظمة أخرى بينهما. تجدر الإشارة أيضًا إلى أن P2PE قابل للتقييم، في حين لا توجد أي معايير مرتبطة بحلول التشفير التي تحمل علامة تجارية على أنها من طرف إلى طرف.

كلمات مفتاحية: E2EE , P2PE

1. مقدمة

عند مشاركة المعلومات أو البيانات عبر الإنترنت، فإنها تمر عبر سلسلة من أجهزة الشبكات في جميع أنحاء العالم، والتي تشكل جزءًا من الإنترنت في مجمله. ومن خلال انتقال البيانات عبر الإنترنت العام، يبرز احتمال أن يتم اختراقها أو سرقتها من قبل المتسللين. وللحيلولة دون ذلك، يمكن للمستخدمين تثبيت برامج أو أجهزة معينة لضمان النقل الآمن للبيانات أو المعلومات. تُعرف هذه العمليات في حقل أمان الشبكات باسم "التشفير"[1].

يتضمن التشفير تحويل النص العادي المقروء بالنسبة للإنسان إلى نص غير مفهوم، وهو ما يعرف بالنص المشفر. يعني هذا بشكل عام أخذ بيانات قابلة للقراءة وتغييرها بحيث تظهر بشكل عشوائي. ويتضمن التشفير استخدام مفتاح تشفير، وهو مجموعة من القيم الرياضية يتفق عليها كل من المرسل والمتلقي. يستخدم المستلم المفتاح لفك تشفير البيانات وإعادتها إلى هيئة نص عادي يمكن قراءته.

وكما كان مفتاح التشفير أكثر تعقيدًا، كان التشفير أكثر أمانًا، حيث نقل احتمالية قدرة الجهات الخارجية على فك تشفيره من خلال هجمات القوة الغاشمة (أي محاولة الأرقام العشوائية إلى أن يتم تخمين المجموعة الصحيحة). يستخدم التشفير أيضًا لحماية كلمات المرور. تعمل طرق تشفير كلمة المرور على بعثرة كلمة المرور الخاصة بك لتصبح غير قابلة للقراءة من قبل المتسللين.

2. أهمية البحث وأهدافه:

على الرغم من أن حلول التشفير من نقطة إلى نقطة والتشفير من طرف إلى طرف (E2EE) متشابهة، إلا أن هناك فرقًا رئيسيًا. أي أن حلول E2EE لا تفي بمعايير مجلس PCI، ويرجع ذلك في الغالب إلى وجود أنظمة أخرى بين نقطة المصدر ونقطة الهدف، مما يزيد من فرص الاختراق. في المقابل، تنقل حلول P2PE البيانات مباشرة، مع عدم وجود أنظمة أخرى بينهما. تجدر الإشارة أيضًا إلى أن P2PE قابل للتقييم، في حين لا توجد أي معايير مرتبطة بحلول التشفير التي تحمل علامة تجارية على أنها من طرف إلى طرف.

3. تقنيات التشفير:

توجد طريقتان للتشفير [2] هما الأكثر شيوعًا: التشفير المتماثل وغير المتماثل. يشير الاسمان إلى ما إذا كان يتم استخدام المفتاح نفسه للتشفير ثم لفك التشفير أم لا:

مفاتيح التشفير المتماثلة: يُعرف هذا أيضًا باسم تشفير المفتاح الخاص. يكون المفتاح المستخدم للتشفير هو نفسه المستخدم لفك التشفير، مما يجعل هذا الأسلوب الأفضل للمستخدمين الفرديين والأنظمة المغلقة. وبخلاف ذلك، يجب إرسال المفتاح إلى المتلقي. على أن هذا يزيد من خطر التعرض للاختراق إذا اعترضته جهة خارجية، مثل المتسللين. لكن هذه الطريقة أسرع من الطريقة غير المتماثلة.

التشفير غير المتماثل: يستخدم هذا الأسلوب مفتاحين مختلفين، أحدهما عام والآخر خاص، مرتبطين معًا حسابيًا. والمفتاحان هما في الأساس أرقام كبيرة، وتم ربطهما معًا لكنهما ليسا متماثلين، ومن هنا جاءت التسمية "غير متماثل". يحتفظ المالك بالمفتاح الخاص سرًا، ويتم إما مشاركة المفتاح العام بين المستلمين المصرح لهم أو إتاحتها للجمهور بشكل عام.

ولا يمكن فك تشفير البيانات المشفرة باستخدام المفتاح العام للمستلم إلا باستخدام المفتاح الخاص المقابل.

أمثلة على خوارزميات التشفير

تُستخدم خوارزميات التشفير لتحويل البيانات إلى نص مشفر. تستخدم الخوارزمية مفتاح التشفير لتغيير البيانات بطريقة يمكن التنبؤ بها بحيث يمكن إعادتها إلى نص عادي باستخدام مفتاح فك التشفير، على الرغم من أن البيانات المشفرة ستظهر بشكل عشوائي.

هناك عدة أنواع مختلفة من خوارزميات التشفير المصممة لتناسب مختلف الأغراض. كما يتم تطوير خوارزميات جديدة عندما تصبح الخوارزميات القديمة غير آمنة. تتضمن بعض خوارزميات التشفير الأكثر شهرة ما يلي:

التشفير وفق معيار تشفير البيانات

يشار لمصطلح "معيّار تشفير البيانات" بالحروف DES. وهو عبارة عن خوارزمية تشفير متماثل عفا عليها الزمن ولا تعتبر مناسبة لاستخدامات اليوم. لذلك، برزت خوارزميات التشفير الأخرى الجديدة من نوع DES.

التشفير وفق المعيار الثلاثي لتشفير البيانات

يشار لمصطلح "المعيار الثلاثي لتشفير البيانات" بالحروف DES. وهذه عبارة عن خوارزمية مفتاح متماثل، ويتم استخدام كلمة "ثلاثي" لأن البيانات يتم تمريرها عبر خوارزمية DES الأصلية ثلاث مرات أثناء عملية التشفير. يتم حالياً الانصراف تدريجياً عن استخدام المعيار الثلاثي لتشفير البيانات، ولكن لا يزال بإمكانه إنشاء حل تشفير يمكن الاعتماد عليه للخدمات المالية والصناعات الأخرى.

التشفير وفق معيار التشفير المتقدم

يشار إلى مصطلح "معيّار التشفير المتقدم" بالحروف AES ، وقد تم تطويره لتحديث خوارزمية معيار تشفير البيانات الأصلية. تتضمن بعض التطبيقات الأكثر شيوعاً لخوارزمية AES تطبيقات المراسلة مثل Signal أو WhatsApp وبرنامج أرشفة وضغط الملفات WinZip.

التشفير بخوارزمية ريفست وشامير وأديلمان

كانت خوارزمية ريفست وشامير وأديلمان أول خوارزمية تشفير غير متماثل تتاح على نطاق واسع للجمهور. وهي واسعة الشعبية نظراً لطول مفتاحها، مما يؤهلها للاستخدام على نطاق واسع لنقل البيانات بشكل آمن. يشار إلى هذه الخوارزمية بالحروف RSA ، وهي الحروف الأولى من لقب عائلة كل من علماء الرياضيات الذين وصفوها لأول مرة: رونالد ريفست وآدي شامير وليونارد أديلمان. تعتبر RSA خوارزمية غير متماثلة بسبب استخدامها لزوج من المفاتيح.

تشفير Twofish

تُستخدم خوارزمية Twofish في كل من الأجهزة والبرامج، وتعتبر واحدة من الأسرع في نوعها. و Twofish غير مسجلة ببراءة اختراع، مما يجعل التشفير بها متاحًا مجانيًا لأي شخص يريد استخدامها. نتيجة لذلك، ستجدها مضمّنة في برامج التشفير مثل PhotoEncrypt و GPG والبرنامج الشهير مفتوح المصدر TrueCrypt.

التشفير وفق خوارزمية RC4

يُستخدم في WEP و WPA، وهما بروتوكولان تشفير شائعًا الاستخدام في أجهزة التوجيه اللاسلكية. تتضمن أمثلة التشفير غير المتماثل RSA و DSA. وتتضمن أمثلة التشفير المتماثل RC4 و DES. وبالإضافة إلى خوارزميات التشفير، هناك أيضًا ما يُعرف بالمعايير المشتركة: (CC)

وهذا ليس معيار تشفير، بل مجموعة من الإرشادات الدولية للتحقق من ادعاءات أمان المنتج ومدى رسوخها أمام التدقيق. تم إنشاء إرشادات CC لتوفير إشراف محايد من قبل البائعين والجهة الخارجية على منتجات الأمان. يتم تقديم المنتجات للمراجعة طوعية من قبل البائعين، ويتم فحص وظائفها الكاملة أو وظائف معينة بها. عندما يتم تقييم منتج ما، يتم اختبار ميزاته وفقًا لمجموعة محددة من المعايير حسب نوع المنتج. وفي البداية، كان التشفير خارج نطاق المعايير المشتركة. ولكن يتم تضمينه بشكل متزايد في معايير الأمان الخاصة به.

4. التشفير من طرف إلى طرف والتشفير من نقطة إلى نقطة:

يضمن التشفير من طرف إلى طرف أو من النهاية إلى النهاية. أن يتم تشفير البيانات والحفاظ عليها سرية حتى تصل إلى المستلم المقصود. سواء كنت تتحدث عن الرسائل والمكالمات المشفرة من طرف إلى طرف، أو البريد الإلكتروني، أو تخزين الملفات، أو أي شيء آخر يتم تبادله بين طرفين عبر الإنترنت، فإن هذا يضمن عدم تمكن أي شخص أو شركة أو حكومة من اعتراض أو رؤية بياناتك الخاصة.[3]

أولاً، لنبدأ بالتشفير وهو طريقة للحفاظ على سرية البيانات بحيث لا يمكن للعموم قراءتها بشكل مفهوم. يمكن فقط للأشخاص الذين يمكنهم فك تشفير البيانات من رؤيتها محتواها. وإذا لم يمتلك الشخص مفتاحاً لفك تشفير هذه البيانات، فلن يتمكن من فك رموزها وعرض المعلومات التي تحتويها.

تستخدم أجهزتك أشكالاً مختلفة من التشفير طوال الوقت. على سبيل المثال، عند الوصول إلى موقع الويب الخاص بالخدمات المصرفية عبر الإنترنت -أو أي موقع ويب يستخدم بروتوكول HTTPS مثل أراجيك، وهو مستخدم من قبل معظم مواقع الويب هذه الأيام- يتم تشفير الاتصالات بينك وبين موقع الويب هذا بحيث لا يمكن لمشغل الشبكة ومزود خدمة الإنترنت وأي شخص آخر متصل معك بنفس شبكة الواي فاي من رؤية كلمة المرور المصرفية الخاصة بك والتفاصيل المالية والصفحات التي تتصفحها وغيرها.

يستخدم التشفير أيضاً لتأمين بياناتك. تتيج الأجهزة الحديثة مثل أجهزة iPhone و Android و iPad و Mac و Chromebooks وأنظمة Linux (على الرغم من أن مايكروسوفت لديها خاصية BitLocker في ويندوز 10، إلا أنه ليس الشركات المصنعة تضيفها في حواسيبها ضمن رخصة نظام التشغيل المستخدمة) بتخزين بياناتها على أجهزتك المحلية في شكل مشفر. يتم فك تشفيرها بعد تسجيل الدخول باستخدام رقم التعريف الشخصي أو كلمة المرور[4].

أولاً: طريقة عمل التشفير من طرف إلى طرف:

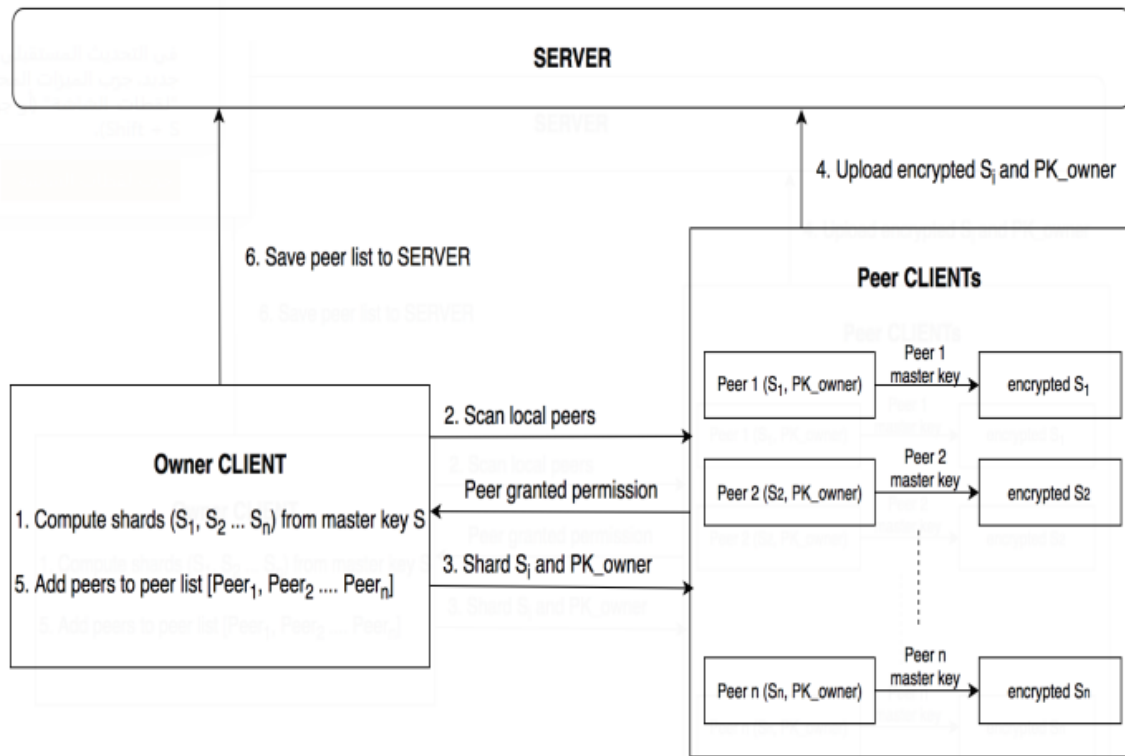
تشفّر بيانات المستخدم من جانب العميل -الجهاز المحمول، وتبقى مشفرة في وسط الانتقال، وتبقى على الخادم، مع منع صالحيات الوصول للأشخاص غير المصرّح لهم؛ إضافة إلى تقديم نظام تشفير المحتوى -الملف- الذي يحقق الخصوصية والتحكم في الوصول الدقيق، ويمكن دمجها بسلاسة مع خدمات استضافة المحتوى الرئيسية التي يعتمد عليها المستخدمون اليوم. في مؤسسة قائمة على بنية المفتاح العام - PKI - يتمتع مسؤولو تكنولوجيا المعلومات فقط بأعلى الامتيازات في أنظمة الكمبيوتر الخاصة بالشركة.

ولكن أيضاً يمتلكون ويديرون أنظمة الهوية مثل Directory Active ، ويمنحهم هذا القدرة على كسر أمان بيانات الموظفين.

يحاول E2EE تحقيق الخصوصية بدون هوية؛ مما يعني أن المهاجمين لا يعرفون من يصل إلى الملف المستهدف بما في ذلك المالك نفسه؛ إذ تُشفّر جميع كتل بيانات المستخدم بواسطة مفاتيح متناظرة مشتركة بين المالك والمستخدمين الآخرين، وجميع كتل بيانات المستخدم لها الطول -استدعاء المفتاح الوهمي- نفسه، ولذلك لا يمكن للمهاجمين إنشاء ترويسة الملف الاستنباط أية معلومات ترتبط بالمستخدمين المصرح لهم وأذونات الوصول الخاصة بهم لملف الهدف.

عملية التشفير (النموذجية) والمبسطة (لحل E2EE هي كما يأتي :

- 1- ينشأ زوج مفاتيح خاص وعام على جهاز المستخدم .
 - 2- يعتمد المستخدم على كلمة المرور الخاصة به لتشفير المفتاح الخاص عن طريق خوارزمية توليد مفتاح تعتمد على كلمة المرور .
 - 3- يرسل المفتاح الخاص المشفر والمفتاح العام الأصلي إلى خادم مفتاح بعيد .
 - 4- عندما يشفر تطبيق محتوى على الجهاز ملفاً؛ فإنه يتحقق مما إذا كان المفتاح الخاص متاحاً محلياً، وإذا لم يكن كذلك ينزل تطبيق المحتوى المفتاح الخاص المشفر ويطلب من المستخدم كتابة كلمة المرور لفك تشفير المفتاح الخاص.
 - 5- ينشأ مفتاح ملف- مفتاح متناظر key-Symmetric -ويستخدم لتشفير الملف، ويُشفّر مفتاح الملف باستخدام المفتاح الخاص .
 - 6- يرسل الملف المشفر ومفتاح الملف إلى خادم استضافة الملفات.
- لمشاركة الملف مع مستخدم آخر؛ يحدد تطبيق المحتوى موقع المفتاح العام للمستلم من خادم المفاتيح، ويستخدمه لتشفير مفتاح الملف، الذي يُرسل إلى خدمة الاستضافة مع الملف المشفر ومفتاح الملف؛ إذ يمكن للمستلم استخدام مفتاحه الخاص لفك تشفير الملف/ المفتاح عن ذلك؛ يمكن لضمان السلامة والموثوقية إنشاء زوج من مفاتيح والملف بالتسلسل. فضالاً التوقيع واستخدامه لتوقيع الملف المشفر ومفتاح الملفات. ونوضح ما سبق بالشكل.



E2EE والخصوصية:

في العديد من أنظمة المراسلة، بما في ذلك البريد الإلكتروني والعديد من شبكات الدردشة، تمر الرسائل عبر وسطاء ويتم تخزينها بواسطة طرف ثالث، يتم استردادها من قبل المستلم. حتى إذا كانت الرسائل مشفرة، يتم تشفيرها فقط "أثناء النقل"، وبالتالي يمكن الوصول إليها بواسطة مزود الخدمة، بغض النظر عما إذا كان تشفير القرص من جانب الخادم مستخدماً أم لا. يعمل تشفير القرص من جانب الخادم ببساطة على منع المستخدمين غير المصرح لهم من عرض هذه المعلومات. لا يمنع الشركة نفسها من عرض المعلومات، حيث أن لديهم المفتاح ويمكنهم ببساطة فك تشفير هذه البيانات.

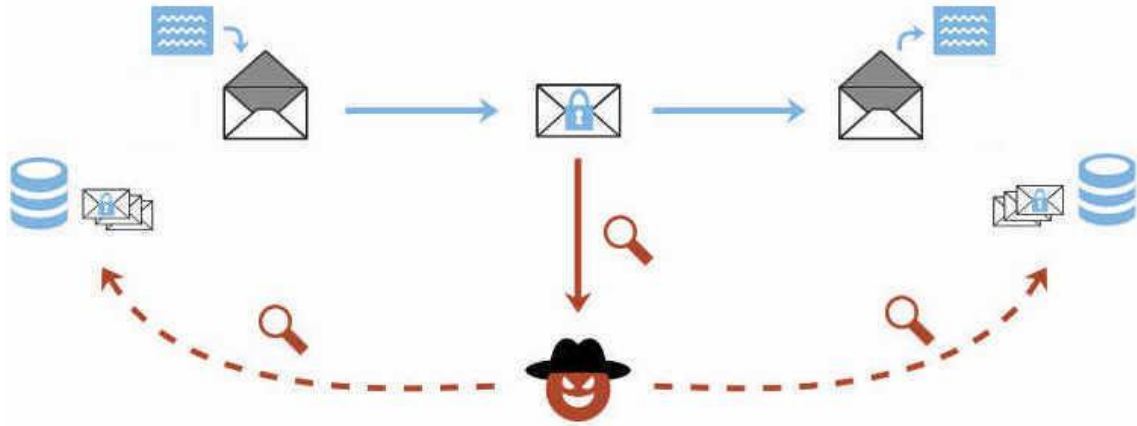
يسمح هذا للطرف الثالث بتوفير البحث والميزات الأخرى، أو البحث عن محتوى غير قانوني وغير مقبول، ولكن يعني أيضًا أنه يمكن قراءتها وإساءة استخدامها من قبل أي شخص لديه حق الوصول إلى الرسائل المخزنة على نظام الطرف الثالث، سواء كان ذلك عن طريق التصميم أو عبر باب خلفي. يمكن اعتبار هذا مصدر قلق في العديد من الحالات التي تكون فيها الخصوصية مهمة للغاية، مثل الشركات التي تعتمد سمعتها على قدرتها على حماية

بيانات الطرف الثالث والمفاوضات والاتصالات التي تعتبر مهمة بما يكفي لخطر "القرصنة" المستهدفة أو المراقبة،
وحيث يتم تضمين مواضيع حساسة مثل الصحة ومعلومات عن القصر

يعمل التشفير من طرف إلى طرف على ترميز الرسائل بطريقة لا يمكن فك تشفيرها إلا من قبل المرسل والمستقبل المقصود. يحدث التشفير من طرف إلى طرف على طرفي الاتصال. يتم تشفير الرسالة على جهاز المرسل، وإرسالها إلى جهاز المستلم بتنسيق غير قابل للقراءة، ثم يتم فك تشفيرها في جهاز المستلم بواسطة مفتاح فك التشفير.

باستخدام التشفير من طرف إلى طرف، لن يتمكن أي طرف موجود في المنتصف -سواء أكان شركة مثل Google أو Facebook أو مزود الانترنت- من رؤية محتويات الرسائل. إنهم لا يملكون مفتاحًا يفتح البيانات الخاصة. فقط المرسل والمستقبل يمسكون بمفتاح الوصول إلى تلك البيانات.

اختراق التشفير من طرف إلى طرف



نعم، يمكن بالفعل كسر التشفير من طرف إلى طرف. لكنه ليس بالأمر السهل والتكنولوجيا المطلوبة للقيام بذلك عادةً ما تكون متوفرة فقط لدى الجهات الحكومية ومجموعات الجريمة المنتظمة المدعومة من دول. لكن هزيمة الدفاعات القوية لا يتطلب سوى فهم نقاط ضعفها: يتمثل الضعف الواضح في التشفير من طرف إلى طرف في أن له نهايتين. غالبًا ما تكون نقاط النهاية (الهاتف الذكي أو الكمبيوتر الشخصي أو أي جهاز آخر) غير آمنة ويمكن اختراقها بطرق

متنوعة لا حصر لها، بالتالي يمكن الاطلاع على محتوى الرسائل قبل وبعد تشفيرها من خلال اختراق أجهزة المستخدمين وتثبيت برمجيات تجسس.

ثانياً: التشفير من نقطة إلى نقطة (P2PE) :

ببساطة، التشفير من نقطة إلى نقطة هو معيار تشفير وضعه مجلس معايير أمن صناعة بطاقات الدفع. وينص على أن معلومات حامل البطاقة يتم تشفيرها مباشرة بعد استخدام البطاقة مع محطة نقاط البيع الخاصة بالتاجر ولا يتم فك تشفيرها حتى تتم معالجتها بواسطة معالج الدفع. يتم استيفاء هذا المعيار من خلال حلول التشفير من نقطة إلى نقطة - وهي خدمات شاملة يقدمها مزودون متخصصون يتألفون من جميع البرامج والأجهزة اللازمة لتلبية متطلبات الامتثال ل P2PE.

طريقة عمل تشفير نقطة إلى نقطة:

يبدأ نظام التشفير نقطة لنقطة بالعمل منذ لحظة وضع البطاقة الائتمانية والتقاطها والمعروفة بنقطة التفاعل، حيث تستخدم خوارزمية تحول البيانات إلى رموز غير قابلة للقراءة، وتنقل هذه الرموز إلى المعالج الذي يفك تشفيرها بطريقة آمنة ثم يتم تمريرها إلى البنك المعني.

وبما أن كامل العملية تتم إلكترونياً فإنه لا يتعين على أي طرف إدخال أو استخلاص المعلومات بشكل يدوي وإنما ترفق كل عملية برقم خاص يساعد التاجر على تصحيح واسترداد وتحديد عملية الدفع في وقت لاحق دون الكشف عن معلومات البطاقة.

إذن تكمن أهمية هذه الطريقة في أنها تنقل البيانات مباشرة من نقطة التفاعل الأولى إلى نقطة المعالجة بشكل مباشر، وبدون وجود أية أنظمة بينهما مما يجعلها أكثر سرعة وأمان

عناصر عملية التشفير

تتطلب عملية التشفير الكثير من الدقة والحذر واستخدام أجهزة خاصة وهي كنظام عام تحوي العناصر التالية:

1. مزود خدمة تشفير نقطة إلى نقطة: وهو معالج للدفع تابع لإحدى الجهات الخارجية أو بوابة خاصة للدفع، وهو يحمي بيانات العملاء ويرفع المسؤولية الأمنية والمخاطر عن التاجر نفسه، حيث يتوجب على التاجر دفع الكثير من الغرامات والعقوبات عند حصول أخطاء في عمليات الدفع.
2. بيئة التشفير: تحوي البيئة القابلة للتشفير وفك التشفير والتهيئة والتصميم وأجهزة لنقطة التفاعل وأية مكونات ضرورية أخرى.
3. نقطة التفاعل: وهي المكان الخاص بالحصول على معلومات البطاقة الائتمانية عند إدخالها وتكون بشكل شريط مغناطيسي أو شريحة وقارئ رموز ويجب أن تكون التطبيقات الموجودة هنا متوافقة مع تشفير نقطة لنقطة.

القسم العملي

مقدمة:

سنقوم ببناء عدة سيناريوهات من تطبيق غرف دردشة جماعية والمؤلف من server يقوم بتقديم الخدمة وclients يقومون بالتحدث معاً على شكل غرف دردشة. بحيث ينضم العميل للغرفة عن طريق اسمها عندها يمكنه استقبال وإرسال رسائل تبث لجميع أعضاء الغرفة.

سيناريوهات العمل:

1- غرف دردشة تتبادل البيانات على شكل نص صريح.

2- غرف دردشة تتبادل البيانات مشفرة بطريقة Point to Point.

3- غرف دردشة تتبادل البيانات مشفرة بطريقة End to End.

السيناريو الأول: غرف دردشة تتبادل البيانات على شكل نص صريح.

في هذا السيناريو يتم إرسال البيانات دون أي عملية تشفير. ويتم استخدام multi-threading في تطبيق السيرفر وذلك لإمكانية خدمة أكثر من عميل بنفس الوقت وتطبيق العميل وذلك لإمكانية استقبال الرسائل أثناء كتابة رسالة من قبل المستخدم.

تطبيق السيرفر:

```
from socket import *
from threading import *

rooms = {}

def send_message(message, room):
    if room in rooms:
        for client in rooms[room]['clients_sockets']:
            client.send(message)
```

```

def worker(client, room):
    while True:
        try:
            message = client.recv(1024).decode()
            print(message)
            send_message(message.encode(), room)
        except:
            # Removing The user
            i = rooms[room]['clients_sockets'].index(client)
            user = rooms[room]['clients_names'][i]
            del rooms[room]['clients_sockets'][i]
            del rooms[room]['clients_names'][i]

            message = '{} left "{}" room'.format(user, room)
            print(message)
            send_message(message.encode(), room)
            break

# Receive a connections
def main():
    # Starting Server
    server = socket(AF_INET, SOCK_STREAM)
    server.bind(('localhost', 12345))
    server.listen()
    print('server is started')

    while True:
        # wait for a Connection
        client, address = server.accept()
        print("New Connection From {}".format(str(address)))

        # Request The Room Name
        client.send('<room>'.encode())
        room = client.recv(1024).decode()

        # Request The User Name
        client.send('<user>'.encode())

```

```

user = client.recv(1024).decode()

if room not in rooms:
    rooms[room] = {}
    rooms[room]['clients_sockets'] = []
    rooms[room]['clients_names'] = []

rooms[room]['clients_sockets'].append(client)
rooms[room]['clients_names'].append(user)

message = '{} joined "{}" room'.format(user, room)
print(message)
send_message(message.encode(), room)

# Start Worker Thread For the Client
thread = Thread(target=worker, args=(client, room))
thread.start()

main()

```

في التابع الرئيسي main نقوم بالتصت على البورت 12345 ثم بشكل متكرر نقوم باستقبال اتصال جديد ومطالبة العميل بإدخال اسمه واسم غرفة الدردشة ثم إرساله لـ worker thread يقوم بخدمته بشكل متزامن مع تنفيذ البرنامج الرئيسي.

التابع send_message يقوم بإرسال رسالة محددة لجميع أعضاء غرفة دردشة محددة. التابع worker يتم تنفيذه بشكل متفرع كـ worker thread لخدمة زبون ما، بحيث يقوم باستمرار باستقبال الرسائل من العميل وإرسالها لجميع العملاء الآخرين في نفس الغرفة. بالإضافة لطباعة الرسالة على نافذة السيرفر. وفي حال حصول Exception للاتصال مع العميل نقوم بحذف بياناته من الغرفة وإرسال رسالة بمغادرته من الغرفة.

يتم تخزين بيانات الغرف واتصالات العملاء بالـ dictionary المسماة rooms. بحيث تحتوي على dictionary خاصة بكل غرفة، ويكل dictionary فرعية يوجد List نحتفظ فيها بـ sockets العملاء و List نحتفظ بها بأسمائهم.

تطبيق العميل:

```
from socket import *
from threading import *

def receive_worker():
    while True:
        try:
            message = client.recv(1024).decode()
            if message == '<room>':
                client.send(room.encode())
            elif message == '<user>':
                client.send(user.encode())
            else:
                print(message)
        except:
            print("connection Error")
            client.close()
            break

def send_worker():
    while True:
        user_message = input('')
        message = '{}> {}'.format(user, user_message)
        client.send(message.encode())
```



```

# main

# Enter Room name
room = input("Enter room name: ")
# Enter User name
user = input("Enter your name: ")

client = socket(AF_INET, SOCK_STREAM)
client.connect(('localhost', 12345))

receive_thread = Thread(target=receive_worker)
receive_thread.start()

send_thread = Thread(target=send_worker)
send_thread.start()

```

في برنامج العميل نقوم بمطالبتة بإدخال اسمه واسم غرفة الدردشة ثم نقوم بإنشاء worker thread لاستقبال الرسائل بشكل متكرر. وأخرى لإرسال الرسائل بشكل متكرر.

السيناريو الثاني: غرف دردشة تتبادل البيانات مشفرة بطريقة Point to Point.

في هذا السيناريو نستخدم التشفير المتناظر الذي يزودنا به الصنف Fernet من الحزمة cryptography. يقوم العميل بإرسال المعلومات التالية للسيرفر: اسمه واسم الغرفة وكلمة مرور مستخدمة لتوليد مفتاح التشفير. يقوم العميل بتشفير البيانات قبل إرسالها للسيرفر وفك تشفير البيانات المستلمة من السيرفر. يمكن للسيرفر أن يقوم بفك تشفير البيانات المستقبلية من العميل ورؤيتها أو تخزينها بشكل صريح وإعادة إرسال البيانات مشفرة لعملاء الغرفة.

نقوم بتنصيب الحزمة أولاً:

```
pip install cryptography
```

نقوم بإضافة تابع يقوم بتوليد مفتاح التشفير من كلمة المرور:

```
import base64
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.kdf.pbkdf2 import PBKDF2HMAC

def keygen(password):
    kdf = PBKDF2HMAC(
        algorithm=hashes.SHA256(),
        length=32,
        salt=b'secrete',
        iterations=390000,
    )
    key =
base64.urlsafe_b64encode(kdf.derive(password.encode()))
    return key
```

تطبيق السيرفر:

نطالب المستخدم بإدخال كلمة المرور أيضاً. ونقوم بتخزينها ضمن معلومات الغرفة.

```
# Request The Room Password
client.send('<password>'.encode())
password = client.recv(1024).decode()
```

نقوم بالتعديلات التالية على ال worker بحيث يقوم بفك تشفير البيانات قبل عرضها:

```
def worker(client, room, password):
    # Key Generate
    key = keygen(password)

    # Cryptography Fernet object
    f = Fernet(key)

    while True:
        try:
            enc_message = client.recv(1024)
```

```

message = f.decrypt(enc_message)
print(message.decode())
send_message(enc_message, room)

```

تطبيق العميل:

نقوم في تطبيق العميل بإدخال كلمة المرور وتوليد مفاتيح التشفير

```

# Enter User name
password = input("Enter room password: ")
# Key Generate
key = keygen(password)

# Cryptography Fernet object
f = Fernet(key)

```

وأيضا نضيف التشفير قبل الإرسال وفك التشفير بعد الاستقبال.

```

def send_worker():
    while True:
        user_message = input('')
        message = '{}> {}'.format(user, user_message)
        enc_message = f.encrypt(message.encode())
        client.send(enc_message)

```

```

def receive_worker():
    while True:
        try:
            message = client.recv(1024)
            if message.decode() == '<room>':
                client.send(room.encode())
            elif message.decode() == '<user>':
                client.send(user.encode())
            elif message.decode() == '<password>':
                client.send(password.encode())
        else:
            try:
                message = f.decrypt(message).decode()

```

```

except:
    message = message.decode()
    print(message)
except Exception as e:
    print("connection Error")
    client.close()
    break

```

السيناريو الثالث: غرف دردشة تتبادل البيانات مشفرة بطريقة End to End.

في هذا السيناريو نستخدم أيضاً التشفير المتناظر الذي يزودنا به الصنف Fernet من الحزمة cryptography. في هذا السيناريو لا نقوم بتعديل تطبيق السيرفر عن السيناريو الأول أبداً. وذلك بسبب أن التشفير يتم بين العملاء من عميل إلى عميل دون معرفة السيرفر لمفتاح التشفير. نقوم فقط بتعديل تطبيق العميل بتعديلات مشابهة لتعديلات السيناريو الثاني باختلاف أنه لا نقوم بإرسال كلمة المرور للسيرفر. لا يمكن للسيرفر أن يقوم بفك تشفير البيانات المستقبلية من العميل ورؤيتها أو تخزينها بشكل صريح وإعادة إرسال البيانات مشفرة لعملاء الغرفة.

```

def receive_worker():
    while True:
        try:
            message = client.recv(1024)
            # print(message)
            if message.decode() == '<room>':
                client.send(room.encode())
            elif message.decode() == '<user>':
                client.send(user.encode())
            else:
                try:
                    message = f.decrypt(message).decode()
                except:

```

```

        message = message.decode()
        print(message)
    except Exception as e:
        print("connection Error")
        client.close()
        break

```

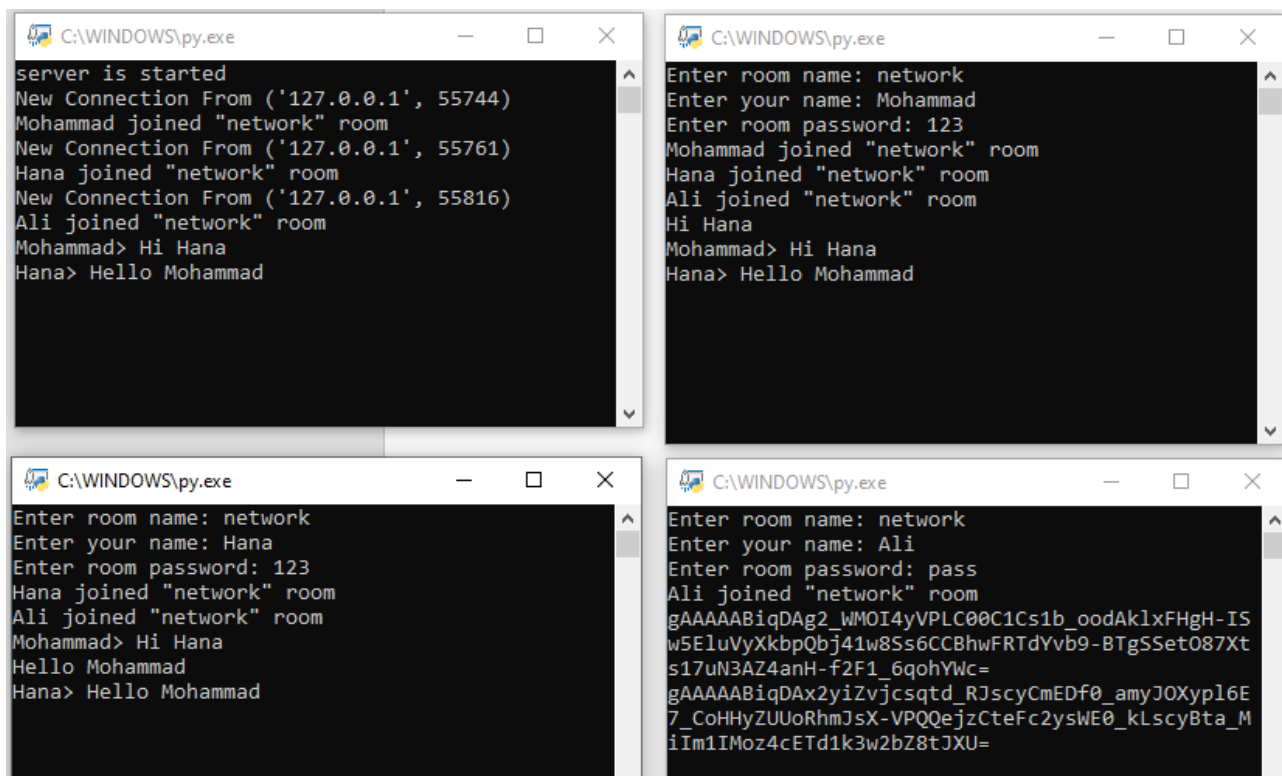
اختبار النتائج:

السيناريو الأول:

The image displays six screenshots of a Python chat application running in Windows command prompts, arranged in a 2x3 grid. Each window shows the progression of a chat room where users join and send messages.

- Top Left:** Shows the initial state where Hana joins the "programming" room. It then shows Mohammad Nour joining and sending "Hello Hana".
- Top Middle:** Continues the chat where Hana sends "Hi" and Mohammad Nour responds with "Hello Hana".
- Top Right:** Shows the same chat history as the top middle window.
- Bottom Left:** Shows a new room named "syria" being created. Ahmad, Ali, and Sara join in sequence. Ahmad sends "Hi all" and Sara responds with "Hi...".
- Bottom Middle:** Continues the "syria" room chat where Ali and Sara join, and Ahmad sends "Hi all".
- Bottom Right:** Continues the "syria" room chat where Sara joins and sends "Hi...".

السيناريو الثاني:



```

C:\WINDOWS\py.exe
server is started
New Connection From ('127.0.0.1', 55744)
Mohammad joined "network" room
New Connection From ('127.0.0.1', 55761)
Hana joined "network" room
New Connection From ('127.0.0.1', 55816)
Ali joined "network" room
Mohammad> Hi Hana
Hana> Hello Mohammad

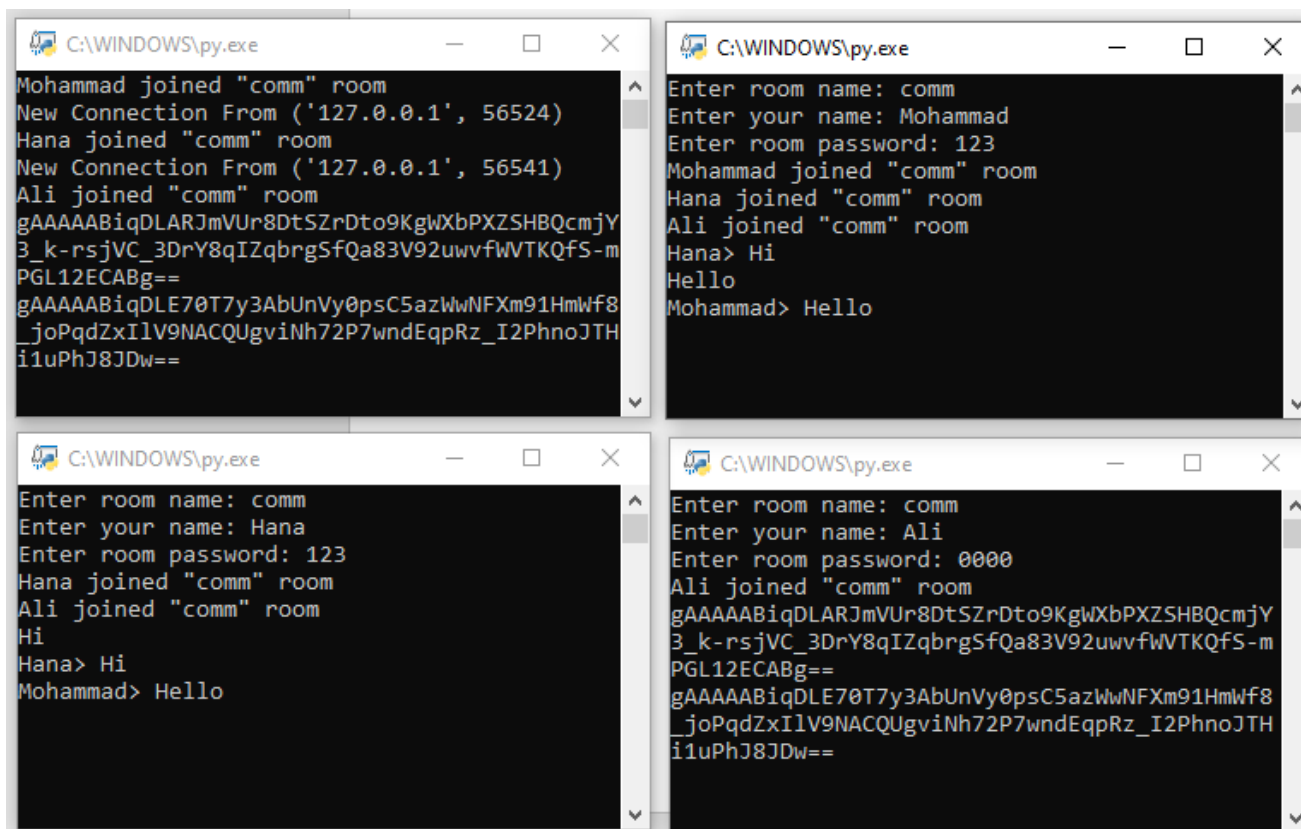
C:\WINDOWS\py.exe
Enter room name: network
Enter your name: Mohammad
Enter room password: 123
Mohammad joined "network" room
Hana joined "network" room
Ali joined "network" room
Hi Hana
Mohammad> Hi Hana
Hana> Hello Mohammad

C:\WINDOWS\py.exe
Enter room name: network
Enter your name: Hana
Enter room password: 123
Hana joined "network" room
Ali joined "network" room
Mohammad> Hi Hana
Hello Mohammad
Hana> Hello Mohammad

C:\WINDOWS\py.exe
Enter room name: network
Enter your name: Ali
Enter room password: pass
Ali joined "network" room
gAAAAABiqDag2_WM0I4yVPLC00C1Cs1b_oodAk1xFHgH-IS
w5EluVyXkbpQbj41w8Ss6CCBhwFRTdYvb9-BTgSset087Xt
s17uN3AZ4anH-f2F1_6qohYWc=
gAAAAABiqDax2yiZvjcsqtd_RJscyCmEDf0_amyJOXyp16E
7_CoHHyZUuoRhmJsX-VPQqeJzCteFc2ysWE0_kLscyBta_M
iIm1IMoz4cETd1k3w2bZ8tJXU=

```

السيناريو الثالث:



```

C:\WINDOWS\py.exe
Mohammad joined "comm" room
New Connection From ('127.0.0.1', 56524)
Hana joined "comm" room
New Connection From ('127.0.0.1', 56541)
Ali joined "comm" room
gAAAAABiqDLARJmVUr8DtSZrDto9KgWxbPXZSHBQcmjY
3_k-rsjVC_3DrY8qIZqbrgSfQa83V92uwwfWVKQfS-m
PGL12ECABg==
gAAAAABiqDLE70T7y3AbUnVy0psC5azWwNFXm91HmWf8
_joPqdZxILV9NACQUgviNh72P7wndEqRz_I2PhnoJTH
i1uPhJ8JDw==

C:\WINDOWS\py.exe
Enter room name: comm
Enter your name: Mohammad
Enter room password: 123
Mohammad joined "comm" room
Hana joined "comm" room
Ali joined "comm" room
Hana> Hi
Hello
Mohammad> Hello

C:\WINDOWS\py.exe
Enter room name: comm
Enter your name: Hana
Enter room password: 123
Hana joined "comm" room
Ali joined "comm" room
Hi
Hana> Hi
Mohammad> Hello

C:\WINDOWS\py.exe
Enter room name: comm
Enter your name: Ali
Enter room password: 0000
Ali joined "comm" room
gAAAAABiqDLARJmVUr8DtSZrDto9KgWxbPXZSHBQcmjY
3_k-rsjVC_3DrY8qIZqbrgSfQa83V92uwwfWVKQfS-m
PGL12ECABg==
gAAAAABiqDLE70T7y3AbUnVy0psC5azWwNFXm91HmWf8
_joPqdZxILV9NACQUgviNh72P7wndEqRz_I2PhnoJTH
i1uPhJ8JDw==

```

5.المراجع:

- [1] Mundink, Ben Rothke, David "End-to-End Encryption: The Holy Grail of PCI Security". CSO Online. Retrieved 11-04-2020.
- [2] <https://arrowpayments.com/blog/why-p2pe-will-always-beat-e2ee-the-definitive-guide/>
- [3] <https://docs.adyen.com/development-resources/e2ee-p2pe-comparison>
- [4] <https://www.schellman.com/blog/what-is-p2pe-an-assessment-matter>