تطوير نظام للمحادثات الآمنة والموثوقة عبر الشبكة

"نظام زاجل"

اعداد:

سعيد محمد سعيد عطا الله

ديسمبر 2013م

بسم الله الرحمن الرحيم

"نظام زاجل للمحادثات الآمنة و الموثوقة"
بحث تكميلي لدرجة البكلاريوس
في علوم الحاسوب – 2013م
سعيد عطا الله – جامعة أمدرمان الأهلية
2014م – جميـع الحقـوق محفـوظة

الاهداء

إلى والدي العزيزين الذين لن أتمكن من أن أوفيهما حقهما مهما فعلت ؛ و إلي كل من علمني حرفا.

المستخلص

هدف هذا البحث هو تطوير نظام لاجراء المحادثات بشكل آمن و موثوق عبر الشبكة و شبكة انترنت تحديدا ؛ و ذلك عن طريق دراسة علم التشفير التطبيقي و قواعد أمن المعلومات و توظيفها عند تطوير النظام المذكور. كما يهدف البحث إلى توفير ملخص مختصر لأساسيات هذا الحقل المهم باللغة العربية و الذي تندر فيه المؤلفات بلغة الضاد بالرغم من أهميته البالغة.

كذلك يهدف البحث إلى إلقاء الضوء – بشكل عملي – على المصادر المفتوحة، و التي هي وسيلة ممتازة للتعلم و التطبيق العملي عند دراسة علوم الحاسوب و تطوير البرمجيات؛ حيث استفاد الباحث من عدة مكتبات برمجية مفتوحة المصدر في تطوير النظام.

Abstract

The purpose of this study is to develop system software for perform secure and reliable conversations over Internet, by studying Applied Cryptography and rules of Information Security and employing them to develop the mentioned system.

This work also aims to provide a short Arabic summary for the fundamentals of this important field which lacks Arabic books in spite of its highly importance.

Besides that, it intends to shed a light, practically, on the open source software, which is an excellent facility for learning and practical application when studying computer science and software development, where the researcher benefited from several open source libraries to develop the system.

الباب الثاني: التحليل وتحديد المتطلبات

الباب الثالث: تصميم النظام

الباب الرابع: التطبيق العملي ونوافذ البرنامج

فهرس الجداول

فهرس الأشكال

المقدمة

تزايد الاعتماد على أجهزة الحاسوب و شبكات الحاسوب في أغلب المجالات للاستفادة من الامكانيات الهائلة التي يوفرها في معالجة و تخزين المعلومات و نقلها بسرعة و دقة بالغتين ؛ إلا أن هناك كثيرا من التحديات التي ظهرت مع التوسع الكبير في استخدام الحاسوب و شبكات الحاسوب ؛ و من أكبر هذه التحديات مسألة حماية و تأمين المعلومات التي يتم تخزينها في الحاسوب و نقلها عبر شبكات الحاسوب و شبكة انترنت تحديدا.

في الواقع هذه المشكلة ليست وليدة العصر ؛ فمنذ قديم الزمان بحث الانسان عن وسائل لتأمين المعلومات المهمة و الحساسة خاصة في الجانب العسكري، و نتيجة لذلك ظهر علم التشفير أو التعمية (Cryptography) و تم اكتشاف و تطبيق خوارزميات متعددة من أشهرها خوارزمية قيصر ؛ ثم بعد ظهور الحواسيب و الأهمية البالغة للمعلومات و حمايتها و تأمينها أصبح هذا الأمر في غاية الأهمية خاصة في التطبيقات العصرية المهمة مثل الحكومة الالكترونية و التجارة الالكترونية و الصيرفة الالكترونية و نحوها.

بالرغم مما ذكر أعلاه لاحظ الباحث ندرة بالغة في الأبحاث المكتوبة بالعربية في هذا الحقل المهم، و لعل ذلك يرجع إلى عدة أسباب : الأول هو اعتماد علم التشفير و خوارزمياته بشكل كبير على علم الرياضيات و على كم كبير من النظريات الرياضية التي يجب دراستها و فهمها جيدا قبل الشروع في البحث في الموضوع ؛ كما أن الجانب التطبيقي يحتاج إلى معرفة جيدة بمفاهيم البرمجة و الالمام بلغات برمجة متقدمة كلغة سي بلس بلس و نحوها من لغات البرمجة المستخدمة في الأغراض العلمية و الرياضية. أما السبب الثاني فهو ندرة الكتب و المراجع العربية في هذا الحقل. و السبب الثالث هو وفرة الأنظمة الحاسوبية التي توفر أمن المعلومات بشكل جيد مما يدفع بعض الباحثين إلى الاعتقاد بعدم الحاجة للبحث أو كتابة مزيد من البرمجيات في هذا الحقل ؛ و هذا مفهوم خاطئ لأن مبرمج أو محلل أمن المعلومات ليس بالضرورة أن يبرمج نظاما من الصفر بل قد يسهم في تطوير كثير من الأنظمة مفتوحة المصدر و يقوم بتوظيفها لسد الحاجة الماسة لأمن المعلومات.

الباحث

2013/12/4م

الباب الأول

مقدمة و نظرة عامة

1.1 تمهيد:

هذا الباب يقدم مقدمة نظرية أو عرض عام لمواضيع مختلفة في علم التشفير، و تعريف لأهم المصطلحات و عرض لأهم أساسيات هذا الموضوع، بحيث يتمكن القاريء من الالمام بالمفاهيم المهمة في الموضوع.

2.1 ما هو علم التشفير ؟ :

في هذه الفقرة يتم تعريف علم التشفير و ذكر بعض الجوانب الأساسية المتعلقة بالموضوع باختصار:

1.2.1 تعريف:

علم التشفير أو التعمية (Cryptography) : هو دراسة تقنيات رياضية (Mathematical Techniques) و استخدامها في الجوانب المختلفة المتعلقة بأمن المعلومات بغرض تحقيق مجموعة من الأهداف.

يستخدم التشفير في الحفاظ على سرية المعلومات عن طريق تحويلها إلى رموز عشوائية غير مفهومة (gibberish) و بالتالي لن يتمكن المخترق (قرصان المعلومات)[1] من الاطلاع على محتواها حتى و إن تمكن من الحصول عليها ؛ كما يستخدم في ضمان تكامل و سلامة المعلومات عن طريق التحكم في الوصول لهذه المعلومات و حصره في الشخص المصرح له فقط (أي يملك هوية صالحة). في حالة حدوث خلل و تمكن المخترق من الوصول للمعلومات و تعديلها، يجب أن نتمكن من اكتشاف أن هذه البيانات قد تم تعديلها و ليست على حالتها الأصلية التي يجب أن تكون عليها. كذلك من الأهداف ؛ عدم الانكار، أي إذا أثبتنا أن الرسالة (المعلومات) صحيحة و لم يتم التلاعب بها و أنها صادرة فعلا من الشخص المعني، هذا الشخص ملزم بها و لا يستطيع إنكارها ؛ مثلا يطلب العميل من المصرف الذي يتعامل معه إلكترونيا أن يقوم بتحويل مبلغ مالي كبير من حسابه إلى حساب شخص آخر، لا يستطيع هذا العميل إنكار ذلك و إلقاء المسئولية على المصرف مادمنا نستطيع باستخدام علم التشفير اثبات صحة هذه الرسالة و أنها صادرة منه شخصيا.

(1) المخترق قد يكون شخصا و قد يكون جهة أو مؤسسة حكومية أو غير حكومية.

2.2.1 جوانب علم التشفير الأساسية:

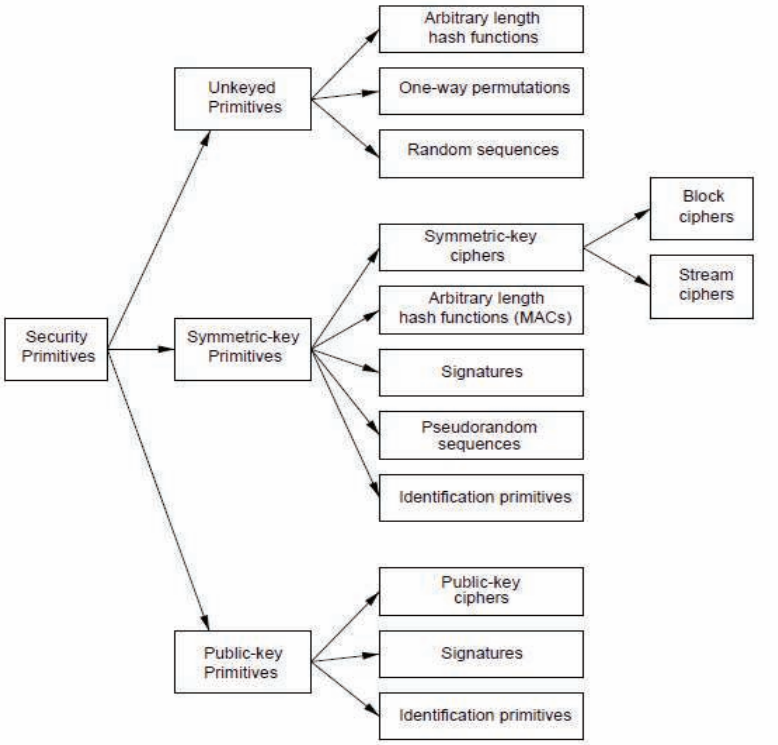
يضم علم التشفير جوانب متعددة – انظر شكل (1.1) – و يقسم من حيث استخدام المفاتيح و نوعها إلى ثلاثة أقسام رئيسية كالتالي:

– قسم متعلق بالمفتاح غير المتناظر – أي بالمفتاح العلني – (Public-Key) و يضم ثلاثة جوانب: التشفير و التوقيع الرقمي و التحقق من الهوية.

– قسم متعلق بالمفتاح المتناظر (Symmetric-key)، و يضم الجوانب التالية: التشفير (تشفير الكتلة Block Cipher و تشفير التيار Stream Cipher) ؛ و توابع (دوال) الاختزال (Hash functions) التي تستخدم للتحقق من صحة الرسالة (MAC) ؛ و التوقيع الرقمي ؛ و الأعداد (البتات) شبه العشوائية ؛ بالاضافة إلى التحقق من الهوية.

– قسم لا يوجد لديه ارتباط بالمفاتيح، و يضم: توابع الاختزال "الهاش" ؛ و عملية التقليب (الابدال) ؛ و الأعداد (البتات) العشوائية.

شكل (1.1): أهم جوانب علم التشفير الأساسية



المصدر: Menezes, et Al, Handbook of Applied Cryptography.

3.2.1 مفاهيم أخرى مهمة :

- التشفير بالاحلال (Substitution) و التشفير بالابدال – التقليب – / (Transposition Permutation) : هما تقنيتان كانتا تستخدمان في طرق التشفير السابقة لعصر الحاسوب ، حيث يتم – في الطريقة الأولى – تغيير بعض الأحرف من النص الأصلي (plaintext) بأحرف أخرى بدلالة مفتاح (key) محدد وباستخدام خوارزمية معينة، فينتج لنا نص مشفر (ciphertext) أما في الطريقة الأخرى – الابدال – فلا يتم تغيير الأحرف و لكن يتم ابدالها – أو تقليبها – أي تبديل مواقع الأحرف في نفس النص، و أحيانا تستخدم الطريقتين لزيادة قوة التشفير. هاتان الطريقتان تستخدمان الآن في كثير من خوارزميات التشفير الحديثة و لكن هنا التعامل يتم مع البتات (bits) و ليس الأحرف و بالتالي يمكن تشفير أي نوع من أنواع البيانات بغض النظر عن اللغة المكتوب بها النص أو الملف، و في أغلب الخوارزميات يتم تكرار عمليتي الاحلال و الابدال عدة مرات لضمان مستوى أعلى من التشفير.
- النص الأصلي (plaintext) : هو البيانات (أو الرسالة) المراد تشفيرها باستخدام خوارزمية التشفير.
- خوارزمية التشفير (Encryption Algorithm): هي الخوارزمية التي تقوم بتشفير النص الأصلي باستخدام المفتاح – السري – و الناتج هو النص المشفر.
- المفتاح السري (secret key) : هو عدد (أو سلسلة من البتات العشوائية) تستخدمها خوارزمية التشفير لتشفير النص الأصلي، و لابد من الحفاظ على سريته لأن كشفه يعني كشف سرية المعلومات التي تم تشفيرها باستخدامه.
- النص المشفر (ciphertext) : هو الناتج عن عملية التشفير، و هو عبارة عن سلسلة عشوائية من الحروف (أو البتات) التي نتجت عن ادخال المفتاح و النص الأصلي إلى خوارزمية التشفير المستخدمة.
- خوارزمية فك التشفير (Decryption Algorithm) : التشفير هو عملية عكسية أي لابد من استخدام خوارزمية فك التشفير لفك التشفير و ارجاع النص المشفر لحالته الأصلية ؛ في العادة خوارزميات التشفير عكسية، أي يتم اعطاءها النص الأصلي و المفتاح – مدخلات – و ينتج عن ذلك النص المشفر – مخرجات – و العكس صحيح ؛ أدخل النص المشفر و نفس المفتاح فيرجع نفس النص الأصلي.

3.1 التشفير و أمن المعلومات :

يعتمد أمن المعلومات بشكل كبير على علم التشفير و البروتوكولات و الخوارزميات التي يوفرها لتحقيق مجموعة مهمة من الأهداف التي يجب توفرها لحماية الكثير من المعلومات المهمة و الحساسة. (أسرار الدول – الخطط العسكرية – الأسرار التجارية للشركات و المؤسسات الحكومية و الخاصة – حسابات البنوك – عمليات التجارة الالكترونية و غيرها الكثير) حيث أن هذه المعلومات قد تتعرض للكشف أو التعديل غير الشرعي عند انتقالها عبر الشبكة ؛ من أهم هذه الأهداف التي يمكن لعلم التشفير توفيرها:

- خصوصية و سرية المعلومات (Privacy & Confidentiality) : أي الحفاظ على سرية المعلومات بحيث لا يطلع عليها إلا الشخص المصرح له فقط و الذي يملك هوية صالحة.
- تكامل و سلامة البيانات (Data Integrity) : ضمان عدم تعديل المعلومات (كالاضافة أو الحذف أو التغيير) من قبل شخص غير مصرح له أو بطريقة غير معروفة.
- التحقق من الهوية (Entity Authentication) : التثبت و التأكد من هوية من يتعامل مع المعلومات (سواء كان شخصا أو برنامجا أو جهاز حاسوب أو غير ذلك) و من أنه هو فعلا الشخص المطلوب الذي يملك التصريح المناسب.
- التحقق من صحة الرسالة (Message Authentication) : اثبات صحة و أصالة الرسالة (المعلومات)، أي أنه لم يتم تعديلها أو التلاعب بها بواسطة شخص غير مصرح له أو بطريقة غير معروفة.
- عدم الانكار (Non-Repudiation) : ضمان عدم انكار مصدر الرسالة (المعلومات) لها أو ادعاؤه عدم مسؤليته عنها.

4.1 التشفير المتناظر (Symmetric Cryptography) :

سمي بالمتناظر لأن مفتاحي التشفير و فك التشفير متناظران ؛ أو بعبارة أخرى نفس المفتاح المستخدم في تشفير البيانات يستخدم في فك تشفير هذه البيانات ؛ إذا لم يتم استخدام نفس المفتاح

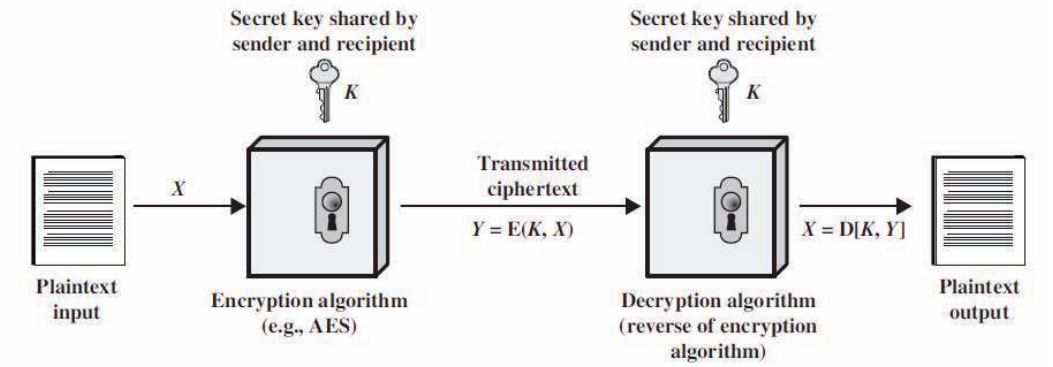
فبالطبع سيكون الناتج مختلفا عن البيانات الأصلية (plaintext) ؛ الشكل (1.2) أدناه يوضح هذه العملية.

هناك خوارزميات كثيرة و مشهورة من هذا النوع لعل أشهرها خوارزمية معيار تشفير البيانات (DES) و هي أول معيار عالمي في التشفير ظهرت في منتصف سبعينيات القرن الماضي، و انتشر استخدامها بشكل واسع ؛ ثم ظهرت في العام 2001م خوارزمية معيار التشفير المتقدم (AES) كمعيار عالمي جديد، و سنتطرق لتفاصيل ذلك في الباب التالي.

التشفير المتناظر هو الأكثر استخداما ، حيث يوفر مستوى جيدا من الأمان – مع مراعاة أن يكون المفتاح بالطول و العشوائية المطلوبين – مع مستوى مقبول من حيث سرعة الأداء ؛ لكن ما يعيبه هو صعوبة توزيع المفاتيح، حيث أنه لايمكن فك التشفير

إلا بنفس المفتاح ؛ إذا أرسلت رسالة مشفرة لصديقك من خلال قناة اتصال غير مؤمنة – كشبكة انترنت مثلا – فكيف

الشكل (1.2) : التشفير/فك التشفير باستخدام التشفير المتناظر



المصدر: William Stallings, Cryptography and Network Security.

تستطيع ارسال المفتاح له ؟ ، خاصة إذا كان موجودا في بلد بعيد عنك ؛ و إذا استعنت بطرف آخر موثوق (Trusted Third Party) فما هو الضامن أن هذا الطرف الثالث لن يقوم باستخدام هذا المفتاح السري في الاطلاع على كل رسائلك السرية ؟ ! . أمثال هذه الأسئلة دفعت العلماء إلى البحث لتطوير تقنية تمكن من حل هذه الاشكالات بأيسر الطرق.

5.1 التشفير غير المتناظر (التشفير بالمفتاح العلني Public-key) Cryptography :

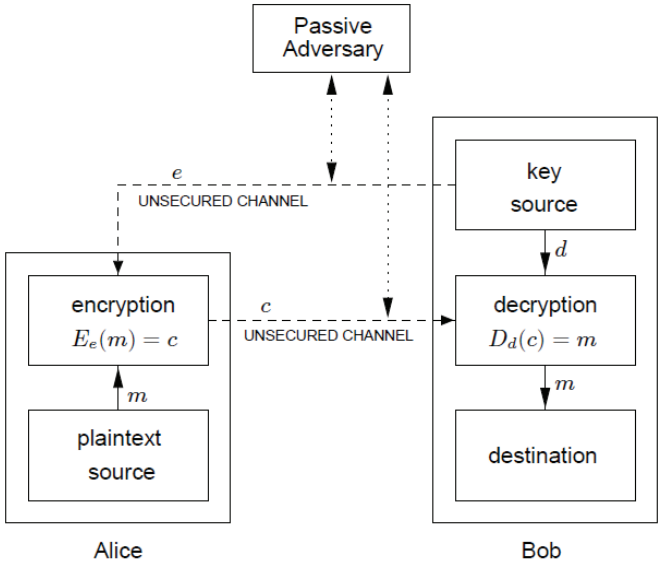
في العام 1976م قدم العالمان Whitfield Diffie و Martin Hellman ورقة بحث علمي في مجال التشفير اقترحا فيها استخدام التشفير غير المتناظر أي استخدام زوجين من المفاتيح أحدهما للتشفير – يكون علنيا و معروفا للجميع – و الآخر لفك التشفير – و يجب على صاحبه ابقاؤه سريا – مما يوفر حلا لمسألة توزيع المفاتيح لأن مفتاح التشفير علني و يمكن توزيعه بسهولة ؛ ثم ظهرت بعد ذلك و في السنوات اللاحقة عدة خوارزميات عملية تطبق هذا المفهوم ؛ من أشهرها خوارزمية (RSA) – تستخدم للتشفير و كذلك للتوقيع الرقمي – و خوارزميتي (DSA) و (ECDSA) – و الأخيرتان تستخدمان للتوقيع الرقمي – و تم اعتماد الثلاثة خوارزميات كمعيار معتمد للتوقيع الرقمي (DSS) فى الاصدارة FIPS-186-3 و التي صدرت من معهد (NIST) في يونيو 2009م[1] – و سيأتي تفصيل أكثر في موضعه بالباب الثاني.

في هذا النوع من التشفير كل شخص له مفتاحان أحدهما خاص (سري) يستخدم لفك التشفير و الآخر علني يستخدم للتشفير (و يمكن توليده من المفتاح الخاص و العكس غير صحيح) ؛ إذا أردت ارسال رسالة سرية لشخص ما فكل ما عليك هو الحصول على مفتاحه العلني و تشفيرها به ثم ارسال الرسالة المشفرة له و لن يتمكن شخص آخر من الاطلاع عليها لأن مفتاح فك التشفير (المفتاح الخاص) سري و لا يعرفه أحد إلا صاحبه، كما يوضحه الشكل (1.3).

النقطة المهمة هنا أن المخترق لا يمكنه فك التشفير حتى و لو عرف المفتاح العلني و الذي هو غير سري، أي باستطاعة كل أحد الوصول له. ثم بناء على هذه التقنية تطور مفهوم التوقيع الرقمي الذي تمت الاشارة له سابقا. من أهم عيوب هذه التقنية هي الحاجة لاستخدام مفاتيح ذات طول كبير نسبيا – 1024 بتا أو أعلى – مما يجعل عملية التشفير أبطأ. أغلب أنظمة التشفير الحديثة تستخدم كلا التقنيتين للاستفادة من مميزات كلاهما.

(1) و هذه الاصدارة منشورة على موقع معهد NIST على الويب – انظر قائمة المراجع في آخر البحث.

الشكل (1.3): استخدام التشفير غير المتناظر لحل مشكلة توزيع المفاتيح باستخدام قناة غير آمنة



المصدر: Menezes, et Al, Handbook of Applied Cryptography.

6.1 مقارنة بين التشفير المتناظر و التشفير غير المتناظر:

كلا النوعين له مميزاته و عيوبه، سنستعرضها هنا باختصار ؛ و مما يجب ذكره هنا أن المقارنة لن تكون حول مستوى أو قوة التشفير لأن كلا النوعين – باستخدام البروتوكولات و الخوارزميات القياسية – يوفران مستوى جيدا من التشفير.

سيتم اجراء مقارنة بين النوعين لتوضيح محاسن و عيوب كل نوع على شكل جداول كما يلي:

1.6.1 مقارنة محاسن كلا النوعين :

جدول (1.1) مقارنة بين محاسن التشفير المتناظر و التشفير غير المتناظر.

م	محاسن التشفير المتناظر	م	محاسن التشفير غير المتناظر
1	يقدم سرعة عالية في الأداء (قد تصل إلى عدة ميقابتات في الثانية الواحدة).	1	المفتاح الخاص فقط هو الذي يجب ابقاؤه طي الكتمان، أما المفتاح العلني فيمكن نشره.
2	المفاتيح المستخدمة في هذا النوع أقصر بكثير من النوع الآخر، بالتالي يسهل الاحتفاظ بها و نقلها.	2	عملية إدارة و توزيع المفاتيح في هذا النوع أسهل بكثير.
3	مفاتيح هذا النوع يسهل توظيفها في تقنيات أخرى مهمة كتوليد الأرقام شبه العشوائية و توابع الاختزال.	3	زوج المفاتيح (الخاص و العلني) يمكن الاحتفاظ بهما لفترة طويلة قد تصل لعدة سنوات.
4	مفاتيح هذا النوع يمكن تركيبها و دمجها لتوفير مستوى أعلى من التشفير و التعمية.	4	التوقيع الرقمي المعتمد على هذا النوع عمليا أفضل، حيث أن المفاتيح المستخدمة في عمليتي التوقيع و التحقق من صحة التوقيع تكون أقصر بكثير.
5	التشفير بالمفتاح المتناظر له تاريخ طويل و يضم الكثير من الخوارزميات المشهورة التي قام الكثير من العلماء و الباحثين بتجربتها و اختبارها عبر الزمن و بالتالي تقدم مستوى عالٍ من الموثوقية.	5	عند اجراء الاتصالات عبر الشبكات ، سيكون عدد المفاتيح المطلوب أقل عند استخدام هذا النوع من التشفير.

2.6.1 مقارنة عيوب كلا النوعين:

جدول رقم(1.2) يوضح عيوب التشفير المناظر و التشفير غير المتناظر.

م	عيوب التشفير المتناظر	م	عيوب التشفير غير المتناظر
1	عند اجراء اتصال بين طرفين على الشبكة، لابد من ابقاء المفاتيح سرية للمحافظة على سرية المحادثة.	1	التشفير بالمفتاح العلني أبطأ كثيراً من التشفير بالمفتاح المتناظر.
2	في الشبكات الواسعة ستكون عملية إدارة و توزيع المفاتيح مسألة شاقة. هذا في الواقع أكبر مساويء التشفير المتناظر.	2	التشفير بالمفتاح العلني يستخدم مفاتيح أطول بكثير من النوع المستخدم في النوع الآخر و بالتالي ستكون عملية حفظها و التعامل معها أصعب.
3	الحاجة إلى تغيير المفاتيح بشكل دوري، حيث أن استخدام نفس المفتاح لفترة طويلة قد يعرض المعلومات السرية للخطر.	3	عدد الخوارزميات المستخدمة في التشفير غير المتناظر التي أثبتت كفاءتها و إمكانية تطبيقها عمليا أقل بكثير مقارنة بالنوع الآخر.
4	تقنية التوقيع الرقمي المبنية على هذا النوع غير عملية حيث تحتاج لمفاتيح طويلة.	4	خوارزميات التشفير بالمفتاح العلني لها تاريخ قصير نسبياً، أي لم يتم تجربتها بكثافة كبيرة و لفترة طويلة

3.6.1 ملخص المقارنة : أيهما أفضل و لماذا ؟ :

كلا النوعان – كما رأينا سابقا – له مميزاته و عيوبه ؛ لذلك قام العلماء – في أنظمة التشفير الحديثة – باستخدام النوعين معا لتحقيق الفائدة القصوى و اجتناب مساوىء كل منهما، لتوضيح ذلك لنفترض هذا "السناريو" المبسط كالتالي:

هناك شخصان أو طرفان ؛ محمد و علي ، يرغبان في اجراء محادثة آمنة (مشفرة) على الشبكة ، يجب أن يقوما بالآتي:

- يقوم كلاهما بالاتفاق على اختيار خوارزمية تشفير قياسية و معتمدة (و هذا النوع من الخوارزميات علني و معروف لدى الجميع) حتى يستخدمانها في تشفير الرسائل بينهما.

- ثم يقوم كل منهما بتوليد مفتاح للتشفير المتناظر خاص به (و يسمى مفتاح الجلسة Session Key) و ابقاؤه طي الكتمان ، (سيستخدم كل طرف مفتاحه في تشفير الرسائل التي سيرسلها للطرف الآخر).
- يتم استخدام التشفير بالمفتاح العلني في عملية توزيع المفتاح المتناظر و توقيعه رقميا كما يلي:
- يقوم محمد أولاً بتشفير مفتاحه المتناظر بمفتاح التشفير غير المتناظر الخاص (السري) – أي يقوم بتوقيعه[1] بمفتاحه الخاص غير المتناظر – أولا ثم بعد ذلك يقوم بتشفير الناتج بمفتاح التشفير غير المتناظر العلني الذي يتبع لعلي – بذلك نضمن أن على هو الوحيد القادر على الاطلاع على محتوي الرسالة التي تحوي مفتاح التشفير المتناظر الذي سيستخدمه محمد – ، ثم يقوم بارساله إلى علي، و يقوم علي بنفس الشي من جانبه.
- عندما يستلم علي ما سبق من محمد يقوم بعكس العملية (حيث أن التشفير و فك التشفير عمليتان عكسيتان) ؛ أولا يستخدم مفتاحه للتشفير غير المتناظر الخاص – أي السري – لفك تشفير الرسالة – و التي هي مفتاح الجلسة الخاص بمحمد – ثم يأخذ الناتج و يستخدم المفتاح العلني الخاص بمحمد ليطبقه لفكك تشفير الرسالة مرة أخرى ، و بهذا يتأكد من أن توقيع الرسالة صحيح لأن محمد قام بتشفيرها بمفتاحه السري للتشفير غير المتناظر و لا يمكن فك هذا التشفير إلا بمفتاحه العلني.
- يقوم محمد كذلك بالمقابل بتطبيق نفس العملية و بالتالي يحصل كل منهما على مفتاح الجلسة الخاص بالآخر.
- يقوم محمد بتشفير الرسائل بمفتاح الجلسة – التشفير المتناظر – الخاص به و يرسلها له ، و كذلك يفعل علي.
- لقراءة الرسائل المرسلة من علي يقوم محمد باستخدام مفتاح الجلسة الخاص بعلي و الذي حصل عليه باستخدام التشفير غير المتناظر – كما هو مشروح أعلاه – يستخدم محمد هذا المفتاح لفك تشفير الرسائل القادمة من علي و بالمقابل يقوم علي بنفس الشيء.

في هذا السيناريو يستفيد محمد و علي من مميزات التشفير المتناظر مثل سرعة عملية التشفير و فك التشفير (و هي دائما الجزء الذي يحتاج للوقت)، و سرعة عملية توليد المفاتيح ، و يتجنبان عيوب هذا النوع عن طريق استخدام التشفير غير المتناظر في عملية توزيع المفاتيح المتناظرة (مفاتيح الجلسة) و توقيعها رقميا، بالاضافة إلى أن مفاتيح التشفير غير المتناظر يمكن ابقاؤها لفترة طويلة و بالتالي يمكن استخدامها لفترة قد تمتد لسنوات، و تجنب استخدامها لتشفير كمية كبيرة من البيانات (لأنها أبطأ) بل تشفر بها مفاتيح

(1) انظر الفقرة 8.1 لمزيد من التفصيل حول التوقيع الرقمي.

الجلسة و التي هي صغيرة نسبيا و بالتالي يمكن تغييرها في كل جلسة اتصال لتوفير مزيدا من الأمان ؛ أضف إلى ذلك استخدام التوقيع الرقمي المبني على التشفير غير المتناظر – و هو الأفضل – للتحقق من هوية كل طرف.

من الواضح أن هذه العملية معقدة و بها تفاصيل كثيرة و هناك جوانب تم اختصارها هنا لغرض التبسيط، لكن الرائع في الأمر أن استخدام الحاسوب للقيام بهذه العملية بشكل آلي سيجعل المسالة بسيطة و سهلة للغاية لمحمد و علي !. و مما يجدر ذكره أن هذا أحد أهم أهداف هذا البحث لتطوير برنامج يقوم بتطبيق هذه المفاهيم عمليا بحيث يتمكن المستخدمون من إجراء محادثات آمنة على شبكة الانترنت.

7.1 كيفية اختيار خوارزمية التشفير الأنسب ؟ :

هذا سؤال مهم جداً ؛ في السيناريو المذكور أعلاه كانت الفقرة الأولى هي قيام كلا من محمد و علي بالاتفاق على الخوارزمية التي سيستخدمانها ، لكن كيف سيتم الأختيار و على أي معيار؟ خاصة و أن هناك الكثير من خوارزميات التشفير التي تم تطويرها عبر السنوات ، هذه الفقرة مهتمة بالاجابة على هذا السؤال.

يرى بروس اسشنيير (Bruce Schneier)[1] أن خوارزميات التشفير العلنية (أي أن تفاصيلها جميعها تم دراستها و نشرها)، و المعروفة لدى علماء التشفير هي دائما الأفضل و الأكثر أمانا، و يعلل ذلك بأن هذه الخوارزميات تمت تجربتها و تحليلها و اختبارها من قبل كثير من الخبراء المستقلين – أي الذين لايتبعون لأي جهة حكومية كالسي آي إيه مثلا ! – و لفترات طويلة قد تصل لعشرات السنين، و بالتالي يقل احتمال وجود ثغرة بها. و هذا تعليل منطقي و مقبول بلا شك.

يمكنك اختراع خوارزمية تشفير و ابقاؤها سرية عن الجميع ، و لكن ما هو الضامن أنها خوارزمية صالحة و توفر المستوى المطلوب من الأمان؟ حيث أنه لم يتم اختبارها و تقييمها من الخبراء في المجال. لذلك من الأفضل دائما استخدام خوارزميات علنية و معروفة لدى الجميع حتى قراصنة المعلومات ، بالتالي لن يستفيد القرصان شيئاً من معرفة انك تستخدم الخوارزمية العلنية لأنه يدرك أنه بدون المفتاح من الصعب جدا فك تشفير الرسالة (المعلومات).

(1) من علماء التشفير المخضرمين، و له أبحاث و كتب مشهورة في علم التشفير ؛ كما قام – مع آخرين – بتصميم خوارزميات تشفير مشهورة ؛ مثل خوارزمية السمكة المنتفخة Blowfish و غيرها.

أمر آخر مهم ؛ من الأفضل دائما اختيار خوارزميات التشفير القياسية (المعيارية) ، حيث أن هناك الكثير من المعاهد و مراكز البحوث (حكومية أو غير حكومية) التي تقوم باصدار توصياتها و اجراء اختباراتها لتحديد الخوارزمية الأكثر أمانا و التي يمكن تطبيقها عملياً ، لا يتسع المجال هنا لذكر كل هذه الجهات التي تصدر معايير التشفير إلا أن من أشهرها المعهد القومي للمعايير و التكنولوجيا (NIST) و هو مؤسسة فدرالية أمريكية تصدر معايير التشفير و أمن المعلومات ؛ المعايير التي تستخدمها الحكومة الأمريكية ، بخاصة وزارة التجارة الأمريكية في تأمين المعلومات السرية – مثل حسابات البنوك و تفاصيل العمليات التجارية الالكترونية للمؤسسات التجارية المختلفة، و غير ذلك من المجالات الحيوية التي تهم الجميع. بالاضافة إلى ذلك تقوم هذه المؤسسات و المعاهد باختبار التطبيق العملي للخوارزمية (Implementation) و هذا أمر مهم جدا، لأنه قد يتم استخدام خوارزمية مشهورة و معتمدة و موثوقة، لكن التنفيذ البرمجي على الحاسوب قد يكون بطريقة غير مناسبة تؤدي لحدوث ثغرة تنتهك أمن المعلومات ، لذلك تقوم مؤسسة (NIST) باصدار نشرات دورية (تسمى: FIPS-PUB) تخصص فيه بعضها لتبيين نتائج اختبار مجموعة واسعة من أدوات التشفير التي تم تطويرها من قبل جهات مختلفة – سواء كانت برمجيات أو أجهزة تشفير – و تقوم باعتمادها بعد اجتيازها الاختبارات اللازمة لتحديد مستوى الأمان الذي تقوم بتوفيره، كما تصدر هذه المؤسسات معايير توضح البروتوكولات و الاجراءات التي يلزم اتباعها لتحقيق مستوى الأمان القياسي المطلوب .

8.1 التحقق و التوقيع الرقمي:

التحقق (Authentication) يستعمل للدلالة على أمرين : التحقق من الهوية ، أي أن هذا الشخص الذي يدعي أنه "محمد" هو فعلا الشخص المطلوب و الموجود الآن على الطرف الآخر من قناة الاتصال ؛ الأمر الآخر هو التحقق من صحة و أصالة الرسالة (المعلومات). التوقيع الرقمي (Digital Signature) يستخدم لربط هوية محددة (Identity) مع معلومات محددة ؛ بعبارة أخرى : يستخدم التوقيع الرقمي لإثبات أن هذه الرسالة المعينة هي صادرة من الشخص المطلوب – عند اجراء الاتصال عبر الشبكة – ؛ و من ناحية أخرى يستخدم للتحقق من عدم تعرض الرسالة للتعديل بشكل غير شرعي أثناء انتقالها عبر الشبكة.

من هذا تتضح أهمية التوقيع الرقمي في تحقيق هدفين من أهم أهداف أمن المعلومات ؛ التحقق من الهوية و التحقق من صحة و أصالة المعلومات.

هناك عدة بروتوكولات و آليات تستخدم التوقيع الرقمي – لا يتسع المجال هنا لتفصيلها – لكنها بشكل عام تقوم باستخدام توابع الاختزال (Hash functions) التي تقوم بتوليد رقم ثنائي فريد (يكتب عادة باستخدام الترميز الست عشري .Hex) ذو طول ثابت من الرسالة (المعلومات) المراد توقيعها و ذلك حسب الخوارزمية المستخدمة ؛ الفكرة هنا أن حدوث أي تغيير في الرسالة – حتى و إن كان التغيير في قيمة بت واحد فقط – فستولد توابع الاختزال رقما مختلفا عن الرقم الأول ، بالتالي سنتمكن من معرفة حدوث أي تغيير بالرسالة بسبب اختلاف الرقمين. من الجدير بالذكر أن توابع الاختزال هي ذات اتجاه واحد (One Way Function) أي أنها غير قابلة للعكس ؛ بعبارة أخرى : لايمكن معرفة محتوى الرسالة الأصلي من الرقم الذي قامت دالة الاختزال "الهاش" بتوليده.

دوال الـ"هاش" توفر جزءا من الحل ؛ التحقق من أن الرسالة لم يتم تعديلها ، الجزء الآخر أي اثبات صحة و هوية مصدر الرسالة ، يستخدم فيه خوارزميات مختلفة من أشهرها خوارزميات التشفير بالمفتاح العلني – كخوارزمية (RSA) مثلا – حتى و إن كان تشفير الرسالة غير مطلوب بل المطلوب التأكد من صحة الرسالة – أي عدم تعديلها – و من صحة مصدرها – أي التحقق من هوية المرسل لها- . بروتوكولات التوقيع الرقمي تقوم بتوليد رقم الـ"هاش" من الرسالة أولا (و بعض البروتوكولات تستخدم الختم الزمني أي كل رسالة يضاف لها التاريخ و الزمن عند المرسل ثم يتم توليد رقم الهاش منهما – أي من الرسالة و ختمها الزمني – معا للتأكد من مزامنة الطرف الآخر)، ثم يتم تشفير هذا الرقم باستخدام التشفير غير المتناظر ثم يتم ارسال الرسالة – بغض النظر عن كونها مشفرة أو ليست كذلك – و ارسال التوقيع الرقمي الخاص بها معها ؛ على مستلم الرسالة أن يقوم بعملية التأكد من صحة التوقيع عن طريق فك التشفير باستخدام المفتاح السري الخاص به أولا ثم فك تشفير الناتج مرة أخرى باستخدام مفتاح التشفير العلني الخاص بالشخص المرسل ثم يقوم بتوليد رقم الهاش للرسالة التي استلمها و يقارنه مع الرقم المرسل له و الذي قام بفك تشفيره، إذا تطابق الرقمان فهذا يعني صحة الرسالة و صحة مصدرها ، أما إذا اختلف الرقمان فهذا يعني تعرض الرسالة للتعديل بشكل غير شرعي أثناء عملية نقلها عبر الشبكة أو قناة الاتصال.

9.1 الأعداد العشوائية و الأعداد شبه العشوائية (Pseudorandom):

تستخدم الأعداد العشوائية أو شبه العشوائية (أي التي تم توليدها آليا باستخدام الحاسوب و يفترض أن تكون عشوائية لا تتكرر) في عملية توليد المفاتيح السرية التي تستخدم في عملية التشفير و فك التشفير. في خوارزميات التشفير القياسية المختلفة ، يلعب المفتاح دورا مركزيا حيث تعتمد عملية التشفير و تأمين المعلومات على المفتاح المستخدم حيث أن الخوارزمية و تفاصيلها علنية و معروفة لدى الجميع أما المفتاح فيجب أن يبقى سريا ؛ و يجب أن يكون المفتاح بالطول المناسب و العشوائية المطلوبة. العشوائية هنا أي أن احتمال التكرار يجب أن يكون نادرا جدا ، لماذا؟ ، حتى يصعب جدا على المخترق تخمين المفتاح الصحيح.

الطول المناسب يختلف باختلاف الخوارزمية المستخدمة و نوع التشفير (متناظر أو غير متناظر) لكن عموما كلما كان المفتاح أطول كلما كان أفضل لأن طول المفتاح يجعل عدد المفاتيح التي يجب أن يقوم المخترق بتجربتها كبيرا جدا (و لاتنس أن التطور الكبير – عبر السنوات – في مجال تقنية المعلومات سواء من حيث العتاد أو من حيث البرمجيات جعل عملية البحث أسرع بكثير من السابق – قانون مور المشهور: قوة الحواسيب تتضاعف مرة كل 18 شهرا تقريبا).

إذا افترضنا أن لدينا مفتاح عشوائي (أو شبه عشوائي) بطول 56 بتا ، يجب على المخترق إجراء 2^{56} (آلتي الحاسبة تقول أن هذا العدد تقريبا = 72057594037927936 !) عملية بحث ضمن مجموعة المفاتيح (key space) بنفس الطول للوصول للمفتاح الصحيح، هذا سيجعل هذه العملية صعبة جدا و مكلفة للغاية و تحتاج للكثير من الوقت ؛ أحد العلماء[1] في عام 1995م قام بتصميم آلة لإيجاد المفاتيح بطول 56 بتا (خوارزمية DES) هذه الآلة تكلف مليون دولار! – كان ذلك قبل عدة سنوات و لاشك ستكون أرخص كثيرا الآن – ؛ هذه الآلة يمكنها ايجاد المفتاح خلال 7 ساعات فقط!! (لذلك يقوم علماء التشفير و المؤسسات التي تضع المعايير بتغيير الخوارزميات المعتمدة كل عدة سنوات، و خوارزميات التشفير القياسية المعتمدة حاليا تستلزم مفاتيح ذات أطوال ما بين 128 إلى 256 بتا – أو أعلى – في التشفير المتناظر و 1024 إلى 3072 بتا – أو أعلى – في التشفير غير المتناظر).

هذه الطريقة في تجربة المفاتيح واحدا بعد آخر تسمى عند محللي و كاسري الشفرات باسم هجوم القوة العنيفة (Brute Force Attack) ، و هي من أشهر طرق تحليل و كسر التشفير (Cryptanalysis)،

(1) انظر لذلك كتاب Schneier: التشفير التطبيقي Applied Cryptography الباب السابع.

و من الواضح أنه كلما كان المفتاح طويلا كلما صعب الوصول إليه باستخدام هذه الطريقة و أصبح ذلك مكلفا جدا.

قام Schneier بوضع جدول لخص فيه أرقام تقريبية لتكلفة ايجاد مفاتيح متغيرة الطول (مابين 40 إلى 128 بتا) ، من حيث الوقت اللازم للتوصل لها (مابين أجزاء من الثانية إلى ملايين السنين !) و المبلغ الذي ستكلفه هذه العملية (من عدة آلاف إلى ملايين أو مليارات الدولارات !) و ذلك باستخدام هجوم القوة العنيفة، إلا أنه أشار إلى أن التطور الهائل و السريع في مجال تقنية المعلومات و أجهزة و برمجيات الحاسوب يجعل هذه الأرقام مجرد تخمينات غير مؤكدة – خاصة مع مرور الزمن !. إلا أن المعلومة المستفادة من هذا الجدول هي أنه كلما كان المفتاح طويلا كلما كان اكتشافه اكثر كلفة من حيث الوقت و المال و الجهد ؛ لذلك سيسأل من سيقوم بذلك نفسه : هل تستحق هذه المعلومات المراد كسر تشفيرها كل هذا الجهد ؟ و هل سينفع الوصول لهذه المعلومات بعد انقضاء الفترة اللازمة لكسر تشفيرها أم أن الأوان سيكون قد فات ؟.

و السؤال الذي يرد هنا : إلى أي مدى يجب أن تبقى هذه المعلومات سرية ؟ لا شك أن الاجابة تختلف من حالة لأخرى و من شخص – أو جهة – لآخر ؛ فهناك أسرار المطلوب المحافظة عليها لفترة قصيرة نسبيا – ساعات أو أيام – مثل تفاصيل استراتيجيات العمليات العسكرية في الحرب القائمة بين دولتين مثلا ، و هناك أسرار يجب المحافظة عليها لزمن أطول لعدة سنوات مثل الأسرار الدبلوماسية للدولة أو أسرار المنتجات و الاختراعات التجارية للشركات و المؤسسات التجارية و المعلومات الشخصية الحساسة مثل أرقام و تفاصيل الحسابات البنكية و ما شابه ذلك، و هناك أسرار أخرى يجب إبقاؤها طي الكتمان لفترات طويلة جدا – عدة عقود – مثل أسرار الدول العليا و المعلومات السرية المتعلقة بعالم المخابرات و الجاسوسية و ما شابهها !.

هناك أمر آخر مهم لابد من ذكره هنا، و هو أن تغيير المفاتيح بشكل دوري سيرفع من مستوى الأمان ؛ إذا تم اكتشاف المفتاح – أو سرقته – فلن تنكشف كل المعلومات السرية و إنما جزء منها فقط مما سيقلل الخسائر ، كما أن استخدام أنظمة تشفير هجينة ؛ أي تستخدم التشفير المتناظر و غير المتناظر معا ، سيرفع مستوى الأمان بشكل أكبر لأنه سيجعل عمل المخترق شاقا و مكلفا جدا،

10.1 دراسة الحالة: نظام لإجراء المحادثات الآمنة عبر الشبكة:

سيقوم الباحث بتطبيق مفاهيم و جوانب علم التشفير المختلفة و توظيفها لتطوير برنامج دردشة لإجراء المحادثات بشكل آمن عبر شبكة الانترنت ، مع الالتزام ببروتوكولات و خوارزميات التشفير القياسية التي تقدم مستوىً عال من الأمان.

في البداية يتم تقديم دراسة مختصرة حول بعض برامج الدردشة المشهورة – سكايب مثلا – و محاولة دراسة مستوى الأمان الذي تقدمه و بروتوكولات و خوارزميات التشفير التي تستخدمها، ثم تحديد المشكلة و توصيف الحل المقترح و تحديد المتطلبات.

ثم يقوم الباحث بتصميم و تطوير مكتبة برمجية للربط الديناميكي (DLL) باسم (CryptoEngine) تضم بعض خوارزميات التشفير القياسية المهمة ؛ تقدم هذه المكتبة خدمات التشفير المتنوعة لأي برنامج يقوم باستخدامها ، مع مراعاة أن يتم تصميمها بشكل جيد و سهل الفهم و التطبيق، مع الالتزام بخوارزميات التشفير القياسية الحديثة، على أن تكون هذه المكتبة متوافقة مع المكتبات البرمجية التي تقدمها أداة (Qt-SDK) – و هي أداة مفتوحة المصدر و واسعة الاستخدام لتطوير برمجيات متعددة المنصات بلغة C++ ؛ إلا أنها لا تضم مكتبة شاملة للتشفير[1] ؛ على أن يتم اجراء عدة اختبارات على المكتبة للتأكد من مستوى التشفير الذي تقوم بتوفيره و مطابقته للمعايير[2] .

ثم بعد الانتهاء من اعداد المكتبة و اختبارها يتم تصميم و تطوير برنامج دردشة باسم (زاجل) يوفر هذا البرنامج لمستخدميه خدمة اجراء المحادثات و تبادل البيانات بشكل آمن و موثوق عبر الانترنت ؛ حيث يستخدم البرنامج المكتبة المذكورة لتشفير/فك تشفير الرسائل و التأكد من صحتها و صحة هوية مرسلها. يلي ذلك اجراء تجارب مكثفة على البرنامج للتأكد من تحقيقه لجميع المتطلبات و أهمها هو تقديم مستوى عال و قياسي من الأمان عند اجراء المحادثات عبر الشبكة.

(1) و ذلك حتى تاريخه – على حسب اطلاع الباحث.

(2) و يتم التأكد من ذلك بشكل مزدوج حيث سيتم تطوير المكتبة بناء على مكتبة أخرى مفتوحة المصدر باسم Crypto++ وهي مكتبة تم اعتمادها من معهد NIST بأنها تحقق خوارزميات التشفير القياسية، انظر الاصدارة FIPS-140-2، كما سيتم اجراء الاختبارت اللازمة.

11.1 الخلاصة – هل يوفر التشفير الأمن المطلوب ؟ :

بعد استعراض مختلف المواضيع المتعلقة بعلم التشفير بشكل سريع و مبسط – قدر الامكان – يبقى هذا السؤال : هل يوفر التشفير الأمن المطلوب ؟. علم التشفير هو أحد أهم أركان أمن المعلومات و لكنه ليس الركن الوحيد فهناك عوامل أخرى لها تأثير كبير، إلا أن هذا البحث ليس مخصصا للحديث عنها ؛ بالنسبة للتشفير فليس هناك ضمان – كما يقول علماء و خبراء التشفير - بنسبة 100% و لم يستطع أحد تقديم إثبات لذلك المستوى من الضمان[1] حتى الآن، و لكن يمكن للتشفير – بالرغم من ذلك - أن يقدم مستوى عال و مقبول من الأمان بشرط استخدامه بالشكل الصحيح و الشروط المختلفة التي قام العلماء بتقديم بحوث متنوعة عنها ؛ و باستخدام البروتوكولات و الخوارزميات القياسية الموثوقة و تطبيقها بشكل سليم.

(1) انظر الباب الأول من كتاب : Handbook of Applied Cryptography تأليف مينيزيس و آخرين.

الباب الثاني

التحليل و تحديد المتطلبات

1.2 تمهيد:

يتناول الباحث في هذا الباب موضوع التحليل و تحديد المتطلبات ، و ذلك بغرض تحديد مشكلة البحث و مناقشة بعض الدراسات السابقة في الموضوع بغرض الوصول للحل المقترح ثم تحديد المتطلبات الواجب توفرها في الحل المقترح مع اجراء دراسات الجدوى اللازمة و توضيح حالات الاستخدام ؛ و لكن قبل كل ذلك لابد من ذكر بعض خوارزميات التشفير القياسية المهمة و التي ستمثل حجر الزاوية في عملية تأمين الاتصال و المحادثات و التحقق من صحتها و صحة مصدرها ؛ هذه الخوارزميات سيأتي ذكرها في هذا الباب و الأبواب التالية لذلك كان مناسبا وضع مدخل مختصر حولها في بداية هذا الباب.

تنبيه مهم: *يرغب الباحث بتوضيح أن أسماء المنتجات البرمجية التي سيتم تناولها في هذا الباب أو في أي أبواب أخري ضمن هذا البحث هي علامات تجارية معروفة و مسجلة لمالكيها ، والباحث يحترم هذه العلامات ويقر بها لماليكها سواء كانوا أفرادا أو شركات أو أي جهات تنظيمية أخرى ؛ و لم يتم ذكرها بالتفصيل بغرض الاختصار لا غير ؛ كما أن دراسة و تحليل مستوى الأمان لهذه المنتجات يتم لأغراض الدراسة الأكاديمية الجامعية البحتة بواسطة الباحث (و هو طالب في مرحلة التعلم) و ليس لأي غرض آخر ؛ و لا يقصد الباحث من ذلك انتقاص أو تقليل قيمة هذه المنتجات أو الجهات المالكة أو المشغلة لها بأي شكل من الأشكال.*

2.2 مدخل إلى خوارزميات التشفير القياسية:

سيتم استعراض مجموعة من خوارزميات التشفير القياسية بشكل مختصر، تتضمن نوعي التشفير – متناظر و غير متناظر، مما يوفر للقارئ مدخلا سريعا لفهمها.

1.2.2 خوارزمية معيار تشفير البيانات (DES):

هذه الخوارزمية هي أول و أشهر معيار عالمي للتشفير و ما زالت تستخدم بشكل كبير حتى اليوم، و هي و إن كانت غير مستخدمة في هذا البحث إلا أنه سيتم الحديث عنها هنا – باختصار – لأهميتها و لدورها في تطوير كثير من الخوارزميات التالية لها ؛ حيث أنها تمثل مدخلا لفهم كل خوارزميات التشفير الحديثة التي جاءت بعدها.

- مدخل تاريخي:

تم اصدار هذه الخوارزمية في 76-1977م بواسطة المعهد القومي للمعايير و التكنولوجيا (NIST) – و كان يسمى حينها بـ(NBS) ؛ و اعتمادها كمعيار عالمي للتشفير ؛ و منذ ذلك الوقت أصبحت تستخدم بشكل واسع حول العالم. هذه الخوارزمية في الأساس مبنية على خوارزمية تشفير باسم (Lucifer) قامت بتطويرها شركة (IBM) بواسطة مجموعة من خبرائها في أوائل سبعينيات القرن الماضي؛ حيث كانت هذه الخوارزمية – أي Lucifer – هي الخوارزمية المختارة من مجموعة من الخوارزميات المقترحة.

- من أهم الشروط التي تم على أساسها اختيار الخوارزمية:

- توفير مستوى عالٍ من الأمان.
- الأمان الذي توفره الخوارزمية يجب أن يعتمد على المفتاح و ليس على سرية الخوارزمية (انظر الفقرة 7.1 في الباب السابق).
- يجب أن تكون الخوارزمية قابلة للتنفيذ العملي على الأجهزة الالكترونية بدون تكلفة عالية و تتميز بالكفاءة في الاستخدام و يمكن اختبارها و تصديرها و أن تكون متاحة لجميع المستخدمين.

- وصف مختصر للخوارزمية:

خوارزمية (DES) هي خوارزمية للتشفير المتناظر، و هي من نوع تشفير الكتلة (Block Cipher)؛ أي يتم تقسيم النص المراد تشفيره إلى مجموعة من الكتل بطول ثابت ، في هذه الخوارزمية كل كتلة بطول 64 بتا ، ثم يتم تشفير كل كتلة بالمفتاح – و هو بطول 64 بتا (و هو في الواقع بطول 56 بتا و البتات الثمانية الاضافية تستخدم للتحقق من التماثل parity) و ينتج عن عملية التشفير كتل من النص المشفر كل كتلة بطول 64 بتا يتم ضمها معا لينتج لنا النص المشفر ؛ و يعتمد الأمان بالكامل على المفتاح.

تضم هذه الخوارزمية 16 دورة (Round) في كل دورة يتم اجراء عمليتي احلال (Substitution) و ابدال – تقليب- (Transposition / Permutation) على النص (الكتلة) المراد تشفيره باستخدام المفتاح. أي يتم تكرار عمليتي الاحلال و الابدال 16 مرة على النص المراد تشفيره.

ثم ظهرات أنواع أخرى من نفس الخوارزمية من أشهرها Triple DES و تستخدم فيها 3 مفاتيح حيث تكرر العملية 3 مرات كل مرة بمفتاح مختلف، مما يرفع مستوى الأمان بشكل كبير ؛ إلا أن من أهم عيوبها أنها بطيئة جدا بالمقارنة مع خوارزمية DES الأصلية.

2.2.2 خوارزمية معيار التشفير المتقدم (AES):

- مدخل تاريخي:

تم اعتماد و نشر هذه الخوارزمية في عام 2001م بواسطة معهد NIST كبديل لخوارزمية DES ؛ و ذلك لظهور قصور في مستوى الأمان الذي تقدمه الخوارزمية السابقة نسبة للتطور الكبير في مجال تقنية المعلومات و الحاسوب (و تقنيات تحليل و كسر الشفرات) خاصة مجال الحوسبة المتوازية أو الموزعة. تمكنت مؤسسة تعمل في أمن المعلومات – في العام 1999م في أحد المؤتمرات العلمية – من كسر نص مشفر بخوارزمية DES (أي تمكنت من الوصول للمفتاح عن طريق هجوم القوة العنيفة – انظر الباب السابق فقرة 9.1 – و بالتالي فك التشفير) في ساعات قليلة باستخدام الحوسبة المتوازية و السرعة الكبيرة التي توفرها المعالجات الحديثة.

في عام 1997م تم الاعلان – بواسطة معهد NIST – عن منافسة لاختيار خوارزمية جديدة أكثر أمانا بحيث تدعم استخدام مفاتيح متغيرة الطول و تقدم مستوى أكثر أمانا مع سرعة أعلى من سابقتها ، تم اختيار أفضل 5 خوارزميات في 1999م ثم في عام 2000م تم اختيار خوارزمية اسمها (Rijndael) و تم اعتمادها في العام 2001م كمعيار عالمي معتمد للتشفير باسم AES.

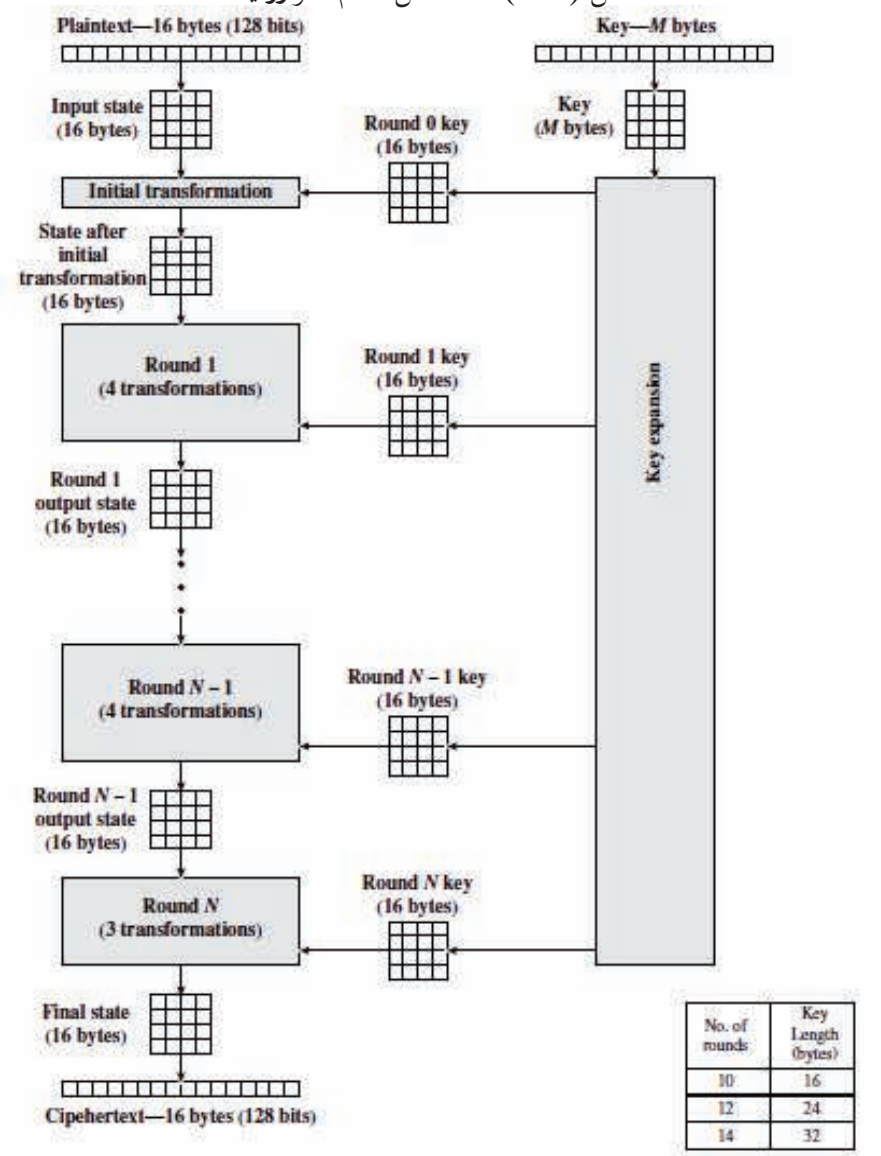
- وصف مختصر للخوارزمية :

هذه الخوارزمية أيضا من نوع التشفير المتناظر و تستخدم تشفير الكتلة ، حيث يتم تقسيم النص المراد تشفيره إلى مجموعة من الكتل طول كل كتلة 128 بتا (أي 16 بايتا) و يتم التشفير باستخدام مفتاح متغير الطول ؛ حيث يمكن أن يتم استخدام مفتاح بطول 128 أو 192 أو 256 بتا (أي 16 أو 24 أو 32 بايتا).

تستخدم هذه الخوارزمية عددا من الدورات (Rounds) يعتمد على طول المفتاح (10 أو 12 أو 14 دورة إذا كان المفتاح بطول 128 أو 192 أو 256 بتا) إلا أن دوراتها أكثر تعقيدا من دورات خوارزمية DES بكثير (مما يجعلها أكثر أمانا منها) ؛ حيث أن كل دورة (ما عدا الأخيرة) تضم داخلها أربع عمليات مختلفة : الاحلال و الابدال – التقليب – و عملية احلال مبنية على عملية حسابية باستخدام نظرية المجال المحدود (Finite Field) و عملية (XOR) و كل ذلك باستخدام المفتاح الذي يتم توليد مفاتيح فرعية منه ؛ أي مفتاح فرعي لكل دورة. الشكل (2.1) يوضح الشكل العام للخوارزمية ؛ حيث يتم وضع كل كتلة من النص المراد تشفيره على شكل مصفوفة 4 X 4 أي 16 بايتا بحيث تحوي الخلية الواحدة في المصفوفة على بايت واحد، و تسمى هذه المصفوفة بالحالة State ؛ هذه الحالة تمر من دورة لأخرى لإجراء

العمليات الأربع – المذكورة آنفا – عليها باستخدام المفتاح الفرعي الخاص بكل دورة. و يتم فك التشفير بطريقة عكسية باستخدام نفس المفتاح.

شكل (2.1) : الشكل العام لخوارزمية AES



المصدر: William Stallings, Cryptography and Network Security.

من أهم مميزات هذه الخوارزمية أنها تقبل مفاتيح مختلفة الطول و هذه المفاتيح أطول من مفتاح خوارزمية DES ؛ كما أنها توفر مستوى أعلى من الأمان و توفر سرعة عالية في التشفير.

3.2.2 خوارزمية (RSA) للتشفير بالمفتاح العلني (غير المتناظر) :

- مدخل تاريخي:

خوارزمية RSA هي أشهر خوارزميات التشفير غير المتناظر – التشفير بالمفتاح العلني – (انظر الباب السابق فقرة 5.1 و ما يليها) و أكثرها استخداما حيث تستخدم في التشفير و في التوقيع الرقمي – و هذا هو الأغلب – ؛ في 77- 1978م تم اختراع و تطوير هذه الخوارزمية على يد ثلاثة من العلماء (Rivest, Shamir and Adleman) في معهد MIT في أمريكا؛ و تم تسميتها بالحروف الأولى من أسمائهم.

هذه الخوارزمية من أفضل خوارزميات التشفير بالمفتاح العلني حيث تتميز بسهولة الفهم و سهولة التطبيق (برمجيا و عتاديا) و تم اختبارها و التأكد منها لفترة طويلة بواسطة الكثير من الخبراء و من محللي و كاسري الشفرات ؛ و تستخدم بشكل واسع في اصدار التواقيع الرقمية و في تبادل مفاتيح الجلسات المستخدمة في تأمين الاتصال و تبادل المعلومات عبر الشبكة. كما تم اعتمادها كأحد خوارزميات التوقيع الرقمي المعيارية في عام 2009م بواسطة مؤسسة NIST (على أن يكون العدد n – انظر التالي عنه - بطول 3072 بتا على الأقل) ، إلا أنها تستخدم بشكل واسع منذ فترة طويلة قبل ذلك و إلى الآن.

- وصف مختصر للخوارزمية:

تقوم هذه الخوارزمية على أساس صعوبة ايجاد العوامل الأولية (prime factors) للأعداد الصحيحة غير الأولية (numbers nonprime) الضخمة جدا، و هي معضلة رياضية قديمة و مشهورة ؛ افترض أن لدينا عددين أوليين ضخمين و ذوا طول واحد هما p و q (ضخمين بمعنى كل واحد منهما قد يبلغ طوله 100 أو 200 خانة أو أطول) و أن حاصل ضربهما = n و هو رقم ضخم جدا و طويل جدا ؛ من الصعب جدا إذا عرفت قيمة n فقط أن تقوم بمعرفة (أي حسابيا) العددين الأوليين p و q اللذان استخدما في توليد هذا الرقم الصحيح الضخم. يتم توليد رقم أولي آخر يسمى e و هو مع الرقم n يمثلان المفتاح العلني ؛ كما يتم توليد رقم آخر d (باستخدام خوارزمية اقليدس الممتدة لايجاد القاسم المشترك الأعظم و هي خوارزمية مشهورة) و هو المفتاح الخاص $[d = e^{-1} \bmod ((p - 1)(q - 1))]$ ؛ و بالطبع يجب ابقاؤه سريا ؛ الأرقام الأولية p و q يجب تدميرها – لعدم الحاجة لهما بعد توليد المفاتيح – أو ابقاؤهما سريين.

للتمثيل لذلك بمثال مبسط[1] ؛ نأخذ العددين الأوليين p=47 ؛ و q=71 ؛ حاصل ضربهما n=3337 ؛ إذا كان (p-1) = 46 و (q-1) = 70، فإن حاصل ضربهما = 3220 ؛ الرقم e = 79 ؛ و بالتالي فالرقم d سيتم توليده باستخدام خوارزمية اقليدس الممتدة $[d=79^{-1} \bmod 3220]$ = 1019 ؛ الناتج:

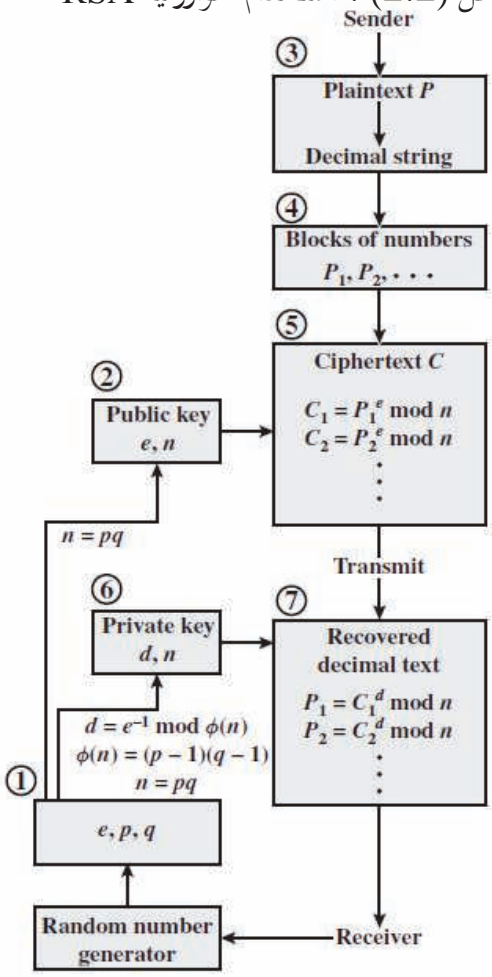
(1) انظر كتاب Schneier: التشفير التطبيقي Applied Cryptography الباب التاسع عشر.

المفتاح العلني هو الزوج (e=79 و n=3337) و المفتاح الخاص (السري) هو d=1019 ، للتشفير تستخدم المعادلة $[c=m^e \bmod n]$ حيث أن m ترمز للنص المراد تشفيره و c ترمز للناتج أي النص المشفر؛ أما فك التشفير فيتم عن طريق المعادلة $[m=c^d \bmod n]$

الشكل رقم (2.2) يوضح الشكل العام لاستخدام هذه الخوارزمية من خلال الخطوات التالية:

1- توليد الأرقام الأولية العشوائية p و q و e – و ذلك بواسطة المستلم للرسالة (أي المرسل له).
2- توليد المفتاح العلني (e و n) و نشره.
3- تحويل النص المراد تشفيره لسلسلة من الأرقام الصحيحة – و يقوم بذلك مرسل الرسالة.
4- تقسيم هذه السلسلة إلى كتل من الأرقام ذات طول ثابت.
5- تشفير هذه الكتل باستخدام المفتاح العلني الخاص بالمستلم و ارسالها له.

شكل (2.2) : استخدام خوارزمية RSA



المصدر: Menezes, et Al, Handbook of Applied Cryptography.

6- المفتاح الخاص d و يحتفظ به المستلم سريا، كما أنه يحتفظ بالرقم n بالطبع.

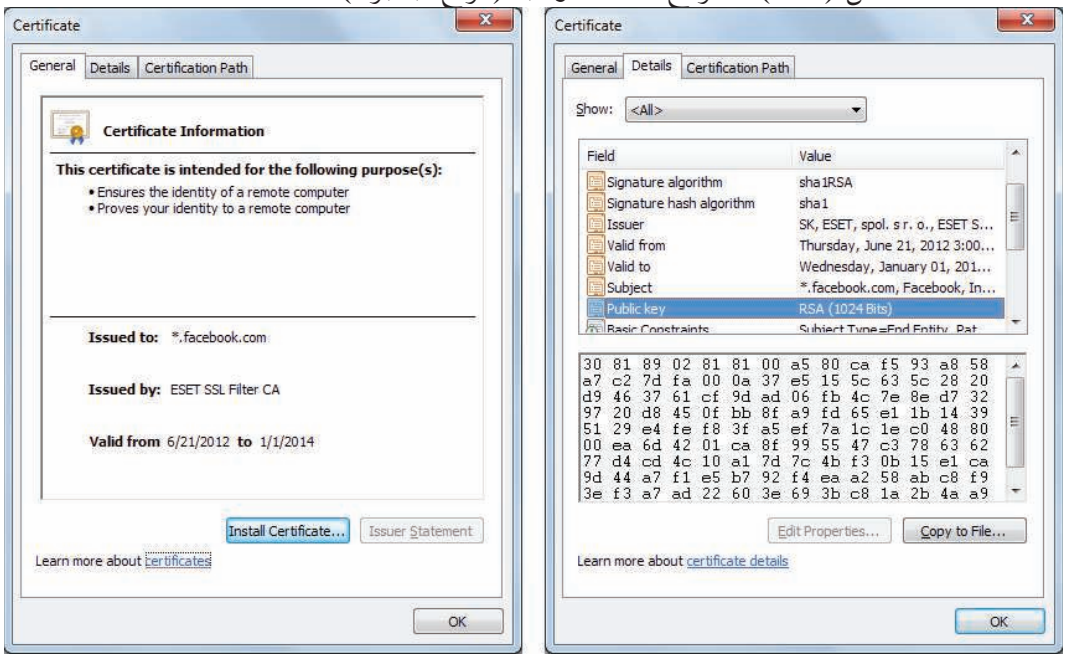
7- فك تشفير الرسالة بواسطة المستلم باستخدام مفتاحه الخاص.

أهم عيوب هذه الخوارزمية – و خوارزميات التشفير غير المتناظر عموما – أنها تستخدم مفاتيح طويلة جدا مما يجعل عملية التشفير تستغرق وقتا طويلا خاصة بحسب طول البيانات ؛ بالاضافة إلى أنها قد تكون أكثر أمانا عند تشفير نصوص طولها أقل من طول المفتاح العلني ؛ لذلك تستخدم الخوارزمية في تشفير مفاتيح الجلسات لتبادلها بشكل آمن و تستخدم كذلك في تطبيقات التوقيع الرقمي و الشهادات الرقمية المتنوعة.

- **الشهادات الرقمية (Digital Certificate):**

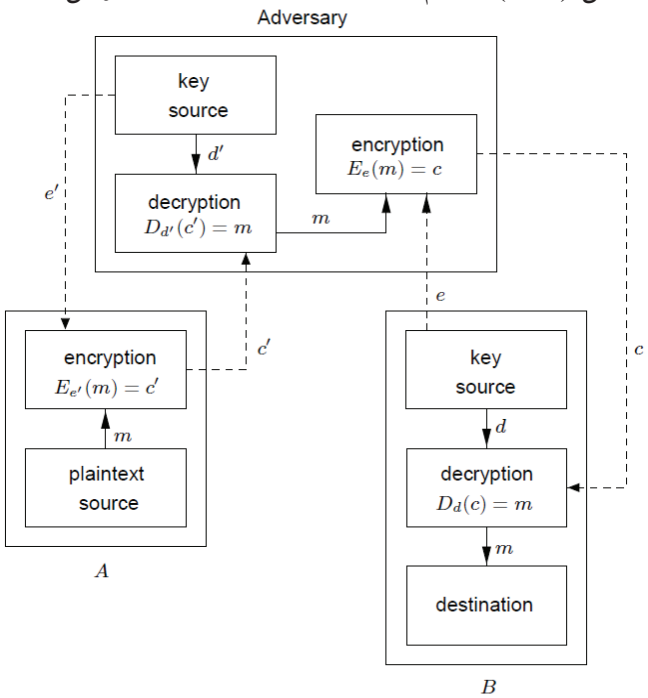
يتم اصدارها بواسطة شركات متخصصة و معروفة (أي طرف ثالث موثوق TTP) ؛ الشكل (2.3) يوضح نموذجا لشهادة رقمية لموقع فيسبوك ؛ تم اصدارها بواسطة شركة ESET لاثبات أن هذا المفتاح العلني هو ملك لفيسبوك و يلاحظ أيضا أن هناك تاريخ اصدار و تاريخ انتهاء للشهادة بالاضافة للمفتاح العلني لفيسبوك و الكثير من التفاصيل الأخرى المهمة.

شكل (2.3) : نموذج لشهادة رقمية (موقع فيسبوك)



تثبت الشهادة الرقمية أن المفتاح العلني – الفلاني – هو مفتاح الشخص (أو الجهة) الفلاني ؛ أي يتم ربط بيانات مالك المفتاح (اسمه – عنوانه – الخ) مع مفتاحه العلني ؛ و ذلك بغرض حماية المستخدم من انتحال الشخصية الذي قد يقوم به المخترق (القرصان) ؛ أي قد يدعي القرصان أن مفتاحه العلني هو مفتاح شخص آخر أنت تريد ارسال رسالة آمنة له و بالتالي تستخدم مفتاح القرصان – بدلا عن مفتاح الشخص المقصود – في تشفير الرسالة مما يمكن المخترق من الاطلاع بكل سهولة على محتوى الرسالة عن طريق فك تشفيرها بمفتاحه الخاص ؛ ثم يعيد تشفيرها و يرسلها إلى وجهتها النهائية حتى لا يكتشفه أحد!– انظر شكل (2.4) ؛ و يسمى هذا النوع من الهجوم (الاختراق) بهجوم رجل في المنتصف (Man-In-The-Middle Attack) . كذلك للشهادات الرقمية وظيفة مهمة جدا في التحقق من صحة موقع الويب الذي تستخدمه حيث أن القرصان بإمكانه تصميم موقع مطابق للموقع المستهدف – موقع لتقديم الخدمات المصرفية مثلا – و يعطيه اسما شديد الشبه به حتى يخدع المستخدم فيعتقد أنه الموقع الصحيح و يقوم بادخال بيانات سرية مثل رقم الحساب المصرفي و الاسم و كلمة السر مما يمكن المخترق من استخدام هذه البيانات بشكل غير قانوني مما قد يلحق الضرر الكبير بصاحبها؛ لذلك يقوم أصحاب الموقع بوضع شهادة رقمية صادرة من مصدر معتمد على ملقماتها (سيرفراتها) حتى يمكن التحقق من أنه هو الموقع الأصلي.

شكل (2.4) : قيام المخترق بانتحال شخصية المرسل له



المصدر: Menezes, et Al, Handbook of Applied Cryptography.

4.2.2 خوارزمية الاختزال "الهاش" الآمن الاصدارة الثانية (SHA2):

دوال الاختزال "الهاش" لها دور مهم في التحقق من صحة المعلومات، أي سلامتها من التعديل أثناء نقلها عبر الشبكة (كما مر في الباب الأول) – أو بعد حفظها على القرص – ؛ و حتى تكون آمنة لابد أن تكون غير قابلة للعكس (Function One Way) أي لا يمكن معرفة البيانات الأصلية من رقم الهاش الفريد الذي قامت الخوارزمية بتوليده.

هناك الكثير من الخوارزميات المستخدمة و التي لا توفر جميعها المستوى المطلوب من الأمان ؛ لذلك تم تطوير هذه الخوارزمية بواسطة معهد NIST بالتعاون مع وكالة الأمن القومي NSA و سميت بخوارزمية الاختزال الآمن SHA و ذلك في العام 1993م لاستخدامها كمعيار فيدرالي في عملية التحقق من صحة المعلومات ؛ ثم تم اكتشاف قصور في الخوارزمية فتم تطوير اصدارة أخرى أكثر أمانا في العام 1995م و سميت SHA-1 و تقوم بتوليد رقم هاش بطول 160 بتا ؛ تم بناء هذه الخوارزمية بناء على خوارزمية أخرى اسمها MD4 تم تطويرها بواسطة Rivest في معهد MIT.

في العام 2002م تم اصدار نسخة جديدة من الخوارزمية و سميت SHA2 و يمكنها توليد أرقام متغيرة الطول، 256 أو 384 أو 512 بتا و تعرف أيضا بـ SHA256 و SHA384 و SHA512 ؛ و تم اعتمادها في العام 2005م كمعيار جديد للاختزال "الهاش" و في عام 2010م تم اعتبار خوارزمية SHA1 غير معيارية لاكتشاف فريق من الباحثين أنها تحتاج فقط لـ 2^{69} عملية حسابية لكسرها – و ليس 2^{80} ، كما كان الاعتقاد سائدا – و هو ما اعتبر غير آمن مع التطور الكبير في سرعة و أداء الحواسيب ، و بالتالي لابد من استخدام SHA2 لأنها أكثر أمانا منها بكثير.

ثم لاحقا في العام 2012م تم الاعلان عن منافسة لاختيار خوارزمية معيارية جديدة ؛ فازت في المنافسة خوارزمية (Keccak) في ديسمبر 2012م و سميت باسم SHA3 كمعيار جديد للاختزال.

من أهم استخدامات دوال الهاش – بالاضافة إلى ما سبق – استخدامها في تأمين كلمات السر المخزنة على قواعد البيانات ؛ حيث لا يتم حفظ كلمة السر الأصلية مباشرة داخل قاعدة البيانات لأنه يمكن كشفها عن طريق مستخدمين آخرين (أو حتى عن طريق مدير النظام أو غيره) ؛ بل يتم استخدام خوارزمية الاختزال في توليد رقم هاش من كلمة السر و يخزن رقم الهاش بدلا عن كلمة السر الأصلية ؛ بالتالي حتى و لو تم كشف رقم الهاش لا يمكن معرفة كلمة السر الأصلية من خلاله. عندما يتم ادخال كلمة السر يتم توليد الهاش و مقارنته مع الهاش المخزن، إذا تطابقا كانت كلمة السر التي أدخلها المستخدم هي

الكلمة الصحيحة و إلا فلا ؛ و لكن يجب الحرص على اختيار كلمات سر آمنه أي أكثر طولا و أكثر عشوائية (تضم حروف و رموز و أرقام و ليس لها معنى محدد) حيث أن المخترق بامكانه استخدام هجوم يسمى (Dictionary Attack) في هذا الهجوم يقوم المخترق بتجربة كلمات سر مخزنة عنده في قاموس يضم آلاف الكلمات و التعابير المتدولة ؛ حيث يقوم بتوليد الهاش من كلمة السر المأخوذة من القاموس و مقارنتها بالهاش المخزن على قاعدة البيانات ؛ إذا تطابقا فهذا يعني انكشاف كلمة السر ؛ إذا كانت كلمة السر طويلة و عشوائية و لا تضم كلمات مألوفة فلن يستطيع المخترق كشفها بهذه الطريقة.

كذلك من الاستخدامات المهمة لدوال الاختزال "الهاش" استخدامها في توليد مفاتيح التشفير عن طريق مولدات للأرقام شبه العشوائية (Pseudo Random Number Generator) هذه المولدات يتم توفيرها من خلال نظام التشغيل أو من خلال برنامج منفصل.

3.2 الدراسات السابقة :

- عنوان الدراسة: How safe is instant messaging? .
- تاريخ الدراسة: 2008/6/9م.
- الجهة التي قامت باجراء الدراسة: الكاتب و الصحفي Declan McCullagh و نشرت في موقع CNET.COM[1].

دراسة مثيرة للاهتمام، أعدها الكاتب المذكور أعلاه حول مستوى السرية و الأمان الذي توفره مجموعة من برامج التراسل الفوري (IM) ؛ قام الصحفي المذكور باعداد مجموعة من الأسئلة المهمة المتعلقة بالموضوع – أي استبيان حول الموضوع – و ارسالها إلى الشركات المنتجة و المشغلة لهذه البرامج ؛ هذه الأسئلة – بتصرف يسير – كما يلي:

– هل تستخدمون التشفير من أجل التحقق عندما يقوم المستخدم بتسجيل الدخول؟ – هل تستخدمون التشفير عند تبادل الرسائل؟ – هل تستخدمون تقنية تشفير غير قياسية و إذا كان كذلك فما هي التقنية المستخدمة؟ – هل تقومون بحفظ سجلات (أرشيف) على الملقم – السيرفر – لعمليات تسجيل الدخول و لمحتوى الرسائل التي يتبادلها المستخدمون؟، و لكم من الزمن يتم حفظ هذه السجلات؟ – هل تلقيتم أي

(1) http://news.cnet.com/8301-13578_3-9962106-38.html

أمر قضائي من المحكمة – أو من أي جهة قانونية أخرى – بكشف معلومات عن حسابات المستخدمين؟ (أي مثل الاسم و رقم IP و ما شابه) ، و كم عدد المرات التي تم فيها ذلك؟ – هل تلقيتم أي أمر قضائي من المحكمة – أو من أي جهة قانونية أخرى – باجراء عملية استراق للسمع أو تنصت مباشر (live interception or wiretap) للمحادثات؟ ، أي أن محتوى الاتصالات بين المستخدمين يتم تمريره مباشرة للجهة القانونية.

قام الكاتب بارسال هذا الاستبيان لمجموعة كبيرة من الشركات المعروفة و التي تقدم خدمة التراسل الفوري ، ثم قام بعرض و تلخيص النتائج كما في الجدول (2.1).

كما قام الكاتب بعرض الأسئلة و الأجوبة لهذه الشركات ؛ ولايتسع المجال هنا لعرض كل التفاصيل لذلك قام الباحث باختيار ثلاثة من أشهر هذه المنتجات (البرمجيات) لتلخيص ما توصل له الكاتب ثم مناقشة نتائجه و التعليق عليها.

من الجدير بالذكر هنا أن هذه الدراسة المذكورة تم اجراؤها في العام 2008م و بما أن مجال تقنية المعلومات و الحاسوب هو أسرع المجالات تطورا ، لذلك قد تكون هذه البيانات قديمة – نسبيا – و لا شك أن بعضها – على الأقل – قد تغير الآن، لذلك يرغب الباحث – و حسب الامكانات المتاحة – في اجراء اختباراته الخاصة على هذه المنتجات و ذلك لتحديد مستوى الأمان و السرية بها ؛ أي هل يستخدم البرنامج (أو الموقع) خوارزميات و بروتوكولات التشفير لتأمين الاتصالات و المحادثات؟ و إذا كان كذلك ؛ فما هي هذه الخوارزميات التي يستخدمها و هل هي معيارية (قياسية)؟ و ما هي أوجه القصور في هذه البرامج من حيث السرية و الأمان؟ ، سيحاول الباحث الاجابة على هذه الأسئلة – بقدر الامكان – في السطور التالية، و ذلك بعد تلخيص أهم ما جاء في هذه الدراسة.

هذه المنتجات الثلاث المشار اليها أعلاه هي خدمة الدردشة على موقع الفيسبوك Facebook Chat و برنامج سكايب Skype و برنامج ياهوو ماسنجر Yahoo Messenger ؛ حيث أن هذه البرامج – على حد علم الباحث – من أشهرها و أكثرها استخداما.

جدول (2.1) يوضح نتائج استبيان مقدم لمجموعة من شركات برامج التواصل الاجتماعي.

	Secure logging-in	Secure conversations	Logs kept of user logins	Logs kept of message content	For how long	Government wiretapping
AOL AIM	Yes	Yes	Yes	No	Won't say	Won't say
AOL ICQ	Yes	No	Yes	No	Won't say	Won't say
Facebook Chat[1]	No	No	Refused to answer	Refused to answer[2]	Refused to answer	Refused to answer
Google Talk	Yes	Yes[3]	Yes	No[4]	Four weeks	Won't say
IBM Lotus Sametime	Yes	Yes	Yes	Configurable	Configurable	N/A
Microsoft's Windows Live Messenger	Yes	No[5]	No	No	N/A	Won't say
Skype	Yes	Yes	Yes	No	"A short time"	Cannot comply with wiretaps[6]
Yahoo Messenger	Yes	No	Yes	No	As long as "necessary"	Won't say

[1] Over the course of a week, Facebook refused to reply to questions.
[2] Facebook has said both that chat history "is not logged permanently" and that it is archived for 90 days.
[3] Encryption is on by default for the downloadable client, off by default for the Web, and not supported with the Google Talk Gadget.
[4] Configurable: users can choose to log conversations in their Gmail chat archives if they wish.
[5] Conversations are unencrypted, but files exchanged via Windows Live Messenger are encrypted.
[6] Skype was the only IM Company that said it could not perform a live interception if presented with a wiretap request: "Because of Skype's peer-to-peer architecture and encryption techniques, Skype would not be able to comply with such a request."

- **أولا: خدمة الدردشة على موقع الفيسبوك:**

أفاد الكاتب – المشار إليه أعلاه – أن الشركة أبلغته برسالة بريد الكتروني مقتضبة أنها ترفض الإجابة على الأسئلة المذكورة!!. الجدير بالذكر هنا أن الكاتب أشار إلى أن خدمة الدردشة على الفيسبوك هي الأسوأ من حيث السرية و الأمان مقارنة ببقية المنتجات التي تقدم هذه الخدمة – و ذلك حتى تاريخه بطبيعة الحال –.

- **ثانيا: برنامج سكايب:**

– هل تستخدمون التشفير من أجل التحقق عندما يقوم المستخدم بتسجيل الدخول؟.

الاجابة: نعم.

– هل تستخدمون التشفير عند تبادل الرسائل؟.

الاجابة: نعم. معمارية سكايب تقوم على نموذج الند للند (peer-to-peer) و بالتالي لا يمكن تعقب أو التنصت على المحتوى أثناء النقل على الشبكة، باختصار تستخدم سكايب TLS[1] لحماية محتوى الرسائل من التنصت و استراق السمع.

– هل تستخدمون تقنية تشفير غير قياسية و إذا كان كذلك فما هي التقنية المستخدمة؟.

الاجابة: لا ؛ تستخدم سكايب تشفيرا قويا باستخدام AES – 256 بتا ؛ و يتم التحقق باستخدام التشفير بالمفتاح العلني PKI[2].

– هل تقومون بحفظ سجلات (أرشيف) على الملقم – السيرفر – لعمليات تسجيل الدخول و محتوى الرسائل التي يتبادلها المستخدمون؟ و لكم من الزمن يتم حفظ هذه السجلات؟.

الاجابة: لا يتم أرشفة أو تسجيل محتوى الرسائل. بما أن الملقمات تستخدم لتقديم بعض المنتجات مثل SkypeOut فيتم تسجيل بعض معلومات الاتصال مثل اسم المستخدم و رقم IP و ذلك لفترة قصيرة.

– هل تلقيتم أي أمر قضائي من المحكمة – أو من أي جهة قانونية أخرى – بكشف معلومات عن حسابات المستخدمين، و كم عدد المرات التي تم فيها ذلك ؟.

الاجابة: نعم. نحن نتعاون مع الوكالات القانونية في الحدود الممكنة قانونيا و تقنيا . (كم عدد المرات التي تم فيها ذلك): هذه معلومات سرية.

– هل تلقيتم أي أمر قضائي من المحكمة – أو من أي جهة قانونية أخرى – باجراء عملية استراق للسمع أو تنصت مباشر (live interception or wiretap) للمحادثات ، أي أن محتوى الاتصالات بين المستخدمين يتم تمريره مباشرة للجهة القانونية ؟.

الاجابة: نحن لم نتلقى أي أمر قضائي من المحكمة – أو من أي جهة قانونية أخرى – باجراء عملية استراق للسمع أو تنصت مباشر على اتصالات سكايب – إلى – سكايب ، و عموما لن تستطيع سكايب تنفيذ ذلك نظرا لمعمارية سكايب القائمة على نموذج الند للند و نظرا لتقنيات التشفير المستخدمة.

- **ثالثا: برنامج ياهوو ماسنجر:**

– هل تستخدمون التشفير من أجل التحقق عندما يقوم المستخدم بتسجيل الدخول؟.

الاجابة: عند استخدام الماسنجر على الويب أو على البرنامج الذي يتم تحميله ، نستخدم SSL[3] لحماية كلمة السر الخاصة بالمستخدم عند تسجيل الدخول.

– هل تستخدمون التشفير عند تبادل الرسائل؟.

(1) بروتوكول Transport Layer Security و هو حاليا الاصدار المعياري لتبادل البيانات بأمان على انترنت على مستوى طبقة النقل.

(2) Public Key Infrastructure مجموعة القواعد و السياسات و البرمجيات المستخدمة في ادارة الشهادات الرقمية.

(3) بروتوكول Secure Socket Layer و هو أيضا بروتوكول معياري مثل TLS إلا أن الأخير هو الأحدث.

الاجابة: لا يستخدم ياهوو ماسنجر التشفير عند تبادل الرسائل.

– هل تستخدمون تقنية تشفير غير قياسية و إذا كان كذلك فما هي التقنية المستخدمة؟.

الاجابة: لا نستخدم تقنية تشفير غير قياسية؛ نستخدم SSL المعياري عند تسجيل الدخول كما هو مذكور في اجابة السؤال الأول.

– هل تقومون بحفظ سجلات (أرشيف) على الملقم – السيرفر – لعمليات تسجيل الدخول و لمحتوى الرسائل التي يتبادلها المستخدمون؟ و لكم من الزمن يتم حفظ هذه السجلات؟.

الاجابة: ياهوو ماسنجر يتيح للمستخدم اختيار هذه الخاصية (أي ابقاء سجل للرسائل) إذا أراد ذلك، في حالة ماسنجر على الويب يتم ابقاء السجلات على ملقم ياهوو، أما بالنسبة للنسخة التي يتم تحميلها من الماسنجر يتم حفظ السجل على جهاز المستخدم.

– هل تلقيتم أي أمر قضائي من المحكمة – أو من أي جهة قانونية أخرى – بكشف معلومات عن حسابات المستخدمين، و كم عدد المرات التي تم فيها ذلك ؟.

الاجابة: ياهوو تستجيب للطلبات القانونية بما يسمح به القانون.

– هل تلقيتم أي أمر قضائي من المحكمة – أو من أي جهة قانونية أخرى – باجراء عملية استراق للسمع أو تنصت مباشر (live interception or wiretap) للمحادثات، أي أن محتوى الاتصالات بين المستخدمين يتم تمريرها مباشرة للجهة القانونية ؟.

الاجابة: ياهوو لا تناقش تفاصيل الأوامر القانونية ، ياهوو تستجيب للطلبات القانونية بما يسمح به القانون.

■ محاولة تحديد مستوى السرية و الأمان – و نوع التشفير المستخدم – في البرامج الثلاث المذكورة أعلاه و تحديد أوجه القصور بها (و ذلك حتى تاريخ كتابة هذا البحث):

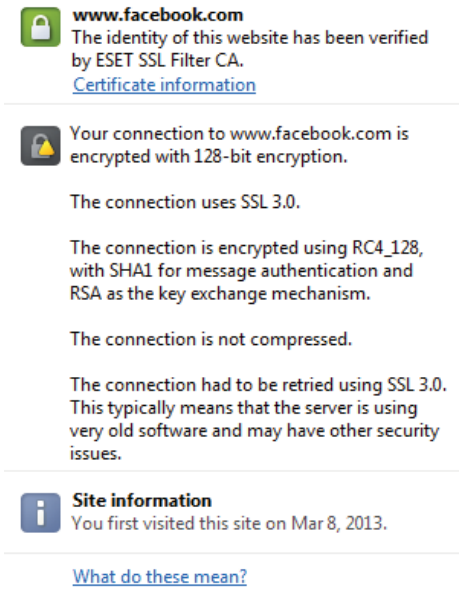
– **موقع فيسبوك** :

يستخدم موقع الفيسبوك التشفير عند تسجيل الدخول – يستخدم الباحث متصفح كروم من قوقل – ، انظر الشكل (2.5) أدناه ؛ من معلومات الشهادة الرقمية المرفقة مع الموقع يتبين أن الموقع يمتلك شهادة رقمية موقعة من ESET SSL Filter CA، و يتم تأمين الاتصال باستخدام بروتوكول SSL 3.0 ؛ و يتم التشفير باستخدام خوارزمية RC4_128 ؛ و SHA1 للتحقق من صحة الرسالة ؛ و خوارزمية RSA

لتبادل المفاتيح، و من الواضح أن تقنية SSL ليست هي الأحدث- المعيار الحيالي هو TLS - كما تمت الاشارة لذلك في الشكل (2.5).

بعد ذلك يتم تصفح الموقع و اجراء المحادثات بدون تشفير ، إلا أنه يمكن للمستخدم تغيير اعدادات الأمان لديه لتصفح الموقع بشكل آمن باستخدام خوارزميات التشفير المذكورة أعلاه ؛ و يبدو أن هذه الاضافة لم تكن متوفرة عندما قام الكاتب المذكور باعداد تقريره حول الموضوع. و يبقى السؤال عن مدى الأمان عند تصفح و اجراء المحادثات عبر موقع الفيسبوك باستخدام أجهزة الموبايل (الهاتف النقال) – الملاحظ أن الكثير من المنتجات التي تستخدم التشفير لتأمين الاتصال و المحادثات باستخدام الحاسوب لا توفر ذلك عند تقديم الخدمة بواسطة الموبايل و لكن ربما يتغير هذا الآن نسبة للتطور الكبير في الأجيال الجديدة في أجهزة الموبايل.

شكل (2.5) تقرير عن الشهادة الرقمية لموقع فيسبوك عند تسجيل الدخول باستخدام متصفح كروم



يمكن تلخيص أهم نقاط القصور في مستوى الأمان الذي تقدمه فيسبوك فيما يلي:

1. تقنية التشفير المستخدمة عند تسجيل الدخول – و إن كانت جيدة – ليست هي الأحدث (الأفضل).
2. لا يتم استخدام تقنيات التشفير عند استخدام الموقع (و اجراء الدردشة) بشكل افتراضي (أي تلقائي بدون طلب المستخدم) ؛ و لا يتم ذلك إلا إذا قام المستخدم بتغيير الاعدادات على حسابه لتنشيط

خدمة التصفح الآمن للموقع. وجه القصور هنا أن بعض المستخدمين قد لاينتبه لذلك مع حاجته لإجراء المحادثات بشكل آمن مما يمثل تهديدا لمستوى الأمان.

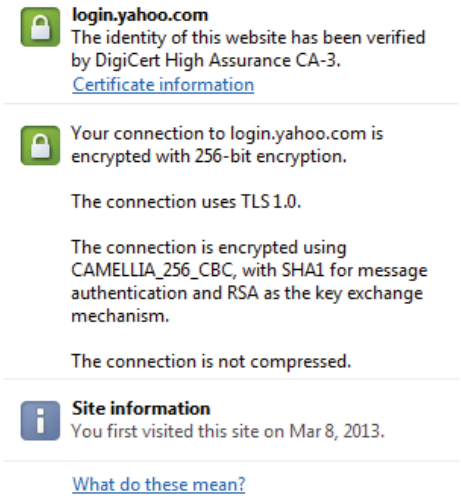
3. تقوم فيسبوك بحفظ سجل للمحادثات (أي المحاثات النصية) على الملقم الخاص بها لفترة طويلة – عدة أشهر حسب تجربة الباحث الشخصية عند استخدام الموقع – و جه القصور هنا من ناحيتين ؛ الأولى: هذه السجلات قد تتعرض للقرصنة و بالتالي يتم كشفها ؛ و نحن لا نعلم هل تحفظ بشكل آمن و محمي أم لا. الناحية الأخرى: يمكن كشف هذه السجلات إذا تم تقديم طلب قانوني من الجهات القانونية الأمريكية – أو غير الأمريكية – بذلك مما سيلزم فيسبوك بكشفها و هي معلومات خاصة قد لا يرغب صاحبها في كشفها مهما كانت الأسباب.

4. النقطة الأخيرة متعلقة بالنقطة السابقة ؛ هل فيسبوك ملزمة بكشف مفاتيح التشفير – أو اجراء عملية تنصت و استراق للسمع – إذا طلبت ذلك جهة قانونية ملزمة مثل وكالة الأمن القومي NSA أو غيرها من الوكالات الحكومية الأمريكية؟.

- **موقع ياهوو:**

عند تسجيل الدخول على البريد الالكتروني يتم استخدام التشفير لتأمين الاتصال – انظر شكل (2.6) أدناه، حيث يتم استخدام TLS 1.0 لتأمين الاتصال ؛ و يتم التشفير باستخدام خوارزمية CAMELLIA_256_CBC و التحقق من صحة الرسائل باستخدام SHA1 و يتم استخدام خوارزمية RSA لتبادل المفاتيح ، و ذلك كما هو موضح من الشهادة الرقمية الموقعة بواسطة DigiCert High Assurance CA-3. كما يوفر الموقع خيار استخدام SSL لتأمين تصفح الموقع و ينطبق على هذا الخيار نفس ما ذكرناه آنفا عند الكلام على موقع الفيسبوك.

شكل رقم (2.6) يوضح تقريرا عن الشهادة الرقمية لموقع ياهوو عند تسجيل الدخول.



- **برنامج سكايب :**

برنامج سكايب هو الأكثر أمانا – كما يبدو – حيث يتم استخدام خوارزمية AES المعيارية مع مفتاح بطول 128 بتا لتسجيل الدخول و 256 بتا عند تبادل الرسائل (حسب ما ذكرته سكايب) – و هو أطول مفتاح تسمح به الخوارزمية – انظر الشكل (2.7) أدناه الذي يوضح تفاصيل مستوى الأمان عند تسجيل الدخول الآمن لموقع سكايب.

لكن المشكلة أن الحكومة الأمريكية (و كالة الأمن القومي NSA مثلا) تلزم جميع الشركات – الأمريكية بالطبع – التي تقدم خدمات الاتصالات – بما فيها الشركات التي تقدم خدمات المحادثة و تبادل الملفات عبر انترنت – تلزمها بتمكينها من الاطلاع على محتوى هذه الاتصالات في حالة كانت تجرى بواسطة شخص – أو جهة – مصنفة بأنها تهدد الأمن القومي الأمريكي ؛ و لا ينفع إدعاء سكايب أن هذا – عمليا – لا يمكن تطبيقه ، حيث يمكن بسهولة أن تقوم سكايب باعطاء مفاتيح التشفير و عناوين IP الخاصة بالمستخدمين – بما أنها مخزنة على الملقم – للجهة الحكومية التي تطلبها – حسب قانون الأمن القومي – لتمكنها من الاطلاع على محتوى المحادثات.

و في الواقع واجهت سكايب كثيرا من الانتقادات و الاتهامات[1] بوجود ثغرات أمنية فيه يمكن تلخيص جزء منها هنا:

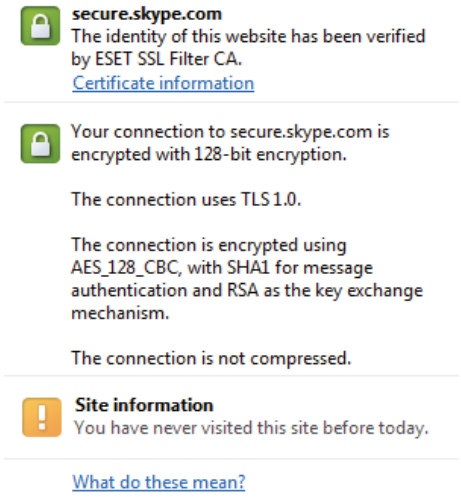
(1) انظر مقالة بعنوان Skype_security على موقع www.wikipedia.org.

– في 13 نوفمبر 2012م قام مستخدم روسي بكشف ثغرة أمنية في سكايب يمكن من خلالها الاستيلاء على حساب أي مستخدم لسكايب إذا تم معرفة بريده الالكتروني فقط في سبع خطوات بسيطة!. احتاجت سكايب لعلاج هذه الثغرة لمدة 12 ساعة و لكن يفترض أنها كانت موجودة قبل الاعلان عنها لفترة طويلة.

– ظهور عدة تقارير – *غير رسمية* – تتهم سكايب بوجود باب خلفي (back door) في البرنامج مما يمكن من التنصت و استراق السمع للمحادثات التي تتم باستخدام سكايب ؛ بل في يونيو 2008م صرح مصدر رفيع في وزارة الداخلية النمساوية – *تصريح غير رسمي* – بتمكنهم من التنصت على محادثات تتم بواسطة سكايب ؛ و قد رفضت سكايب التعليق على هذه التقارير. *إذا صحت هذه التقارير* فهذا قد يعني أن سكايب لها القدرة على التنصت على المحادثات و تمكين آخرين من ذلك!.

– أبلغ بعض مستخدمي برنامج سكايب على نظام لينكس (Linux) بأن سكايب يحاول الوصول لملف من ملفات النظام (/etc/passwd) على لينكس و هو ملف يضم بيانات مهمة عن حسابات المستخدمين للنظام (لينكس) و يفترض أن سكايب لا يحتاج للوصول لها فلماذا يحاول القيام بذلك؟ ؛ تم اكتشاف ذلك بمراقبة (نداءات النظام system calls) التي قام برنامج سكايب بتنفيذها أثناء تشغيله. و هناك تقارير مشابهة لذلك عند تشغيل سكايب على نظام ماكنتوش (Mac).

شكل (2.7) : تقرير عن الشهادة الرقمية لموقع سكايب عند تسجيل الدخول باستخدام متصفح كروم



4.2 تحديد المشكلة:

يمكن تحديد المشكلة أو المشاكل التي يهدف هذا البحث إلى حلها في النقاط التالية:

1. تزايد المخاطر التي تهدد سلامة و موثوقية المعلومات و تطبيقاتها المختلفة مع تزايد الاعتماد على شبكات الحاسوب و الانترنت في اجراء المحادثات و تبادل المستندات و الملفات المهمة ؛ هذه المعلومات قد تتعرض للكشف أو التلاعب عند ارسالها عبر قنوات غير آمنة – كشبكة انترنت مثلا – ؛ حيث قد يقوم القرصان أو المخترق بالتنصت على قناة الاتصال عبر الشبكة حتى يتمكن من الاطلاع على محتوى الرسائل (مباشرة إذا كانت نصية أو بعد فك ترميزها إذا كانت صوتية) بل قد يقوم بتعديل هذه الرسائل و اعادة ارسالها لهدفها و بالتالي يستقبل الطرف الآخر معلومات غير سليمة ، أي تم التلاعب بها ؛ و قد يقوم بانتحال هوية الشخص المرسل لخداع الطرف الآخر و دفعه إلى كشف معلومات مهمة و حساسة. كل هذا قد ينتج عنه أضرار جسيمة إذا كانت هذه المعلومات حساسة و سرية (مثل أسرار المؤسسات و الشركات التجارية و تفاصيل الحسابات البنكية و المعلومات الشخصية المهمة و غير ذلك).
2. لا تقوم كل برمجيات (و مواقع) الدردشة و التواصل الالكتروني بتأمين الاتصال و تبادل الرسائل و الملفات بين المستخدمين لهذه البرمجيات مما قد يؤدي للمخاطر المشار لها في الفقرة الأولى.
3. بعض البرمجيات تقوم بتأمين عملية تسجيل الدخول فقط، لحماية كلمة السر و اسم المستخدم ؛ و لكن لا تقوم بتأمين المحادثات و تبادل و الملفات ؛ أو لا تقوم بذلك بشكل افتراضي (تلقائي) بل يجب أن يتم ذلك عن طريق ضبط الاعدادات بواسطة المستخدم.
4. بعض البرمجيات قد تقوم بتأمين تسجيل الدخول و تأمين المحادثات و تبادل الملفات و لكنها تستخدم تقنيات تشفير قديمة – إلى حد ما – أو غير قياسية مما قد يؤدي للتشكيك في مستوى الأمان الذي توفره.
5. بعض البرمجيات تقوم بحفظ سجلات لهذه المحادثات على ملقماتها (سيرفراتها) مما قد يعرضها لخطر القرصنة – خاصة إذا كان الملقم غير مؤمن بالمستوى المطلوب أو اذا لم يتم حفظها بشكل آمن (مشفر).
6. بعض هذه البرمجيات – أي التي توفر مستوى جيدا من الأمان – تخضع لبعض القوانين الأمريكية (حيث أنها منتجات تملكها شركات أمريكية) ؛ هذه القوانين قد تُلزِمها (افتراضيا على الأقل) بكشف محتوى الرسائل أو كشف مفاتيح التشفير لبعض الجهات القانونية – وكالة الأمن القومي مثلا – مما قد يؤدي إلى كشف معلومات خاصة قد لا يرغب صاحبها في كشفها بدون إذنه.

7. أغلب هذه البرمجيات هي برمجيات مغلقة المصدر (أي لايمكن الاطلاع على الكود المصدري لها) و بالتالي يصعب تقييمها و دراستها بواسطة الخبراء المستقلين لتحديد تقنيات التشفير التي تستخدمها و بالتالي مستوى الأمان الذي تقدمه.

5.2 الحل المقترح:

بعد دراسة كل ما سبق اقترح الباحث حلا لهذه المشاكل ؛ و يتكون هذا الحل من قسمين هما:

I- تصميم و تطوير مكتبة برمجية للتشفير المعياري – مفتوحة المصدر – ؛ هذه المكتبة تضم مجموعة متنوعة من خوارزميات التشفير الحديثة و المعيارية و يتم تصميمها و اختبارها بشكل جيد حتى يتمكن أي برنامج من استخدامها لتقديم خدمات التشفير المتنوعة بشكل جيد.

II- تصميم و تطوير برنامج دردشة – مفتوح المصدر – يستخدم المكتبة المذكورة أعلاه لتأمين الاتصال بين المستخدمين عن طريق تشفير المحادثات و التحقق من الهوية و من صحة و سلامة الرسائل (التوقيع الرقمي).

6.2 مزايا الحل المقترح:

1- توفير مستوى عالٍ من الأمان عند اجراء المحادثات عن طريق تشفير الرسائل باستخدام خوارزمية AES حيث أنها – حاليا – أهم معيار للتشفير المعتمد عالميا ؛ و يتم استخدام مفتاح بطول 256 بتا و هو أطول مفتاح تسمح به الخوارزمية.

2- يتم استخدام مفتاح جلسة (Session Key) خاص بكل محادثة ؛ أي يتم استخدام مفاتيح مختلفة عند كل محادثة ؛ بالتالي إذا انكشف احد هذه المفاتيح لن يتمكن المخترق من الاطلاع على محتوى كل الجلسات (المحادثات) بين الطرفين.

3- كل مستخدم سوف يكون له توقيعه الرقمي الخاص به (Digital Signature) مبنى على خوارزمية RSA للتشفير بالمفتاح العلني بطول 3072 بتا و خوارزمية الاختزل "الهاش" الآمن SHA2

بطول 512 بتا ، و كلاهما خوارزميتان معياريتان؛ يتم استخدام التوقيع الرقمي لتبادل مفاتيح الجلسة بأمان و للتحقق من هوية الطرف الآخر على قناة الاتصال و للتحقق من صحة و سلامة الرسائل.

4- هذا النظام الهجين ؛ أي الذي يستخدم التشفير المتناظر (خوارزمية AES) و التشفير غير المتناظر (خوارزمية RSA) يستفيد من محاسن كلا النوعين و يتجنب عيوبهما – كما سبق ذكر ذلك في الباب السابق.

5- لايتم تخزين مفاتيح الجلسة على الملقم (السيرفر) ؛ (حيث أن الاتصال بين الطرفين – أي المستخدمين - يكون من نوع الند للند peer–to-peer) بل ان الملقم لن يطلع على هذه المفاتيح من الأساس ؛ و بالتالي لن يتمكن مشغل الملقم (مدير النظام) – أو غيره - من الاطلاع على محتوى الرسائل ؛ أضف إلى ذلك أنه إذا حدث اختراق للملقم فلن يتمكن المخترق من الاطلاع على مفاتيح الجلسة السرية.

6- لا يتم حفظ سجل للمحادثات على السيرفر ، بل تحفظ على جهاز المستخدم بعد تشفيرها بكلمة السر الخاصة به و توقيعها رقميا. يضمن لنا هذا مستوى أعلى من الأمان، إذا تم قرصنة الملقم فلن يتمكن المخترق من الاطلاع على سجل المحادثات.

7- القسم الأول من الحل المقترح – أي تصميم مكتبة برمجية للتشفير المعياري – يمكننا من استخدام هذه المكتبة في كل التطبيقات التي تحتاج للتشفير بأنواعه المختلفة و ليس فقط تشفير (تأمين) الرسائل عبر الشبكة ؛ بالاضافة إلى استخدامات التوقيع الرقمي المختلفة.

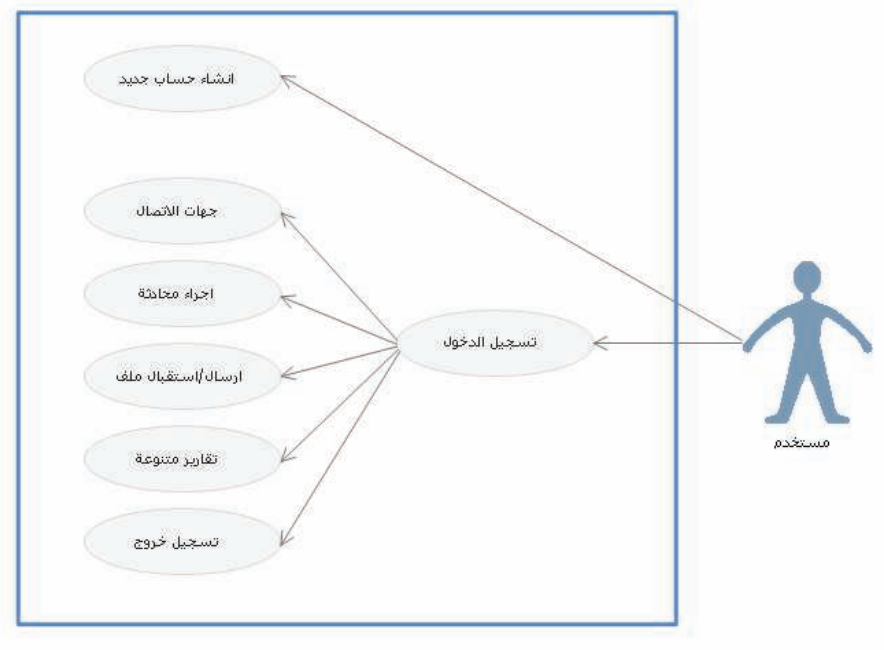
8- كون المكتبة و البرنامج المقترحان مفتوحا المصدر يقدم مستوى أفضل من الأمان لأنه سيصبح بامكان الخبراء المستقلين في المجال الاطلاع على الخوارزميات المستخدمة و اجراء كل الاختبارات اللازمة للتأكد من مستوى و معيارية الأمان و جودة الخدمة.

7.2 حالات الاستخدام :

في هذه الفقرة سيحاول الباحث عرض حالات الاستخدام (Use Case) للبرنامج تمهيدا لتحديد المتطلبات التي يجب توفرها في البرنامج ؛ يوضح شكل (2.8) مخطط عام لحالات الاستخدام و التي سيتم عرضها على شكل جداول توضح طلبات المستخدم و استجابة البرنامج لهذه الطلبات ؛ ثم يتم في الباب

الثالث تصميم كلاسات (Classes) البرنامج و تحديد العلاقات التي تربطها معا و مسئولية كل كلاس منها و وظيفته بناء على حالات الاستخدام.

شكل (2.8) : حالات الاستخدام بشكل عام



✓ **حالة الاستخدام: انشاء حساب جديد:**

يعرض الجدول التالي حالة الاستخدام : انشاء حساب جديد ؛ أي قيام المستخدم بانشاء حساب على البرنامج حتى يتمكن المستخدم من تسجيل الدخول و الاستفادة من الخدمات التي يقدمها البرنامج.

جدول (2.2) : حالة الاستخدام : انشاء حساب جديد.

طلبات المستخدم	استجابة البرنامج
1. يقوم المستخدم بملء نموذج (فورم) انشاء حساب جديد. 2. يقوم المستخدم بعد ذلك بضغط زر انشاء.	3. يقوم البرنامج أولا بالتحقق من ادخال كل البيانات المطلوبة ؛ ثم يتحقق من أن كلمة السر بالطول و العشوائية المناسبان (أي طول كلمة السر 16 حرفا على الأقل و تضم حروف و رموز و أرقام) و تم تكرارها بشكل سليم و أن عنوان البريد الالكتروني مكتوب بصيغة سليمة.

	إذا لم يتحقق ذلك يتم ابلاغ المستخدم ليقوم بتصحيح الخطأ. 4. يقوم البرنامج باجراء اتصال آمن بالملقم لارسال البيانات لانشاء حساب جديد ؛ إذا كانت كل البيانات كما هو مطلوب يقوم الملقم بتسجيل اسم المستخدم و كلمة السر (هاش و ليس كلمة السر نفسها) و بريده الالكتروني بشكل آمن على قاعدة البيانات ثم يبلغ البرنامج بنجاح العملية. 5. يقوم البرنامج بتوليد زوجي المفاتيح الخاص و العلني للمستخدم (التوقيع الرقمي) و يرسل نسخة من مفتاح المستخدم العلني للملقم بشكل آمن. 6. يقوم الملقم بحفظ المفتاح العلني في قاعدة البيانات مع بقية بيانات المستخدم بشكل آمن و يبلغ البرنامج بنجاح العملية. 7. يقوم البرنامج بحفظ المفاتيح بشكل آمن على القرص الصلب ثم يبلغ المستخدم بنجاح العملية ثم يتم فتح نافذة تسجيل الدخول حتى يقوم المستخدم بتسجيل الدخول.

✓ **حالة الاستخدام: تسجيل الدخول:**

جدول (2.3) : حالة الاستخدام : تسجيل الدخول

طلبات المستخدم	استجابة البرنامج
1. يقوم المستخدم بادخال اسم المستخدم و كلمة السر. 2. يقوم المستخدم بعد ذلك بضغط زر تسجيل الدخول.	3. يقوم البرنامج باجراء اتصال آمن بالملقم و ارسال البيانات له مع توقيعها رقميا لتسجيل الدخول. 4. يقوم الملقم بالتحقق من أصالة البيانات (سلامتها) ثم يقوم بالتأكد من صحة اسم المستخدم و كلمة السر و يبلغ البرنامج بذلك مع تسجيل دخول

	المستخدم و عنوان الآي بي الخاص به و زمن و تاريخ العملية في سجل الدخول على الملقم بشكل آمن. 5. إذا كان اسم المستخدم و كلمة السر صحيحان يتم فتح النافذة الرئيسية للبرنامج مع تحميل كل البيانات اللازمة من على القرص، و إلا يتم ابلاغ المستخدم بأن اسم المستخدم أو كلمة السر خطأ ليعيد المحاولة. 6. في حالة تم تسجيل الدخول بنجاح يتصل البرنامج بالملقم بشكل آمن للاستعلام عن حالة جهات الاتصال الخاصة بالمستخدم – إذا وجدت – أي هل هم متصلون الآن أم لا حتى يتم اظهار حالتهم على النافذة الرئيسية و الحصول على عناوين الآي بي خاصتهم لارسال تنبيه لهم بأن المستخدم قام بتسجيل الدخول. 7. بعد نجاح عملية تسجيل الدخول يتم اضافة تاريخ و وقت العملية على تقرير تسجيل الدخول على القرص بأمان.

✓ **حالة الاستخدام: جهات الاتصال:**

أي الأشخاص المضافين كجهات اتصال لدى المستخدم ؛ الذي قد يرغب في اضافة أو حذف جهة اتصال لديه. بغرض التبسيط سيتم الحديث هنا عن حالة اضافة جهة اتصال جديدة (أي اضافة شخص جديد للقائمة من الأشخاص المستخدمين للبرنامج).

جدول (2.4) : حالة الاستخدام : جهات الاتصال (اضافة)

طلبات المستخدم	استجابة البرنامج
1. يقوم المستخدم بالنقر على "اضافة جهة اتصال.	
	2. يقوم البرنامج بفتح نافذة للبحث عن المستخدمين من خلال الاسم ؛ يجب أن يقوم المستخدم بادخال اسم المستخدم الذي يبحث عنه ثم يضغط زر بحث.
3. يقوم المستخدم بادخال الاسم و ضغط زر بحث.	4. يقوم البرنامج باجراء اتصال آمن مع الملقم بغرض البحث عن اسم المستخدم أو الاسماء المشابهة له ؛ مع تنبيه المستخدم لينتظر لحظات.

	5. يقوم الملقم بالبحث عن الاسم المطلوب أو الأسماء المشابهة له و يرسلها بشكل آمن للبرنامج أو يبلغه بأن الاسم غير موجود. 6. يقوم البرنامج بعرض النتائج على الشاشة على شكل قائمة من الأسماء ليختار منها ما يشاء ؛ أو يبلغ المستخدم بعدم وجود الاسم.
8. يقوم المستخدم باختيار الاسم الذي يريد من القائمة التي تم عرضها – في حال كانت نتيجة البحث ايجابية – ؛ و يقوم بالنقر على زر ارسال طلب اضافة.	9. يقوم البرنامج بالاتصال بالملقم بشكل آمن لارسال طلب الاضافة لهذا المستخدم بدلالة اسمه. 10. يقوم الملقم بتسجيل الطلب عنده – بشكل آمن – بغرض ارساله للمستخدم المطلوب اضافته و يتم ابلاغ البرنامج بذلك ؛ في حالة دخول المستخدم الآخر يجب على الملقم ارسال طلب الاضافة له حتى يقبله أو يرفضه ؛ في حالة القبول يتم ابلاغ البرنامج على جهاز المستخدم الذي ارسل الطلب ليتم اضافة المستخدم المطلوب اضافته في قائمة جهات الاتصال. 11. يقوم البرنامج بابلاغ المستخدم بأنه تم ارسال طلب الاضافة بنجاح و أن عليه الانتظار حتى يقرر الطرف الآخر القبول أو الرفض. 12. بعد نجاح عملية اضافة جهة اتصال يتم اضافة تفاصيل العملية إلى تقرير جهات الاتصال على القرص بأمان.

✓ **حالة الاستخدام: اجراء محادثة:**

حالة الاستخدام هذه توضح ما سوف يحدث عند اجراء محادثة بين اثنين من مستخدمي البرنامج.

جدول (2.5) : حالة الاستخدام : اجراء محادثة

طلبات المستخدم	استجابة البرنامج
1. يقوم المستخدم بالنقر المزدوج على اسم المستخدم في قائمة جهات الاتصال لديه أو اختيار اجراء محادثة من	2. يقوم البرنامج و بشكل آمن من التحقق من أن الطرف الآخر موجود الآن – أي قام بتسجيل الدخول – ؛

القائمة.	في حالة كان كذلك يتم فتح نافذة حوار يتم من خلالها تبادل الرسائل ؛ كما يجب أن يقوم البرنامج بالتحقق من هوية الطرف الآخر ؛ إذا كان هناك أي خلل يتم تحذير المستخدم بشكل واضح أن هذا الشخص ليس هو من يدعيه و أن هويته ليست صحيحة. إذا لم يكن الطرف الآخر موجودا – أي لم يقم بتسجيل الدخول – يتم ابلاغ المستخدم بتعذر اجرء هذه المحادثة لأن الطرف الآخر لم يقم بتسجيل الدخول.
3. في نافذة الحوار يقوم المستخدم بكتابة الرسالة التي يريدها ثم يضغط زر ارسال.	4. يقوم البرنامج بتشفير الرسالة و توقيعها رقميا ثم يقوم بارسالها إلى الطرف الآخر مع عرض نسخة منها على نافذة الحوار. 5. عند استلام أي رسالة من الطرف الآخر فيجب فك التشفير و التحقق من سلامتها من التعديل أثناء نقلها عبر الشبكة ؛ كما يجب التحقق من أن هذه الرسالة قادمة فعلا من الشخص المطلوب. 6. يتم عرض الرسالة المستلمة على نافذة الحوار بعد التأكد مما سبق ذكره في الفقرة السابقة و إذا كان هناك خلل في صحة الرسالة أو صحة مصدرها فيتم عرضها للمستخدم مع تحذيره بشكل واضح من أن هذه الرسالة ليست صحيحة – أي تم تعديلها بواسطة طرف آخر – أو أن هوية المرسل غير صحيحة. 7. بعد انتهاء العملية أي المحادثة يتم اضافة تاريخ و وقت البداية و النهاية و اسم الطرف الآخر إلى تقرير المحادثات على القرص بأمان.

✓ **حالة الاستخدام: ارسال / استقبال ملف:**

توضح حالة الاستخدام هذه ما يتم عند قيام المستخدم بارسال أو استقبال ملف خلال البرنامج.

جدول (2.6) : حالة الاستخدام : ارسال / استقبال ملف

طلبات المستخدم	استجابة البرنامج
1. أثناء أي حوار بين المستخدم و طرف آخر على نافذة الحوار يقوم المستخدم بالنقر على ارسال ملف.	2. يقوم البرنامج بفتح نافذة اختيار ملف ؛ هذه النافذة تمكن المستخدم من اختيار أي ملف يريده من الملفات الموجودة على القرص الصلب لديه.
3. يقوم المستخدم باختيار الملف الذي يريده و يضغط على زر اختيار.	4. يقوم البرنامج باغلاق نافذة اختيار ملف مع عرض اسم و مسار الملف و حجمه على نافذة الحوار (المحادثة) و بجوار ذلك يتم عرض زرين احدهما ارسال و الآخر الغاء ؛ على المستخدم إذا كان متأكدا من أن هذا هو الملف فيجب أن يضغط زر ارسال.
5. إذا رغب المستخدم في الارسال عليه ضغط زر ارسال.	6. يقوم البرنامج بابلاغ المستخدم بأنه سيتم ارسال طلب ارسال ملف للطرف الآخر حتى يبدأ الارسال فور استقبال قبول للطلب من الطرف الآخر ؛ و يجب تمكين المستخدم من إلغاء العملية في أي لحظة عن طريق زر الغاء. 7. في حالة قبول الطرف الآخر يتم البدء في قراءة الملف و تقسيمه بشكل مناسب – لتجنب استهلاك الذاكرة بشكل كبير – يتم تقسيمه لأجزاء ثابته حتى يتم تشفيرها و توقيعها ثم ارسالها للطرف الآخر. 8. في حالة كان المستخدم هو من سيقوم باستلام الملف و قام بالنقر على الموافقة على استلام الملف ؛ يقوم البرنامج باستلام أقسام الملف و فك تشفيرها و التحقق من سلامتها بشكل مشابه للرسائل و المحادثات في حالة الاستخدم السابقة و حفظه على مكان مؤقت على القرص؛ و يجب ابلاغ المستخدم بشكل واضح في حالة حدوث أي انتهاك لأمن المعلومات عند استقبال الملف.
10. يقوم المستخدم باختيار اسم و مكان حفظ الملف و يضغط زر حفظ.	9. بعد وصول كل اجزاء الملف بنجاح يتم ابلاغ المستخدم بذلك مع فتح نافذة لاختيار مكان حفظ الملف

	على القرص. 11. يتم حفظ الملف بالاسم و في المكان اللذان تم اختيارهما و يتم ابلاغ المستخدم بنجاح العملية. 12. يتم حفظ اسم الملف و مساره و حجمه مع التاريخ و الوقت على تقرير ارسال/ استقبال ملف على القرص بأمان.

✓ **حالة الاستخدام: تقارير متنوعة:**

توضح هذه الحالة ما يتم عند طلب عرض تقرير ما أو عند طلب حذفه.

جدول (2.7) : حالة الاستخدام : تقارير متنوعة

طلبات المستخدم	استجابة البرنامج
1. يقوم المستخدم بالنقر على خيار تقارير متنوعة.	2. يقوم البرنامج بعرض نافذة تقارير متنوعة و بها تظهر أنواع مختلفة من التقارير على المستخدم تحديد النوع ثم ضغط زر عرض.
3. على المستخدم اختيار النوع الذي يريد ثم ضغط زر عرض.	4. يقوم البرنامج بعرض قائمة للتقارير الموجودة ضمن هذا النوع و على المستخدم اختيار اما عرض التقرير أو حذفه.
5. يقوم المستخدم باختيار التقرير الذي يريد ثم الضغط على زر حذف لحذفه بشكل نهائي أو زر عرض لعرضه على النافذة.	6. إذا قام المستخدم باختيار الحذف يتم مطالبته بادخال كلمة السر للتحقق من صحة الطلب ؛ إذا تم ادخالها بشكل سليم يتم حذف التقرير المحدد بشكل نهائي من على القرص ؛ أما إذا اختار المستخدم عرض التقرير فيتم عرضه في الجزء الأسفل من النافذة مع تنسيق العرض بشكل جيد.

✓ **حالة الاستخدام: تسجيل الخروج:**

جدول (2.8) : حالة الاستخدام : تسجيل الخروج

طلبات المستخدم	استجابة البرنامج
1. يقوم المستخدم بالنقر على تسجيل الخروج أو يقوم باغلاق البرنامج بشكل مباشر.	2. يتم اغلاق نافذة البرنامج. 3. يقوم البرنامج بالاتصال بالملقم بشكل آمن لابلاغه بتسجيل الخروج من البرنامج. 4. يقوم البرنامج باغلاق كل الملفات المفتوحة و يتم تصفير كل مواقع الذاكرة التي تضم المفاتيح أو أي معلومات مهمة ثم يتم تسجيل زمن و تاريخ تسجيل الخروج على ملف التقارير. 5. يتم اغلاق البرنامج.

8.2 تحديد المتطلبات:

1.8.2 المتطلبات الوظيفية:

1) **الأمان:**

أ. يجب أن يقدم البرنامج مستوى عاليا و معياريا من الأمان باستخدام تقنيات التشفير المعيارية؛ حيث يجب أن يتم تشفير كل الرسائل عبر الشبكة باستخدام خوارزمية AES_CBC بمفتاح طوله 256 بتا ؛ لكل جلسة مفتاحين مختلفين ؛ مفتاح لكل طرف في المحادثة مع تغيير المفاتيح في كل جلسة اتصال بينهما، و يجب استخدام خوارزمية RSA – بطول 3072 بتا – لتبادل المفاتيح بشكل آمن.

ب. يجب أن يتم تأمين تسجيل الدخول للمستخدم و كذلك تأمين كل عمليات التواصل أي اجراء المحادثات و تبادل الملفات.

ت. كما يجب إلزام المستخدم باختيار اسم فريد – غير مكرر – و استخدام كلمة سر آمنة ذات طول مناسب (16 حرفا على الأقل) و ذات عشوائية مناسبة (أي تضم حروف و رموز و أرقام) ؛ و لا يتم تخزين كلمة السر نفسها بل تخزين قيمة "الهاش" – الذي يتم توليده منها – ثم بعد تشفيره و تشفير بقية بيانات المستخدم تخزن في قاعدة البيانات.

ث. كما يجب أن تكون عملية التأمين (أي التشفير) افتراضية – أي تلقائية – سواء كان تأمين تسجيل الدخول أو تأمين الرسائل و الملفات المتبادلة بين المستخدمين.

ج. كما يجب استخدام التوقيع الرقمي باستخدام خوارزمية RSA و خوارزمية SHA2 بطول 512 بتا ؛ يتم استخدام التوقيع الرقمي في توقيع الرسائل المتبادلة للتحقق من صحتها و من صحة مصدرها.

ح. كما يجب أن يتم تأمين الملقم بشكل جيد من حيث استخدام نظام تشغيل عالي الأمان، و حفظ بيانات المستخدمين على قاعدة بيانات مؤمنة جيدا عن طريق استخدام التشفير القوي لحفظ هذه البيانات لفترة طويلة و للتأكد من صحتها.

خ. كما يجب على الملقم رفض أي اتصال من عميل غير مسجل لديه و ذلك عن طريق التحقق من صحة رقم العميل أولا – أي قبل التحقق من الاسم و كلمة السر – و يتم رفض الاتصال إذا لم يكن رقم العميل موجودا (و هو رقم فريد يقوم الملقم بتوليده تلقائيا و ارساله لبرنامج العميل عند انشاء حساب جديد).

2) هوية فريدة لكل مستخدم:

يجب أن يكون لكل مستخدم هوية فريدة (Unique ID) يتم توليدها باستخدام خوارزمية RSA بطول 3072 بتا؛ حيث يتم ربط المفتاح العلني الخاص بالمستخدم مع رقمه (أي رقم العميل المذكور في الفقرة أعلاه) و اسمه و بقية بياناته و حفظها – بعد تشفيرها – على الملقم ؛ و يجب أن تنتهي صلاحية المفتاح العلني بعد مرور عامين، و يتم اصدار مفتاح جديد و ذلك لكل مستخدم.

3) المقدرة على اكتشاف أي انتهاك لأمن المعلومات:

يجب أن يكون البرنامج قادرا على اكتشاف أي خلل أو انتهاك لأمن المعلومات ؛ و ذلك من خلال التحقق من هوية الطرف الآخر و تنبيه المستخدم إذا كانت هذه الهوية غير صحيحة ؛ كما يجب التأكد من

صحة و سلامة الرسائل و تنبيه المستخدم إذا حدث أي تعديل لهذه الرسائل عند انتقالها عبر الشبكة ؛ هذه الفقرة و الفقرتين السابقتين لها هي أهم متطلبات البرنامج.

4) حفظ سجلات تسجيل الدخول و سجلات المحادثات:

يجب أن يقوم البرنامج بحفظ سجلات تسجيل الدخول و سجلات المحادثات على جهاز المستخدم فقط ؛ مع الالتزام بتأمينها – أي تشفيرها – باستخدام كلمة السر الخاصة بالمستخدم و لابد من توقيعها رقميا حتى يتمكن البرنامج من تنبيه المستخدم اذا تم التلاعب بها – أي اذا تم تعديلها – .

5) معمارية الشبكة:

يتم الاتصال بين المستخدمين باستخدام تقنية الند – للند (peer-to-peer) ؛ بحيث أن الملقم لا يمكن أن يكون له أي سيطرة على الاتصال بين المستخدمين، بل ينحصر دوره في توزيع عناوين الآي بي (IP) و توزيع المفاتيح العلنية الموقعة ؛ بطريقة آمنة للمستخدمين ؛ و حفظ سجل لكل عمليات تسجيل الدخول للخدمة فقط.

كما يجب أن يدعم النظام امكانية تشغيله على أنواع مختلفة من الشبكات ؛ شبكة انترنت أو شبكة انترانت أو شبكة محلية.

6) جهات الاتصال:

يجب أن يتمكن المستخدم من اضافة جهات اتصال جديدة لقائمة جهات الاتصال لديه ؛ كما يجب أن يكون قادرا على ازالتها من قائمته. كما يجب أن يتمكن المستخدم من البحث عن جهات اتصال جديدة باستخدام الأسماء.

7) التقارير:

يجب أن يوفر البرنامج للمستخدم مجموعة مبسطة و متنوعة من التقارير منها:

- تقرير عمليات تسجيل الدخول التي قام بها.
- تقرير المحادثات مع جهات الاتصال.
- تقرير بانتهاكات أمن المعلومات المكتشفة – إذا وجدت – .

كما يجب أن يوفر الملقم سجل (log) للعمليات التي تتم – لا يضم معلومات تفصيلية عن المستخدمين – و سجل آخر للأخطاء التي قد تحدث – يضم معلومات تفصيلية عن الخطأ و سببه – و ذلك حتي يتمكن مدير النظام من الاطلاع عليها و عمل اللازم عند حدوث أي خطأ.

2.8.2 المتطلبات غير الوظيفية :

1- سهولة الاستخدام:

يجب أن يوفر البرنامج واجهة رسومية (GUI) سهلة الاستخدام – بعبارة أخرى : صديقة للمستخدم – بحيث تكون مبسطة و خالية من التعقيدات، كما يجب أن تتم كل عمليات تأمين الاتصال و المحادثات و كل عمليات التشبيك بشكل آلي تماما و عدم ازعاج المستخدم بتفاصيل هذه العمليات.

2- الكفاءة و الموثوقية:

يجب أن يقوم البرنامج بانجاز كل المهام المطلوبة منه بالدقة و السرعة المطلوبتان مع توفير آلية مناسبة للتعامل مع الأخطاء المتوقعة و غير المتوقعة (الاستثناءات) و عرض معلوماتها – بشكل موجز – للمستخدم.

3 – دعم اللغة العربية:

يجب أن يوفر البرنامج دعما كاملا للغة العربية مع استخدام واجهة باللغة العربية ؛ حيث أن البرنامج موجه بالأساس للمستخدم العربي.

4 – أداة الاختبار (Test Utility):

يفضل أن يوفر البرنامج أداة منفصلة (برنامج خدمي منفصل) تتيح للمستخدم الخبير – إذا أراد – تشغيل هذه الأداة لاختبار خوارزميات التشفير المتنوعة التي توفرها المكتبة للتأكد من معياريتها و عملها بالشكل المطلوب.

9.2 دراسة الجدوى:

1.9.2 دراسة الجدوى الاقتصادية:

تشمل التقديرات الأولية للتكلفة الاقتصادية لتطوير و تشغيل الحل المقترح على حسب الأسعار التقديرية أثناء اعداد البحث – و ذلك لتشغيل النظام على شبكة الانترنت مع ملاحظة أن التكلفة ستصبح أقل بكثير عند تشغيل النظام على شبكة محلية:

جدول (2.9) : تقديرات التكلفة الاقتصادية

م	الوصف	الكمية	سعر الوحدة بالجنيه	ملاحظات
1	Desktop: Core i7 – 64 Bit, Ram 16 GB, Hard Disk 1.5 TB.	1	15000	Server PC
2	نظام تشغيل: Oracle Enterprise Linux 6.3 x86-64	1	-	مفتوح المصدر
3	اسم نطاق domain name	1	1000	
4	رقم IP ثابت	1	1000	
5	خط انترنت عالي السرعة DSL	1	1000	
6	معدات و أجهزة شبكة متنوعة		5000	
الاجـــمـــــــالي			23000	

2.9.2 دراسة الجدوى التقنية :

تهدف دراسة الجدوى التقنية إلى توضيح الجوانب التقنية للبرنامج و تقييم التقنيات التي سيتم استخدامها ؛ و سيتم مناقشة ذلك في النقاط التالية:

1. **تقنيات التأمين (التشفير) :**

I. يتم استخدام تقنيات التشفير المعيارية و المعتمدة لدى الخبراء، و التي تقدم مستويات عالية من الأمان، و يتم تطبيق أهم مفاهيم أمن المعلومات – و التي سبقت الاشارة لها في الباب الأول – . من أهم هذه الخوارزميات خوارزمية AES و التي سيتم استخدامها في تشفير كل الرسائل و كل البيانات التي يتم تبادلها عبر الشبكة مع استخدام مفاتيح عشوائية – يتم توليدها آليا – بطول 256 بتا للمفتاح و استخدام

وضع (CBC Mode)[1] و هو الأفضل بالنسبة لتشفير الرسائل و ارسالها عبر الشبكة ، و استخدام وضع GCM لتشفير البيانات عند حفظها في قاعدة البيانات لأن هذا الوضع يضمن السرية و التحقق من سلامة البيانات و أصالتها ؛ بالاضافة إلى خوارزمية RSA ذات مفاتيح بطول 3072 بتا ، و يتم توليدها بشكل آلي ؛ و تستخدم في توليد الهويات الرقمية للمستخدمين و في التوقيع الرقمي و للتحقق من صلاحية الهويات و سلامة و تكامل البيانات و تستخدم معها خوارزمية SHA2 بطول 512 بتا لتوليد الهاش ؛ بالاضافة إلى استخدامها – أي RSA – في تبادل مفاتيح الجلسات بشكل آمن مع استخدام البروتوكولات القياسية في ذلك و تضيق المساحة عن تفصيلها هنا.

II. للتحقق من صحة الهوية قبل بدء أي محادثة بين مستخدمين ؛ يتم استخدام رقم المستخدم – الذي يقوم الملقم بتوليده بشكل آلي عند انشاء الحساب – و اسم المستخدم و مفتاحه العلني و عنوان IP الخاص به أثناء تسجيله للدخول ؛ إذا كانت كل هذه البيانات صحيحة و مطابقة للبيانات الموجودة بالملقم يتم ابلاغ المستخدم بصحة هوية الطرف الآخر و إلا يتم ابلاغه بوجود خلل في الهوية و تحديد مكان الخلل أي ابلاغه بأن كل أو بعض البيانات المذكورة غير صحيحة أو غير مطابقة. كذلك بالنسبة للرسائل المرسلة بين المستخدمين، تحتوي كل رسالة على اسم المرسل لها و يتم توقيعها ثم تشفير الرسالة بالاضافة إلى تشفير التوقيع، يقوم المستلم بفك التشفير و التحقق من صحة التوقيع و صحة اسم المرسل و إلا يتم الغاء الرسالة.

2. أدوات و لغات البرمجة:

اختار الباحث استخدام لغة البرمجة الأشهر: سي بلس بلس (C++) ؛ و ذلك لخصائص البرمجة القوية و المرنة التي توفرها ؛ كما تتميز البرامج المكتوبة بهذه اللغة بسرعة عالية في الأداء مقارنة بمثيلاتها[2] و هذه السرعة مطلوبة هنا لأن البرنامج يتعامل مع الشبكة، و لأن التشفير يجب أن يتم بسرعة حتى لا يكون سببا في بطء البرنامج ؛ كما أنها توفر دعما كاملا للبرمجة كائنية التوجه OOP و هي الطريقة الأمثل و الأحدث للبرمجة و لها مميزات متعددة لا يتسع المجال لعرضها بالتفصيل. بالاضافة إلى ذلك يعتقد الباحث أن هذه اللغة توفر مرونة أكبر في التعامل مع الذاكرة إذا تم استخدامها بالشكل المناسب، و هي مسألة مهمة عند التطبيق العملي (Implementation) لخوارزميات التشفير.

كما تم اختيار استخدام أداة Qt-SDK للبرمجة بلغة سي بلس بلس ؛ حيث تضم هذه الأداة مئات المكتبات البرمجية المتنوعة و الحديثة و التي قام بتطويرها و اختبارها مئات الخبراء حول العالم ؛ و هي أداة واسعة الاستخدام و تدعم تطوير برمجيات مفتوحة المصدر و متعددة المنصات (Cross-platform) ؛

(1) و انظر كتاب اسشنيير "Applied Cryptography" لتفاصيل أوضاع التشفير في الباب التاسع.

(2) انظر مقارنة ممتعة لمجموعة من لغات البرمجة في بحث بعنوان An empirical comparison تأليف Lutz Prechelt و آخرين – جامعة Karlsruhe بألمانيا في مارس 2000م.

يتم استخدام هذه المكتبات الموثوقة في بناء البرنامج مما يدعم الموثوقية التي يوفرها، و يقلل زمن و تكلفة التطوير بشكل كبير و يجعل عملية صيانة البرنامج و تطويره أسهل و أسرع. تم ترجمة البرنامج باستخدام مترجم MS-Visual C++ 2010 (للنسخة التي تعمل على نظام وندوز) و هو مترجم مشهور و ينتج برامج أصغر حجما و أسرع أداءا من المترجمات الأخرى نسبة لتقنيات التحسين (Optimization) التي أضافتها له شركة مايكروسوفت.

3. تقنية التشبيك:

سيتم استخدام نموذج شبكة الند–للند (peer-to-peer) في البرنامج حيث توفر مزيدا من الخصوصية بين المستخدمين الذين يتصلون بشكل مباشر بدون تدخل أي وسيط؛ و يقوم الملقم بتوزيع عناوين IP الخاصة و يتم كذلك استخدام نموذج شبكة عميل/ملقم (client/server) لربط العملاء مع الملقم.

4. تقنيات رفع الأداء:

سيتم استخدام تقنية تعدد المسالك (Multi-Thread) – و يتوفر الدعم لها في أغلب نظم التشغيل الحديثة – و التي ترفع سرعة الأداء بشكل ملحوظ (بنسبة 30% تقريبا) ؛ حيث سيتم تطبيقها عند التشفير و فك التشفير و اللذان قد يحتاجا لوقت اضافي مما قد يقلل من سرعة أداء البرنامج الذي يحتاج إلى انجاز المهمة بسرعة لأنه يعمل على شبكة ، كما سيتم تطبيقها على الملقم بحيث يقوم بانشاء مسلك تنفيذ (Thread) خاص بكل اتصال يتعامل معه مما يمكنه من التعامل مع مجموعة كبيرة من الاتصالات في نفس اللحظة بشكل متوازي.

5. تقنية التعامل مع الأخطاء:

ستم استخدام تقنية التعامل مع الاستثناءات (Exception Handling) للتعامل مع الأخطاء غير المتوقعة ؛ و التي إذا لم يتم التعامل معها بكفاءة قد تسبب انهيار النظام، كما سيتم توفير آلية مناسبة للتعامل مع الأخطاء المتوقعة – مثل أخطاء ادخال البيانات من المستخدم – و ابلاغه بها.

3.9.2 دراسة الجدوى الفنية:

في هذه الفقرة نتناول بعض الفوائد المهمة التي يوفرها هذا البحث كما يلي:

1. تطوير مكتبة برمجية للتشفير (مكتبة ربط ديناميكي DLL) ؛ تضم هذه المكتبة خوارزميات التشفير المعيارية و المعتمدة، و بالتالي يتمكن أي برنامج يحتاج لخدمات التشفير المتنوعة من استخدام هذه المكتبة بكل سهولة، مما يوفر على المبرمج كثيرا من الوقت و الجهد.
2. تطوير برنامج للتواصل (اجراء المحادثات) بأمان ؛ و يمكن أن يستخدم في تبادل المعلومات المهمة و السرية، مثل أرقام الحسابات البنكية و معلومات الصفقات التجارية أو أي معلومات أخرى عبر شبكة الانترنت مع الالتزام بمستوى عال من الأمان.
3. كون البرنامج (و المكتبة) مفتوحا المصدر يوفر فائدة مزدوجة ! ؛ فهو من ناحية يتيح للخبراء في أمن المعلومات اختباره بكل حرية للتحقق من مستوى الأمان الذي يوفره ؛ كما أنه – من ناحية أخرى – يمكن الباحثين و مطوري البرمجيات من المساهمة في تطويره و اضافة المزيد من المميزات له مما يساهم في الاضافة له بلا شك.
4. تنبيه الباحثين في المجال – و بشكل عملي – إلى أهمية المصادر المفتوحة و دورها المتوقع في تطوير برمجيات الحاسوب و نظم المعلومات التي يتزايد الاعتماد عليها محليا و عالميا ؛ حيث أن هناك قيودا كثيرة تم وضعها على البرمجيات و النظم مغلقة المصدر خاصة تجاه المستخدمين في دول العالم الثالث ! ؛ و بالتالي توفر المصادر المفتوحة حلا لهذه المشكلة بتوفير مادة علمية غنية بلا قيود للطلاب و الباحثين.

10.2 الخلاصة:

في هذا الباب تم عرض و مناقشة الكثير من الجوانب المهمة المتعلقة بالتحليل ؛ لعل من أهم النتائج التوصل إلى تحديد المتطلبات التي يجب أن يوفرها البرنامج بشكل تفصيلي ؛ بالاضافة إلى ذكر حالات استخدام النظام التي بالاضافة إلى دورها في توضيح المتطلبات فلها دور مهم في تسهيل تصميم كلاسات (كائنات) البرنامج و تحديد علاقاتها و مسئولياتها و سيتم ذلك في الباب التالي.

الباب الثالث

تصميم النظام

1.3 تمهيد:

في هذا الباب يقوم الباحث بتصميم مكونات (Components) البرنامج و كلاساته[1] (Classes) مع تحديد وظيفة كل كلاس و علاقته بالكلاسات الأخرى ؛ و يتم استخدام مخططي Sequence diagram و Class diagram حيث يستخدم الأول لتحديد كلاسات البرنامج و وظائفها و تفاعلها سويا لتحقيق طلبات المستخدم – بناء على حالات الاستخدام الموضحة في الباب السابق – كما يستخدم الثاني في توضيح علاقات هذه الكلاسات مع بعضها البعض. و سيتم تقديم تصميم منفصل للجزء الذي يمثل الملقم (السيرفر) الخاص بالبرنامج و تصميم آخر للجزء الذي يمثل العميل مع عرض جميع الكلاسات بشكل مختصر.

2.3 التصميم العام لمكونات البرنامج :

تم تقسيم البرنامج إلى مجموعة من الطبقات (Layers) ؛ كل طبقة تضم كلاسا أو أكثر ، و لكل طبقة وظيفة أو وظائف محددة ، بحيث تستلم كل طبقة الطلبات من الطبقة السابقة لها و تقوم بعملها ثم قد ترسل هذه الطلبات الى الطبقة التالية لها ، و تقوم باستلام النتائج و تمريرها للأعلى.

و قد تم اعتماد هذا النموذج لعدة أسباب ؛ من أهمها أنه يقدم فهما جيدا لمكونات البرنامج و طريقة عملها، بالاضافة إلى امكانية تعديل أحد المكونات – عند تطوير و صيانة البرنامج – دون الاضطرار إلى تعديل جميع المكونات الأخرى كما أن كل طبقة تعزل الطبقة العليا عن تعقيدات الطبقة أو الطبقات الأسفل منها و يوفر حماية لمكونات الطبقات السفلية عن طريق عزلها عن الطبقات العليا.

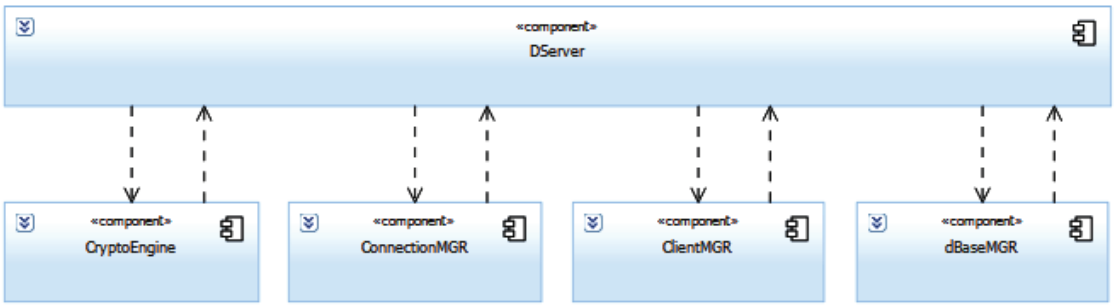
1.2.3 التصميم العام للملقم :

في هذا التصميم (انظر شكل 3.1) يحتوي الملقم على طبقتين من المكونات ، الأولى باسم DServer و هي الطبقة التي تتلقى طلبات العملاء و تقوم بتحليلها و الرد عليها بعد تمريرها الى الطبقة التالية لاجراء اللازم. من الجدير بالذكر هنا أن هذه الطبقة تضم كلاسا بنفس الاسم المذكور آنفا ، حيث اختار الباحث

(1) كلمة class لها ترجمات مختلفة في المراجع العربية (الفئة – الطبقة – الصنف – التصنيف ...) لذلك فضل الباحث استخدام الكلمة كما هي لتجنب الارتباك.

استخدام نمط الواجهة (façade pattern) و هو من أنماط التصميم (Design Patterns) المعروفة و يستخدم لعزل مكونات البرنامج الداخلية عن العالم الخارجي ، كما يسهل عملية صيانة و تطوير البرنامج فعند اجراء تعديل في أي كلاس لا يلزم تعديل جزء كبير من البرنامج بل يتم تعديل الكلاس المعني مع اجراء التعديل المطلوب في الكلاس الواجهة الذي يمثل الغراء أو الصمغ الذي يربط كل الكلاسات معا ، وهو هنا يتمثل في الكلاس DServer و ستتضح تفاصيل ذلك لاحقا. أما الطبقة الأخرى فتضم مجموعة من المكونات ، و كل مكون قد يضم كلاسا أو أكثر – كما سنرى لاحقا- هذه المكونات هي:

شكل (3.1) : التصميم العام للملقم



- CryptoEngine : و يضم كل الكلاسات التي ستستخدم في التشفير و تأمين المعلومات و التحقق من صحتها.
- ConnectionMGR : و هو مسئول من كل علمليات الشبكة.
- ClientMGR : و هو مسئول عن ادارة العملاء.
- dBaseMGR : و مسئوليته تنحصر في توفير الاتصال، و التعامل مع قاعدة البيانات التي تضم معلومات المستخدمين و بعض المعلومات الخاصة بالبرنامج.

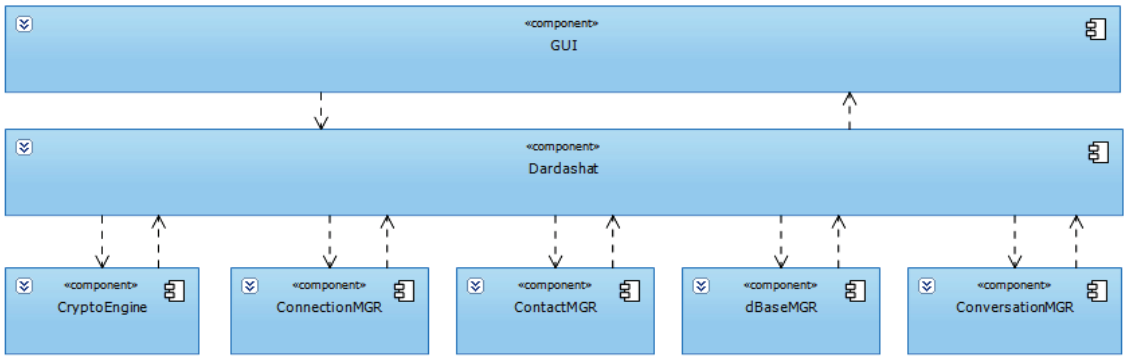
2.2.3 التصميم العام للعميل :

يحتوي التصميم – انظر شكل (3.2) – على ثلاث طبقات من المكونات كما يلي:

- الطبقة الأولى: GUI و هي تمثل الواجهة الرسومية للبرنامج و تضم مجموعة كبيرة من الكلاسات التي سترسم النوافذ و الأزرار و القوائم و غيرها من الكائنات الرسومية ؛ وظيفة هذه الطبقة هي توفير واجهة

رسومية سهلة الاستخدام للمستخدم ، تتلقى هذه الطبقة طلبات المستخدم و ترسلها للطبقة التالية لها ثم تقوم بعرض النتائج القادمة من الطبقة السفلى.

شكل (3.2) : التصميم العام للعميل.



- الطبقة الثانية: Dardashat و تضم كلاسا واحدا بنفس الاسم يمثل الواجهة (façade) التي تعزل مكونات البرنامج الداخلية عن واجهة المستخدم – كما سبقت الاشارة لهذا النمط في الفقرة السابقة. هذا الكلاس له دور مركزي فهو يتلقى كل الطلبات من الطبقة الخارجية (واجهة المستخدم) و يرسل كل طلب للمكون المناسب في الطبقة السفلى حتى يتم اجراء المهمة و ارجاع النتائج لذلك هو الرابط أو الغراء الذي يربط كل مكونات البرنامج معا كما سنرى لاحقا بالتفصيل.
- الطبقة الثالثة: و تضم مجموعة من المكونات – و كل مكون يضم عدة كلاسات – كما يلي:

– CryptoEngine : و هو كما في الملقم مسئول عن كل عمليات التشفير و توفير كائنات التحقق و تأمين المعلومات.

– ContactMGR : هذا المكون مسئول عن كل المعلومات و العمليات المتعلقة بجهات الاتصال الخاصة بالعميل.

– ConversationMGR : مسئول عن ادارة المحادثات التي تتم بين العميل و جهات الاتصال الخاصة به.

– dBaseMGR : و هو مسئول عن ادارة التعامل مع قاعدة البيانات – المصغرة – الخاصة بالعميل التي ستضم سجل المحادثات و توقيعه الرقمي و التقارير المتنوعة و غيرها من المعلومات المهمة.

– ConnectionMGR : و هو مسئول عن ادارة كل عمليات التشبيك.

3.3 مخططات التسلسل Sequence Diagrams :

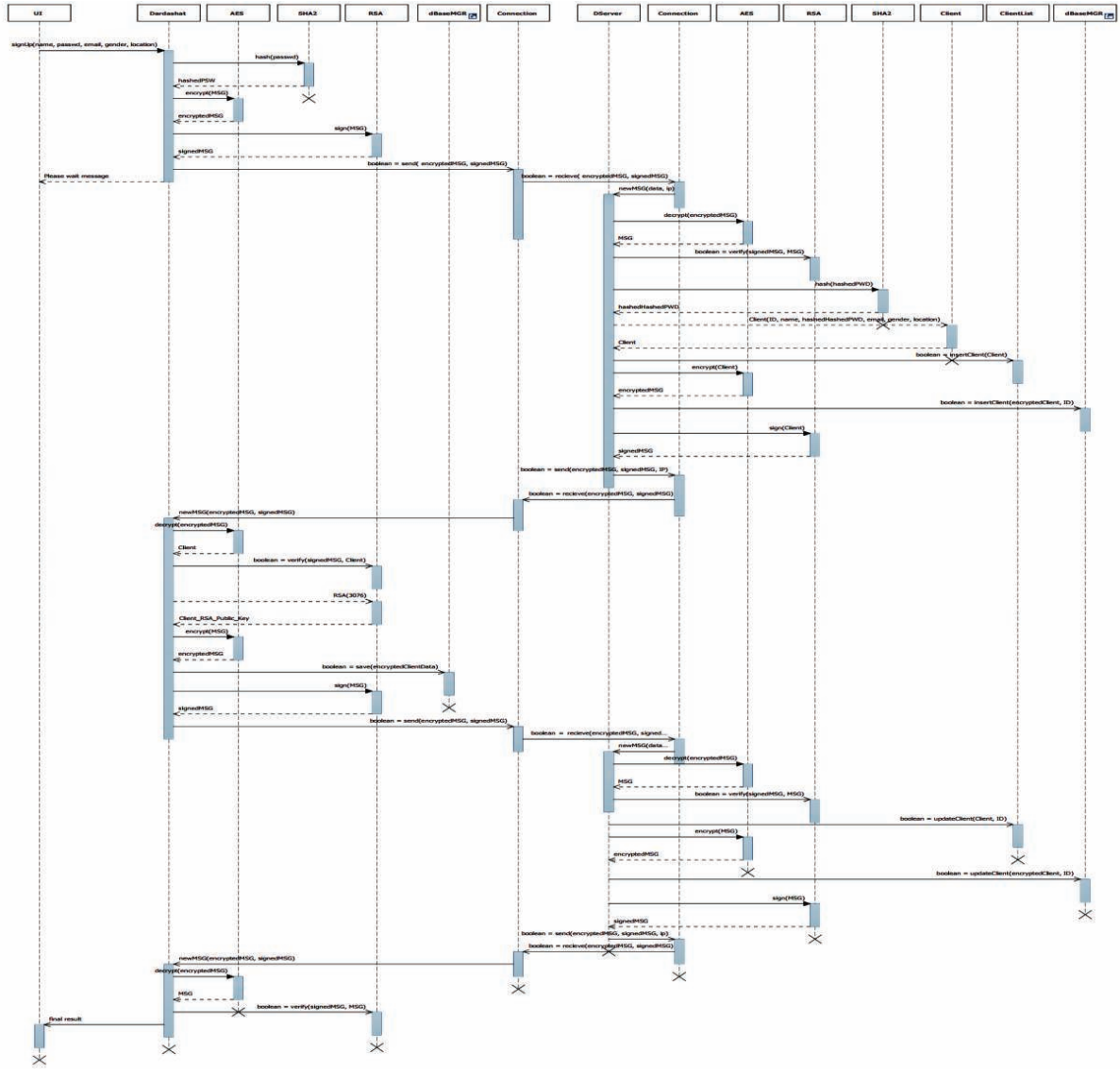
تم بناء مخططات التسلسل Sequence diagram اعتمادا على حالات الاستخدام use case التي سبق تفصيلها في الباب السابق و التي توضح كل الاجراءات و الخطوات التي ستحدث مع كل طلب من العميل ؛ توضح هذه المخططات الكلاسات المستخدمة و مهمة كل كلاس منها كما تساعد في توضيح العمليات (الدوال) و أنواع البيانات (المتغيرات) المستخدمة مما يسهل معرفة تركيب هذه الكلاسات و علاقاتها معا.

1.3.3 انشاء حساب جديد Sign up :

مخطط التسلسل Sequence diagram الخاص بحالة الاستخدام: انشاء حساب جديد موضح بالشكل (3.3) ؛ بعد أن يدخل العميل البيانات المطلوبة لانشاء حساب جديد ؛ تقوم واجهة المستخدم UI – و التي يتعامل معها العميل – بتمرير هذه البيانات إلى كلاس Dardashat، و يتضح هنا أن واجهة المستخدم ينحصر دورها في تلقي الطلبات من العميل و تمريرها للبرنامج ثم عرض النتائج و أن كلاس Dardashat هو الواجهة (facade) التي تعزل واجهة المستخدم عن مكونات البرنامج الداخلية من ناحية ؛ و تقوم بربط جميع مكونات البرنامج الداخلية من ناحية أخرى. أول ما يفعله كلاس Dardashat بعد تلقي البيانات من واجهة المستخدم هو استدعاء الدالة ()hash و هي عضو في الكلاس SHA2 ، تتلقي هذه الدالة كلمة السر الخاصة بالعميل و ترجع الهاش ؛ ثم بعد ذلك يقوم كلاس Dardashat بتجميع كل بيانات العميل بما فيها هاش كلمة السر في رسالة واحدة و يقوم بتشفيرها باستخدام كلاس AES ثم يتم توقيعها رقميا – للتحقق من صحتها لاحقا لدى المستلم – باستخدام كلاس RSA ؛ بعد ذلك يتم ارسال كلا من الرسالة و توقيعها الرقمي للملقم باستخدام كلاس ConnectionMGR مع افادة المستخدم بالانتظار قليلا لارسال البيانات و تلقي الرد من الملقم. في الطرف المقابل يستلم الملقم الرسالة و يقوم الكلاس DServer – و هو واجهة نسخة الملقم من البرنامج – بفك تشفير الرسالة و التحقق من صحتها ثم يقوم بتوليد الهاش من هاش كلمة السر ، بعد ذلك يقوم الكلاس DServer بانشاء كائن من الكلاس Client الذي يمثل المستخدم و بياناته ، ثم يتم اضافة مرجع (reference) الكائن إلى قائمة العملاء التي تتمثل في الكلاس ClientList ؛ إذا تمت العملية بنجاح يتم ارجاع قيمة منطقية : true ؛ ثم يتم تشفير بيانات العميل و يتم تخزينها مع رقم فريد (ID) – يمثل رقم العميل الذي تم توليده – في قاعدة

البيانات بواسطة الكلاس dBaseMGR، بعد ذلك يتم تشفير رسالة تفيد بنجاح العملية مع رقم العميل و اسمه توقيعها رقميا ثم ارسالها للعميل.

شكل (3.3) : مخطط التسلسل الخاص بانشاء حساب جديد



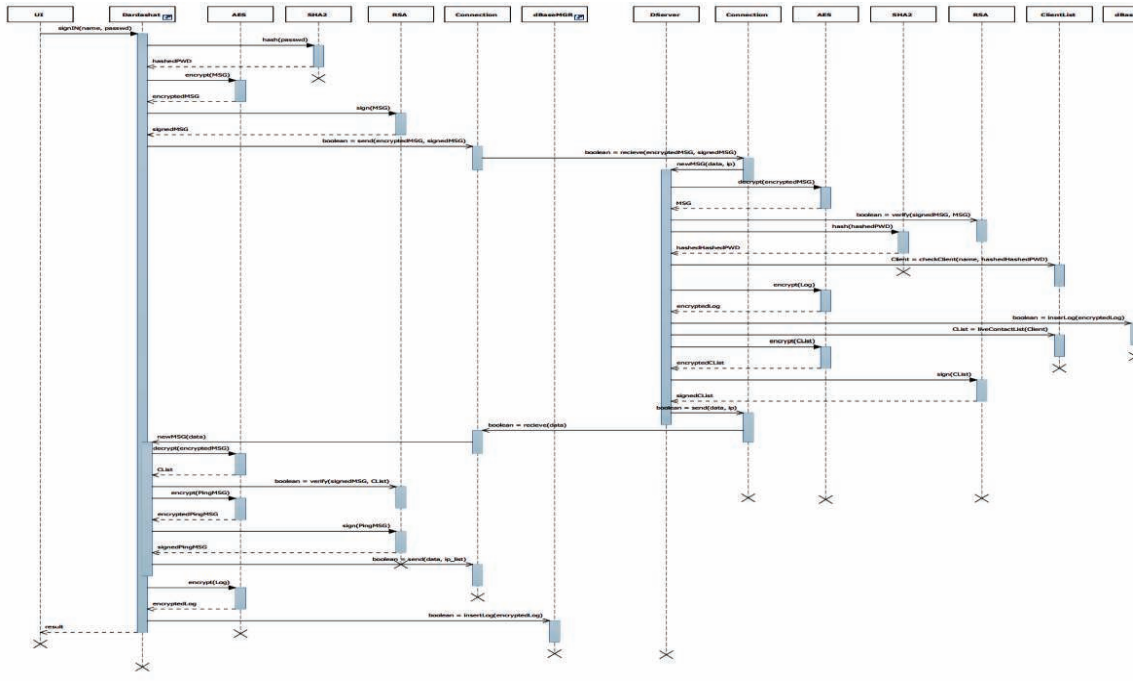
يتلقى العميل رسالة الملقم و يقوم الكلاس Dardashat بفك التشفير و التحقق من صحة الرسالة ؛ ثم يقوم بتوليد مفاتيح التشفير – علني و خاص – للعميل بطول 3072 بتا باستخدام الكلاس RSA ؛ ثم يتم تشفير المفتاح العلني الخاص بالعميل الذي تم توليده و رقمه الفريد الذي سبق استلامه من الملقم ، ثم

يوقع رقميا و يتم ارساله للملقم. يتلقى الملقم الرسالة و يقوم بفك التشفير و التحقق كالمعتاد ؛ ثم يقوم الكلاس DServer بتحديث بيانات العميل – أي اضافة مفتاح التشفير العلني الخاص به و المستلم من العميل – باستخدام الدالة updateClient() و هي دالة عضو في الكلاس ClientList كما يتم أيضا تحديث البيانات في قاعدة البيانات ثم يتم تشفير رسالة بنجاح العملية و يتم توقيعها ثم ارسالها للعميل الذي يتلقى الرسالة و يفك التشفير و يقوم بعملية التحقق كالمعتاد ثم يقوم الكلاس Dardashat بابلاغ واجهة المستخدم بنجاح العملية و بالتالي تعرض واجهة المستخدم النتيجة للمستخدم.

2.3.3 تسجيل الدخول Sign in :

الشكل (3.4) يوضح مخطط التسلسل Sequence diagram الخاص بعملية تسجيل الدخول ؛ يقوم المستخدم بادخال اسم المستخدم و كلمة السر ؛ تتلقى واجهة المستخدم UI هذه البيانات و تقوم بتمريرها لكلاس Dardashat ؛ الذي يقوم أولا بتوليد الهاش من كلمة السر بواسطة الكلاس SHA2 ؛ بعد ذلك يتم تشفير رسالة تتضمن اسم المستخدم و هاش كلمة السر لتسجيل الدخول ، و يتم التشفير باستخدام كلاس AES ثم يتم توقيع الرسالة رقميا باستخدام كلاس RSA ؛ بعد ذلك يتم ارسال الرسالة للملقم.

شكل (3.4) : مخطط التسلسل الخاص بعملية تسجيل الدخول



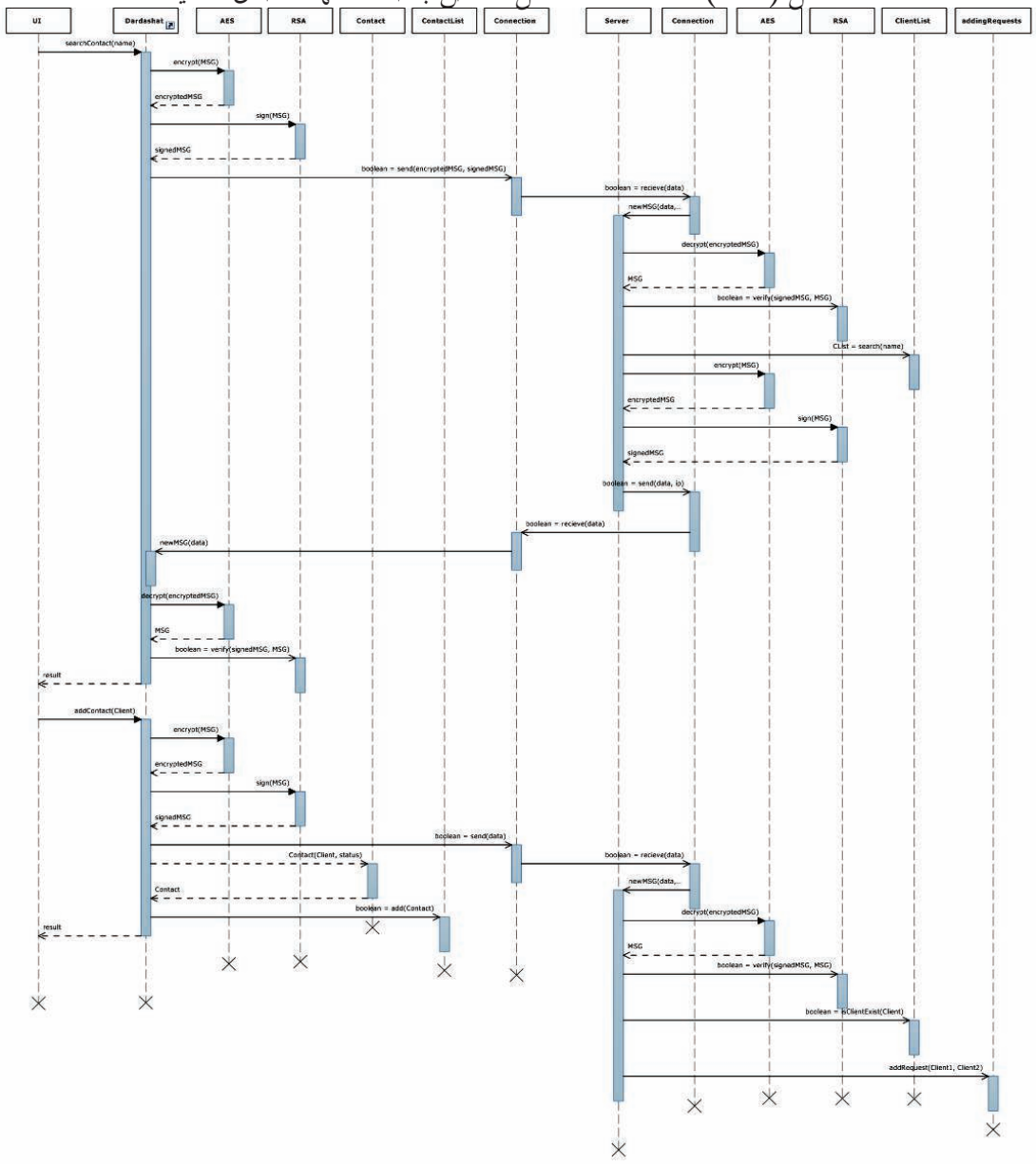
يقوم الملقم – أي كلاس DServer – بتلقي الرسالة و فك تشفيرها و التحقق من صحتها ؛ ثم يتم توليد هاش من هاش كلمة السر الذي تم تلقيه من العميل ؛ ثم يتم التحقق من صحة الاسم و هاش (هاش كلمة السر) عن طريق الدالة checkClient() و هي عضو في الكلاس ClientList ، ترجع هذه الدالة مرجعا reference للكائن Client ؛ أو ترجع القيمة Null و التي تعني أن هذا المستخدم غير موجود ؛ بعد ذلك يتم ادراج سجل عملية (log) في قاعدة البيانات بعد تشفيره ؛ بعد ذلك يتم استخراج قائمة بجهات الاتصال النشطة التي تخص المستخدم و تتضمن هذه القائمة عناوين IP الخاصة بهم و يتم تشفيرها و ارسالها للعميل.

يتلقى العميل رد الملقم و يتم فك التشفير و التحقق كالمعتاد ثم يتم ارسال رسائل ping بعد تشفيرها لكل جهات الاتصال النشطة للتحقق من أنها نشطة (online) و لابلاغها بأن المستخدم قد قام بتسجيل الدخول ؛ ثم يتم تشفير سجل الدخول (log) و حفظه في قاعدة البيانات مع ابلاغ واجهة المستخدم بنجاح تسجيل الدخول ليتم عرض النافذة الرئيسية للمستخدم.

3.3.3 إدارة جهات الاتصال (اضافة) Add Contact:

الشكل (3.5) يوضح مخطط التسلسل الخاص بإدارة جهات الاتصال – اضافة جهة اتصال جديدة – ؛ يقوم المستخدم باختيار اضافة جهة اتصال جديدة، يقوم كلاس واجهة المستخدم UI بفتح نافذة ليكتب داخلها المستخدم اسم جهة الاتصال ؛ ثم يقوم بتمريرها إلى كلاس Dardashat الذي يقوم بتجهيز الرسالة ثم –باستخدام الكلاسات الخاصة بذلك – يقوم بتشفيرها و توقيعها رقميا ثم يقوم بارسالها إلى الملقم.

شكل (3.5) : مخطط التسلسل الخاص باضافة جهة اتصال جديدة



يتلقى الملقم الرسالة و يقوم كلاس DServer بفك التشفير و التحقق من صحة الرسالة ثم يستخدم الدالة search() و هي عضو في الكلاس ClientList تعيد هذه الدالة سلسلة تضم اسم مستخدم أو أكثر (أو حتى خالي إذا لم تجد شيئا) ؛ يتم تشفير النتيجة و توقيعها و ارسالها للعميل. من جانبه يتلقي العميل الرسالة و بعد فك التشفير و التحقق يتم ارسالها لواجهة المستخدم ليتم عرضها للمستخدم. يختار العميل جهة الاتصال التي يريد ليتم اضافتها ؛ يتم ارسال الاسم لكلاس Dardashat باستخدام الدالة

addContact() ؛ يقوم الكلاس Dardashat بتجهيز رسالة تضم اسم جهة الاتصال ثم تشفيرها و توقيعها رقميا و ارسالها للملقم.

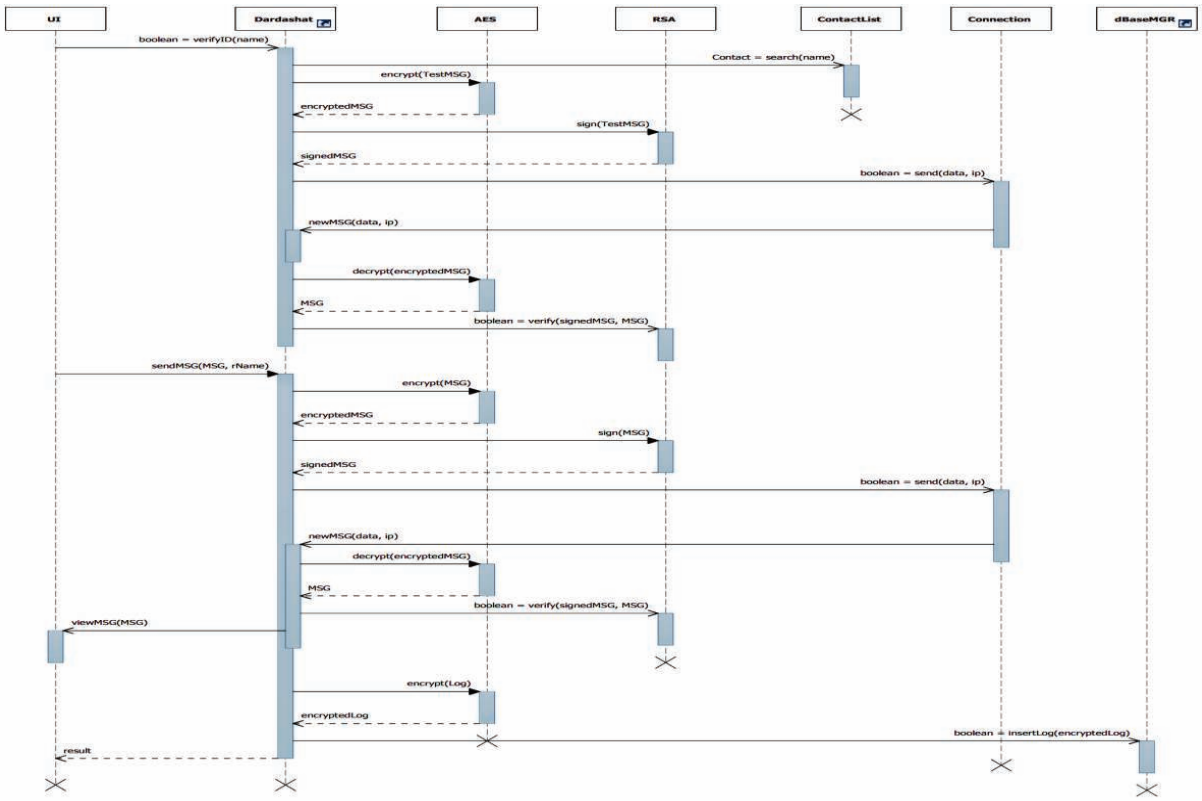
يقوم الملقم و كالمعتاد بفك التشفير و التحقق من صحة الرسالة ثم يقوم أولا بالتحقق من وجود العميل (جهة الاتصال) بواسطة الدالة isClientExist() فاذا كان موجودا في القائمة يتم اضافة طلب اضافة في الكلاس RequestPending الذي يقوم بارسال طلبات الاضافة لجهات الاتصال المطلوبة. في هذه الاثناء يقوم العميل بانشاء كائن من الكلاس Contact يمثل هذا الكائن جهة الاتصال المطلوب اضافتها و يتم اضافته لقائمة جهات الاتصال لدى العميل ContactList ثم يتم ابلاغ العميل بنجاح العملية.

4.3.3 اجراء محادثة :

الشكل (3.6) يوضح مخطط التسلسل الخاص بحالة اجراء محادثة ؛ عندما يختار المستخدم الشخص (جهة الاتصال) الذي يريد محادثته يقوم كلاس UI بفتح نافذة صغيرة لاجراء المحادثة إذا كان الشخص المطلوب نشط حاليا (online) و يتم أولا التحقق من الهوية ؛ حيث يقوم الكلاس Dardashat باعداد رسالة اختبار ثم تشفيرها و توقيعها ثم ارسالها للشخص المعني ؛ يقوم البرنامج على جهاز الشخص المعني بتلقي الرسالة و فك تشفيرها و التحقق من صحتها ثم يرد عليها مع تشفير الرد و توقيعه.

بعد تلقي الرد يتم التحقق من صحته ؛ إذا لم تكن النتيجة ايجابية يتم تنبيه العميل إلى عدم نجاح التحقق من صحة هوية الشخص المطلوب و أنه قد يكون هناك انتهاك لأمن المعلومات. أما غير ذلك فيمكن أن تبدأ المحادثة ، حيث يكتب العميل الرسالة و يضغط زر الارسال فيقوم كلاس واجهة المستخدم UI بتمريرها الى الكلاس Dardashat الذي يقوم بتشفيرها و توقيعها ثم ارسالها للشخص المعني ، و كذلك عندما يتلقى رسالة منه يتم فك التشفير و التحقق ثم ترسل لكلاس واجهة المستخدم حتى يتم عرضها للمستخدم ؛ في أثناء ذلك يتم حفظ سجل للمحادثة على قاعدة البيانات المصغرة بعد تشفيره.

شكل (3.6) : مخطط التسلسل الخاص بعملية اجراء محادثة



و هكذا تم اعداد مخططات تسلسل Sequence Diagrams لكل حالات الاستخدام السابق ذكرها في الباب الثاني ؛ و قد تم ذكر عدة نماذج منها أعلاه على سبيل الاختصار.

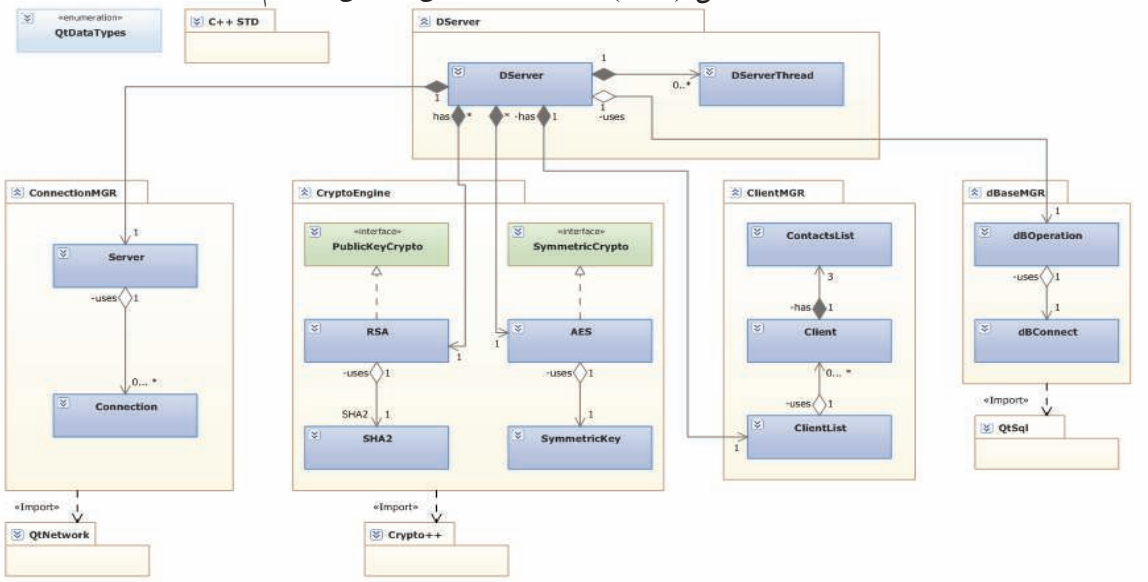
4.3 مخططات الكلاس Class Diagrams :

بعد تجهيز مخططات التسلسل التي توضح كلاسات البرنامج و تفاعلها معا لانجاز المطلوب ؛ في هذه الفقرة يتم عرض مخططات الكلاس التي توضح مكونات البرنامج بشكل تفصيلي أي كلاساته و علاقات هذه الكلاسات في ما بينها ؛ أي العلاقة بين كل كلاس و آخر هل هي استخدام فقط ؛ أم انشاء (ملكية) و استخدام ؛ أم هي توارث ؛ مع تحديد عدد النسخ المستخدمة أو المملوكة من كل كلاس. تحديد علاقات الكلاسات ببعضها و مسئولية كل واحد منها له أهمية كبيرة لا يتسع المجال لتفصيلها هنا.

1.4.3 مخطط كلاس الملقم Server Class Diagram :

الشكل (3.7) يوضح مخطط الكلاس الخاص بالملقم ؛ يظهر فيه جميع مكونات الملقم و داخل كل مكون تظهر الكلاسات التي يتكون منها مع علاقتها مع بعضها و عدد النسخ التي يمتلكها أو يستخدمها كل كلاس من الآخر، كما يتضح أن الكلاس DServer هو الواجهة (facade) التي سبق الحديث عنها حيث يربط جميع المكونات معا و يعزلها عن العالم الخارجي بشكل تام. أيضا يوضح المخطط المكتبات الخارجية التي يتم استخدامها في البرنامج مثل مكتبة Crypto++ التي تستخدمها كلاسات التشفير بالاضافة إلى مكتبات Qt متنوعة و مكتبات C++ القياسية.

شكل (3.7) : مخطط الكلاس الخاص بالملقم



2.4.3 مخطط كلاس العميل Client Class Diagram :

الشكل (3.8) يوضح مخطط الكلاس الخاص بالعميل و يتضمن واجهة المستخدم الرسومية ثم كلاس Dardashat الذي يعزل الواجهة الرسومية عن مكونات البرنامج الداخلية ؛ كما يوضح المكونات الداخلية بالتفصيل بشكل مشابه للمخطط السابق.

5.3 وصف كلاسات البرنامج :

في هذه الفقرة يتم تحديد وظيفة كل كلاس تم ذكره في مخططات الكلاس آنفة الذكر باختصار و لن يتم تفصيل المكونات الداخلية (عناصر البيانات – المتغيرات – و العمليات – الدوال – و التي يوضح جزء منها مخططات التسلسل المذكورة أعلاه) و ذلك من باب الاختصار و منعا للاطالة ؛ سيتم عرض كلاسات مخطط كلاس الملقم أولا ثم يتبع بكلاسات مخطط العميل مع عدم تكرار الكلاسات المشتركة بينهما.

1. كلاس DServer (موجود في نسخة الملقم فقط) :

من أهم مكونات الملقم حيث يمثل الواجهة (facade) – كما سبقت الاشارة لذلك ؛ لهذا الكلاس وظيفتان أساسيتان : أولاهما يمثل الرابط أو الصمغ الذي يربط و يلصق كل مكونات البرنامج الداخلية معا و يعمل كالمايسترو الذي يقوم بتنسيق العمل بين جميع الأطراف ؛ الوظيفة الأخرى هي عزل مكونات البرنامج عن البيئة الخارجية. لهذا التصميم (façade pattern) فوائد ملموسة من أهمها تسهيل عملية تطوير أو صيانة البرنامج ؛ فعند الحاجة لتعديل كلاس ما لن نحتاج إلا لتعديل الكلاس المطلوب تعديله و تعديل كلاس الواجهة أي DServer بدون الحاجة لتعديل كامل البرنامج ؛ بالاضافة لذلك لن يؤثر هذا التعديل على الواجهة الخارجية للبرنامج.

2. كلاس DserverThread (في الملقم فقط):

هذا الكلاس يمثل مسالك التنفيذ (Thread) التي يتم انشاؤها مع كل عملية اتصال جديدة و ذلك حتى يتم معالجها في مسلك منفصل حتى يتفرغ الملقم لاستقبال مزيد من الاتصال أثناء معالجته للاتصالات الحالية مما يحسن أداء النظام بشكل كبير.

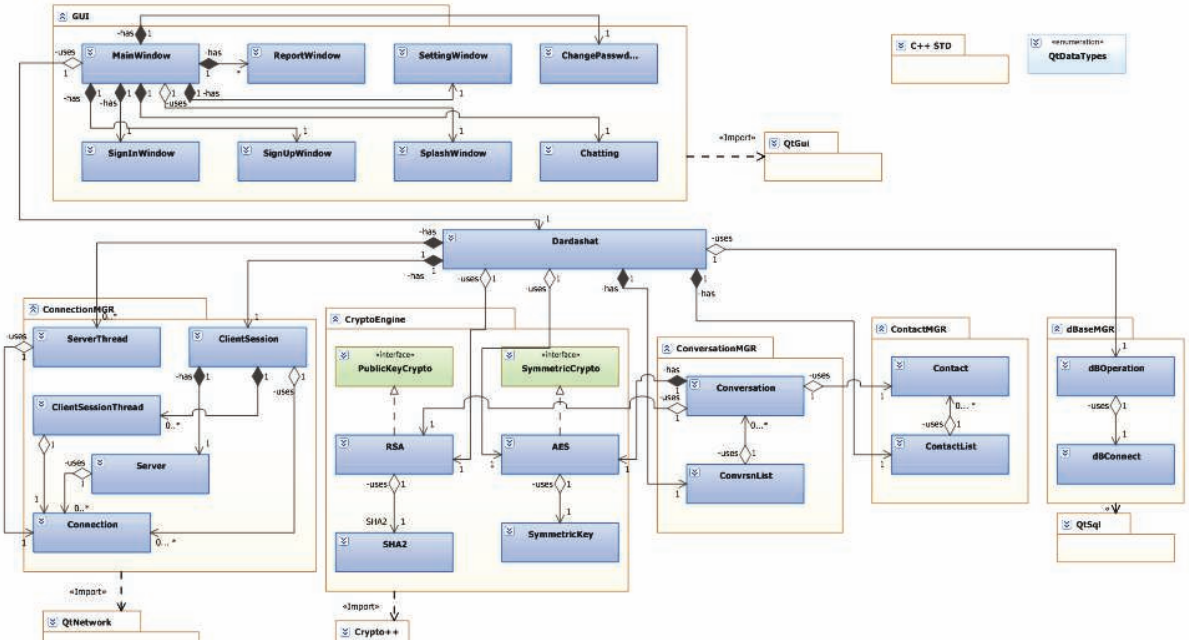
3. كلاس PublicKeyCrypto(أحد كلاسات CryptoEngine في نسختي العميل و الملقم) :

هذا الكلاس هو واجهة Interface يضم الخصائص العامة لخوارزميات التشفير بالمفتاح العلني و التي تقوم كلاساتها بالوراثة منه ، ثم كل كلاس يرث هذه الواجهة يضم التفاصيل الخاصة بخوارزمية التشفير التي يمثلها.

4. كلاس RSA (أحد كلاسات CryptoEngine في نسختي العميل و الملقم) :

يمثل هذا الكلاس خوارزمية RSA للتشفير بالمفتاح العلني و يتم استخدامه لتوليد الهوية الرقمية لكل مستخدم (بما فيها البرنامج نفسه) ؛ كما يستخدم في توقيع الرسائل رقميا و في عملية التحقق من صحة التوقيع و تبادل مفاتيح الجلسات.

شكل (3.8) : مخطط الكلاس الخاص بالعميل



5. كلاس SHA2 (أحد كلاسات CryptoEngine في نسختي العميل و الملقم):

يمثل هذا الكلاس خوارزمية SHA2 لتوليد الهاش (الاختزال) و يتم استخدامه بواسطة الكلاس السابق (RSA) في عمليتي التوقيع و التحقق من صحة التوقيع.

6. كلاس SymmetricCrypto (أحد كلاسات CryptoEngine في نسختي العميل و الملقم):

هذا الكلاس واجهة Interface يضم الخصائص العامة لخوارزميات التشفير بالمفتاح المتناظر حيث تقوم الكلاسات التي تمثل هذه الخوارزميات بوراثة هذا الكلاس مع اضافة التفاصيل الخاصة المتعلقة بالخوارزمية التي تمثلها.

7. كلاس AES (أحد كلاسات CryptoEngine في نسختي العميل و الملقم):

هذا الكلاس يمثل خوارزمية AES للتشفير بالمفتاح المتناظر ؛ و هو من أهم مكونات البرنامج حيث يستخدم في تشفير كل الرسائل المتبادلة بين المستخدمين لضمان سريتها.

8. كلاس SymmetricKey (أحد كلاسات CryptoEngine في نسختي العميل و الملقم):

وظيفة هذا الكلاس توليد مفاتيح التشفير المتناظر التي يستخدمها الكلاس AES في عمليتي التشفير و فك التشفير.

9. كلاس ContactList (أحد كلاسات ClientMGR في نسخة الملقم فقط):

يقوم هذا الكلاس بحفظ قائمة جهات الاتصال للعميل و طلبات الاضافة و ادارتها بين العميل و جهات الاتصال التي يريد اضافتها.

10. كلاس Client (أحد كلاسات ClientMGR في نسخة الملقم فقط) :

هو الكلاس الذي يمثل العميل ؛ حيث يحتوي على بياناته و يوفر مجموعة من الدوال للتعامل معها.

11. كلاس ClientList (أحد كلاسات ClientMGR في نسخة الملقم فقط) :

يضم قائمة مرجعية لكل العملاء لتسهيل الوصول لهم و بالتالي التعامل معهم حسب الحاجة.

12. كلاس dBOperation (أحد كلاسات dBaseMGR في نسختي العميل و الملقم) :

يقوم هذا الكلاس بتلقي كل الطلبات الموجهة لقاعدة البيانات بحيث يقوم بارسالها ثم يقوم بارجاع النتائج

13. كلاس dBConnect (أحد كلاسات dBaseMGR في نسختي العميل و الملقم) :

يوفر هذا الكلاس الاتصال المطلوب بقاعدة البيانات حتى يستخدمه الكلاس السابق في توجيه الطلبات لقاعدة البيانات.

14. كلاس Server (أحد كلاسات ConnectionMGR في نسختي العميل و الملقم) :

هذا الكلاس هو مقبس الملقم (Server Socket) الذي يتلقى طلبات العملاء عبر الشبكة.

15. كلاس Connection (أحد كلاسات ConnectionMGR في نسختي العميل و الملقم) :

هذا الكلاس هو الذي يقوم بتوفير الاتصال بالشبكة ؛ و يتم استخدامه بواسطة بقية كلاسات الشبكة الأخرى الأعلى منه في ConnectionMGR.

16. كلاس Dardashat (في نسخة العميل فقط) :

من أهم كلاسات نسخة العميل حيث يعزل المكونات الداخلية عن واجهة المستخدم ؛ كما يقوم بربط هذه المكونات معا و التنسيق بينها لانجاز المهمة.

17. كلاس MainWindow (أحد كلاسات GUI في نسخة العميل فقط) :

الكلاس الرئيسي في واجهة المستخدم الرسومية حيث يقوم بعرض النافذة الرئيسية للبرنامج و تحوي مجموعة من الكائنات الرسومية مثل الأزرار و القوائم و غيرها ؛ و هو يملك كل كلاسات الواجهة الرسومية و كل كلاسات واجهة المستخدم – بشكل عام – تقوم بعرض النوافذ التي تضم كائنات رسومية مختلفة، كما تقوم باستقبال طلبات المستخدم و ارسالها لكلاس Dardashat ثم استلام النتائج منه – أو رسائل حدوث خطأ – و عرضها للمستخدم بالشكل المطلوب.

18. كلاس ReportWindow (أحد كلاسات GUI في نسخة العميل فقط) :

كلاس نافذة التقارير الذي يقوم بعرضها بالتنسيق المناسب .

19. كلاس SettingWindow (أحد كلاسات GUI في نسخة العميل فقط) :

يعرض نافذة اعدادت الاتصال بالملقم.

20. كلاس SignInWindow (أحد كلاسات GUI في نسخة العميل فقط) :

كلاس نافذة تسجيل الدخول للبرنامج.

21. كلاس SignUpWindow (أحد كلاسات GUI في نسخة العميل فقط) :

كلاس نافذة انشاء حساب جديد.

22. كلاس SplashWindow (أحد كلاسات GUI في نسخة العميل فقط) :

هذه الكلاس يمثل نافذة مؤقتة تظهر للمستخدم لعدة ثوان أثناء بدء تشغيل البرنامج حتى لا يشعر بالملل و ذلك حتى يقوم البرنامج بتحميل المعلومات اللازمة و تهيئة الاتصال و ما شابه.

23. كلاس ChangePasswdWindow (أحد كلاسات GUI في نسخة العميل فقط) :

نافذة تغيير كلمة السر الخاصة بالمستخدم.

24. كلاس ChattingWindow (أحد كلاسات GUI في نسخة العميل فقط) :

نافذة اجراء المحادثات بين المستخدمين.

25. كلاس Conversation (أحد كلاسات ConversationMGR في نسخة العميل فقط) :

وظيفة هذا الكلاس هو ادارة أي محادثة بين العميل و جهة الاتصال التي يقوم بمحادثتها و يتضمن مفاتيح التشفير و التوقيع الرقمي لكل محادثة بالاضافة إلى سجل المحادثات.

26. كلاس ConvrsnList (أحد كلاسات ConversationMGR في نسخة العميل فقط) :

يضم قائمة بكل المحادثات النشطة حاليا لتسهيل ادارتها و التعامل معها.

27. كلاس Contact (أحد كلاسات ContactMGR في نسخة العميل فقط) :

يمثل جهة الاتصال حيث يتضمن بيانات جهة الاتصال و الدوال المخصصة للتعامل معها.

28. كلاس ContactList (أحد كلاسات ContactMGR في نسخة العميل فقط) :

يتضمن قائمة بكل جهات الاتصال الخاصة بالعميل و ذلك لتسهيل ادارتها و التعامل معها.

29. كلاس ClientSession (أحد كلاسات ConnectionMGR في نسخة العميل فقط) :

و هو الكلاس المسئول عن تهيئة جلسة الاتصال بين المستخدمين أي تهيئة شبكة الند – للند و تأمين الاتصال (أي التحقق من الطرف الآخر و تبادل المفاتيح).

30. كلاس ClientSessionThread (أحد كلاسات ConnectionMGR في نسخة العميل فقط) :

يستخدم هذا الكلاس لانشاء مسلك تنفيذ (Thread) منفصل لتهيئة كل جلسة اتصال بشكل مستقل و ذلك لتحسين الأداء.

31. كلاس ServerThread (أحد كلاسات ContactMGR في نسخة العميل فقط) :

يستخدم لانشاء مسلك تنفيذ منفصل عند الحاجة للاتصال بالملقم و هو كذلك يسهم بشكل كبير في تحسين أداء البرنامج.

6.3 الخلاصة :

في هذا الباب تم عرض مكونات البرنامج بشكل عام ؛ نسخة الملقم و نسخة العميل ؛ ثم عرضت مجموعة من مخططات التسلسل Sequence Diagram التي توضح تسلسل معالجة و تنفيذ طلبات المستخدم بواسطة مكونات البرنامج ؛ ثم عرض مخططي الكلاس لكلا من الملقم و العميل ، مع عرض سريع و مختصر لكل كلاسات البرنامج و التي بلغ عددها 31 كلاسا تقريبا. في الباب التالي (الباب الرابع) يتم مناقشة بعض جوانب التنفيذ و التطبيق العملي للبرنامج بناء على التصميم الذي تم تقديمه في هذا الباب.

الباب الرابع

التطبيق العملي و نوافذ البرنامج

1.4 تمهيد :

في هذا الباب يقوم الباحث بمناقشة و عرض بعض المسائل الفنية المتعلقة بالجانب العملي من النظام بشكل مختصر ؛ ثم يتم عرض نوافذ البرنامج (واجهة المستخدم) مع تقديم شرح مبسط لها ؛ ثم عرض بعض الاختبارات التي أجراها الباحث على النظام ؛ و قد تجنب الباحث ذكر نماذج تفصيلية من شفرة النظام في هذا الباب تجنبا للاطالة إلا أن القارئ بامكانه الرجوع لاحقا لملاحق البحث التي تضم نماذج من شفرة النظام.

2.4 لغة البرمجة و أدوات التطوير :

اختار الباحث – كما سبقت الاشارة لذلك من قبل – استخدام لغة سي بلس بلس C++ في تطوير البرنامج، و ذلك لعدة مميزات من أهما المرونة المدهشة التي توفرها و مستوى سرعة الأداء الملحوظ في البرامج التي استخدمت في تطويرها.

كما تم استخدم أداة Qt-SDK ، و هي تتضمن محرر (Editor) سهل الاستخدام و صديق للمستخدم ، و تم تصميمها لربطها مع أي مترجم للغة يختاره المبرمج ، كما أنها مفتوحة المصدر و مرفق معها توثيق و شرح تفصيلي مع الأمثلة – شكرا يا مطوري Qt فقد جعلتم حياة المبرمجين أسهل !. – أضف إلى ذلك أنها تتضمن مكتبات برمجية (أي كلاسات و دوال) غنية و متنوعة و تم تصميمها بشكل جيد.

من الأمثلة المختصرة على الكلاسات التي توفرها و تم استخدامها بشكل مكثف في تطوير البرنامج:

- كلاس QString: تم تصميمه لتخزين و معالجة السلاسل الحرفية أي الحروف و الأرقام و الرموز ؛ يدعم هذا الكلاس ترميز "يونيكود Unicode" و بالتالي يمكنه التعامل مع كل اللغات التي تحتاج لترميز اليونيكود كالعربية مثلا ؛ حيث يتم تمثيل كل حرف بعدد 2 بايت ؛ مع توفير مجموعة كبيرة من الدوال لمعالجة هذه النصوص مثل تحويلها إلى ترميز آخر كترميز "آسكي Ascii"، أو لاجراء عمليات البحث أو الابدال أو الحذف و غير ذلك. تم استخدام هذا الكلاس في تخزين البيانات في الطبقة العليا من البرنامج و ذلك لتوفير دعم كامل للغة العربية.

- كلاس QByteArray : يسمح هذا الكلاس بتخزين البيانات علي شكل مصفوفة من "البايتات" ، و يمكن تخزين البيانات الخام به (Raw) أي على شكل ثنائي (Binary) بدون ترميز محدد، هذا الكلاس تم استخدامه عند التعامل مع مفاتيح التشفير و النصوص المشفرة و التي هي في الواقع سلسلة من البتات (Bits) العشوائية ؛ و بالتالي كان هذا الكلاس هو الأنسب لتخزينها ، كما يوفر هذا الكلاس مجموعة كبيرة من الدوال التي تستخدم في معالجة البيانات المخزنة به. تم استخدام هذا الكلاس لتخزين البيانات في الطبقات السفلى من البرنامج مثل مكتبة التشفير و كائنات الشبكة.

- كلاس QSqlDatabase : و هو أحد 16 كلاسا توفرها Qt للتعامل مع قواعد البيانات المختلفة.

- QLinkedList<T> : يستخدم هذا الكلاس في انشاء القوائم المتصلة و هو مصمم باستخدام القوالب (Templates) و بالتالي يمكنه تخزين أي نوع من أنواع البيانات البسيطة أو المركبة ؛ تم استخدامه في انشاء قوائم المستخدمين و قوائم جهات الاتصال الخاصة بكل مستخدم ؛ و تجدر الاشارة هنا إلى المرونة و السرعة التي توفرها السلاسل المتصلة عند اجراء عمليات البحث و الاضافة و الحذف كما توفر استخداما جيدا للذاكرة بخلاف المصفوفات التقليدية.

- كلاس QTcpSocket : و هو كلاس مهم تم استخدامه – مع بقية أشقائه في مكتبة الشبكات التي توفرها Qt لانشاء المقابس "السوكت" و التعامل مع الشبكة على مستوى طبقتي النقل و الشبكة (Transport & Network Layers).

و استخدام هذه الكلاسات و غيرها إما بشكل مباشر أو توريثها لكلاس جديد يتضمن خصائص الكلاس الأساس و يضيف إليه خصائص جديدة.

3.4 خوارزميات التشفير :

قام الباحث بكتابة مكتبة برمجية للربط الديناميكي باسم CryptoEngine.dll و تحوي هذه المكتبة كلاسات خوارزميتي AES للتشفير المتناظر و RSA للتشفير غير المتناظر (التشفير بالمفتاح العلني) و هما خوارزميتان قياسيتان كما سبق ؛ بالاضافة إلى كلاسات أخرى؛ هذه المكتبة تم بناءها اعتمادا على مكتبة برمجية مفتوحة المصدر و معتمدة اسمها Crypto++ و هي مكتبة ضخمة تضم أغلب خوارزميات التشفير المعروفة ؛ سبقت الاشارة لاسباب اختيارها و استخدامها و لا مجال لتكرار ذلك هنا ؛ و من الجدير

بالذكر أن الفائدة التي تضيفها هذه المكتبة أمران: أولهما أن مكتبة CryptoEngine توفر دعما لأنواع البيانات التي توفرها مكتبة Qt مثل QString و غيرها من الكلاسات المذكورة أعلاه مما يتيح للمبرمجين الذين يستخدمون Qt استخدامها في كتابة أكواد التشفير و تضمينها في برامجهم ؛ علما بأن Qt و حتى تاريخ كتابة هذا البحث لا تتضمن مكتبة شاملة للتشفير فلعل هذه المكتبة – CryptoEngine – تكمل جزءا من النقص الموجود. الأمر الأخر أن مكتبة CryptoEngine تقدم واجهة برمجية سهلة الأستخدام ؛ حيث أن مكتبة Crypto++ مكتبة ضخمة و شاملة إلا أنها تحوي الكثير من التفاصيل و التعقيدات الفنية و التي تحتاج أن يكون المبرمج ملما بأساسيات علم التشفير بشكل عميق ؛ و هو قد لا يتوفر للمبرمج الذي لا يتوفر له الزمن لدراسة التشفير بشكل موسع و لكنه يريد استخدام خوارزميات التشفير القياسية في برامجه بشكل سريع ؛ كما أن الباحث حاول جمع كل التفاصيل المتعلقة بخوارزمية التشفير المحددة في في كلاس واحد و توظيف مفهوم إعادة توصيف الدوال (Function Overloading) لتوفير واجهة غنية و متنوعة لكل كلاس بالاضافة إلى استخدال الدوال الساكنة (Static Function) لتسهيل بعض المهام بدون الحاجة لانشاء كائنات فعلية من الكلاس المعني.

4.4 معمارية الشبكة (Network Architecture) :

تم استخدام نموذج عميل/ملقم (client/server) كأساس لمعمارية الشبكة ؛ و لأسباب متعلقة بأمن المعلومات تم استخدام نموذج الند للند (peer-to-peer) عند اجراء المحادثات بين العملاء لاستبعاد أي تدخل للملقم في هذه العملية. كذلك تم استخدام بروتوكول (TCP/IP) لتوفير الاتصال و ذلك في طبقتي النقل و الشبكة (Transport & Network layers) في نموذج OSI ؛ أما في طبقتي التطبيقات و الجلسة فقد تم كتابة بروتوكول بسيط لتيسير انشاء و تأمين الاتصال (أي تبادل المفاتيح و التحقق من الطرف الآخر) و تبادل الرسائل بشكل آمن ؛ سواء كان ذلك بين العميل و الملقم أو بين عميل و عميل آخر.

5.4 تعدد المسالك (Multithread) :

بما النظام يعمل على الشبكة و يشغل ملقما لخدمة العملاء ، كان لابد من جعل الملقم قادرا على تلقى أكثر من اتصال في نفس اللحظة ؛ و تم ذلك بتطبيق مفهوم تعدد مسالك التنفيذ بحيث يتمكن الملقم من انشاء مسلك تنفيذ جديد عند تلقي الاتصال ؛ يقوم هذا المسلك بالتعامل مع الاتصال و في نفس الزمن يتفرغ المسلك الرئيسي (Main Thread) لتلقي اتصالات أخرى. لتنفيذ ذلك تم استخدام كلاس QThread من مكتبة Qt ؛ صمم هذا الكلاس حتى يتم توريثه لكلاس جديد مع إعادة تعريف (كتابة) دالة ()run ، هذه الدالة هي التي يتم تنفيذها عند تشغيل مسلك التنفيذ لذلك على المبرمج إعادة تعريفها ليضع داخلها الكود الذي يريد تنفيذه.

تم تنفيذ ذلك في كل من الملقم و العميل ؛ في الملقم تم انشاء كلاس DServerThread ، هذا الكلاس يرث (Inherit) كلاس QThread ؛ و تم اعادة تعريف الدالة المذكورة حتى يتم من خلالها استدعاء الدوال التي تتعامل مع الاتصال ؛ يتم انشاء مسلك جديد باستخدام الكلاس المذكور عند تلقي الملقم لاتصال جديد ؛ و بعد انتهاء مهمة التعامل مع الاتصال (أي اعطاء العميل ما يريد أو ابلاغه برسالة خطأ مع تحديث قاعدة بيانات النظام) يتم انهاء مسلك التنفيذ. في العميل تم تنفيذ نفس الفكرة لانشاء مسالك تنفيذ عند اجراء اتصال من عميل للملقم أو من عميل لعميل آخر و ذلك بغرض تحسين أداء البرنامج العميل و تجنبا لتجمد البرنامج أثناء اجراء الاتصال.

6.4 التعامل مع الأخطاء و الاستثناءات (Exception):

لابد لكل نظام يعمل على الشبكة من توفير آلية للتعامل مع الأخطاء بشكل جيد ؛ سواء كانت هذه الأخطاء متوقعة أو غير متوقعة (استثناءات) ؛ لتوفير ذلك تم اعداد جدول يتضمن قائمة من أنواع الأخطاء المتوقعة أي توصيف مختصر لكل خطأ مع اعطاء رقم فريد لذلك الخطأ ؛ هذا الجدول مشترك بين الملقم و العملاء (أي يوجد منه نسخة لدى الملقم و كذلك لدى العميل) و هو يصف الأخطاء بأنواعها المختلفة. يتم استخدام هذا الجدول من قبل الملقم لارسال رمز الخطأ – عند حدوثه – إلى برنامج المستخدم الذي يقوم بابلاغ المستخدم بتفاصيل الخطأ بناء على رقم الخطأ و الوصف المحدد له مسبقا ، كما يقوم

الملقم بتسجيل الخطأ في سجل الأخطاء داخل قاعدة البيانات الخاصة به. كذلك يقوم برنامج العميل بابلاغ المستخدم بأي أخطاء داخلية تحدث لدي تنفيذ أي عملية داخل البرنامج مثل فشل عملية التشفير أو عملية فك التشفير و نحو ذلك.

الأخطاء غير المتوقعة (الاستثناءات Exception) صعبة الاكتشاف و قد تسبب انهيار البرنامج بشكل متكرر مما يضعف استقراره و موثوقيته ؛ لذلك حاول الباحث التعامل معها عن طريق استخدام تقنية التعامل مع الاستثناءات (Exception Handling) التي توفرها لغة سي بلس بلس باستخدام دوال (try ... catch) و تم استخدامها في الملقم و في برنامج العميل، خاصة مكونات التشفير و الشبكة في كلاهما، حيث يتم التقاط الاستثناء و ابلاغ المستخدم به و بمكان حدوثه في البرنامج مع استمراره في العمل.

7.4 استخدام المؤشرات و ادارة الذاكرة :

المؤشرات (Pointers) هي أداة مهمة توفرها لغة سي بلس بلس و تتيح مرونة عالية في التعامل مع عناوين الذاكرة بشكل مباشر لادارتها بالشكل المطلوب ؛ بالرغم من ذلك فهي أداة خطيرة للغاية ! ؛ إذا لم يتم استخدامها بالطريقة الصحيحة و بحذر بالغ فقد تسبب مشاكل صعبة الاكتشاف و تتسبب في انهيار البرنامج أثناء التشغيل. استخدم الباحث المؤشرات بشكل مكثف في البرنامج للاستفادة من المرونة التي توفرها مع الحذر البالغ و الحرص على تجنب الأخطاء عند استخدامها ؛ و أشهر هذه الأخطاء هو محاولة الوصول (dereference) لمؤشر خالي (NULL Pointer) – أي يحوي القيمة 0 (صفر) – مما قد يسبب انهيار البرنامج ؛ لذلك تم الحرص – بشكل دقيق – على التحقق من قيمة أي مؤشر تم استخدامه قبل أي عملية (dereference) له للتحقق من أنه لا يحوي القيمة 0 (صفر).

كما تم استخدام المؤشرات لتحسين أداء البرنامج من ناحية و تحسين ادارة الذكرة من الناحية الأخرى:

تحسين أداء البرنامج من خلال تخزين عناوين الذاكرة الخاصة بالكائنات التي توجد منها نسخ متعددة كالكائنات التي تمثل العملاء (أي في جانب الملقم) و نحوها من الكائنات ؛ تخزين هذه العناوين في قائمة متصلة (Linked List) بحيث يسهل البحث عن عنوان الذاكرة للكائن الذي يمثل العميل المطلوب لاجراء عملية قراءة أو كتابة لبياناته، عملية البحث تتم بسرعة عالية نظرا لصغر حجم المؤشر في الذاكرة (يمثل

ببايت واحد تقريبا) مما يحسن أداء النظام لأن عملية البحث تتم بشكل متكرر مع كل عملية يقوم بها الملقم استجابة لطلب البرنامج العميل.

كائنات العملاء يتم انشاؤها على ذاكرة الكومة (Heap) و هذا النوع لابد للبرمج من القيام بتفريغها (تدميرها) عند الفراغ منها و ارجاعها للنظام و إلا قد تحدث مشكلة تسرب الذاكرة (Memory Leak) مع الوقت (و هذه احدى مشاكل التعامل مع المؤشرات) ؛ لذلك تستخدم نفس القائمة المشار إليها في ادارة الذاكرة حيث تقوم بشكل تلقائي بتدمير كل الكائنات التي تتضمن عناوين لها عند انتهاء البرنامج ؛ أو ازالة بعضها عند عدم الحاجة له أثناء التشغيل.

8.4 تبادل الرسائل بين الكائنات :

تحتاج كائنات البرنامج إلى التخاطب و تبادل الرسائل حتى تستطيع العمل معا لانجاز المهمة ! ؛ لذلك تم استخدام تقنية مدهشة تسمى (الاشارات و الشقوق : Signals & Slots) توفرها Qt لتمكين الكائنات من تبادل الرسائل مع بعضها البعض ؛ الفكرة هنا – باختصار – أن الكائن الأول يرسل اشارة (و هي دالة) يلتقطها كائن آخر من خلال شق (و هو أيضا دالة) ؛ أي يتم تنفيذ الدالة التي تمثل الشق (slot) عند ارسال الاشارة (signal) و لكن لابد أولا من ربط الاشارة و الشق معا باستخدام دالة ساكنة اسمها connect() ؛ يتم ارسال مؤشرات لكلا الكائنين مع أسماء الدوال أي الاشارة و الشق لهذه الدالة. سر نجاح هذه التقنية أن جميع كلاسات Qt ترث – بشكل مباشر أو غير مباشر – كلاس جذر اسمه QObject (و هو مشابه لكلاس Object في لغة جافا) مما يمكن من ربط هذه الكائنات معا باستخدام تقنية تعدد الأشكال (Polymorphism) مما لا يتسع المجال لتفصيله هنا.

تم استخدام هذه التقنية بشكل مكثف في البرنامج لأنهما عملية جدا و غير مكلفة بل و تيسر التخاطب بين أجزاء البرنامج و مسالك التنفيذ المختلفة داخل البرنامج ؛ و قد توظيفها بشكل كبير في الطبقة التي تضم مكونات الشبكة تحديدا بالاضافة إلى غيرها.

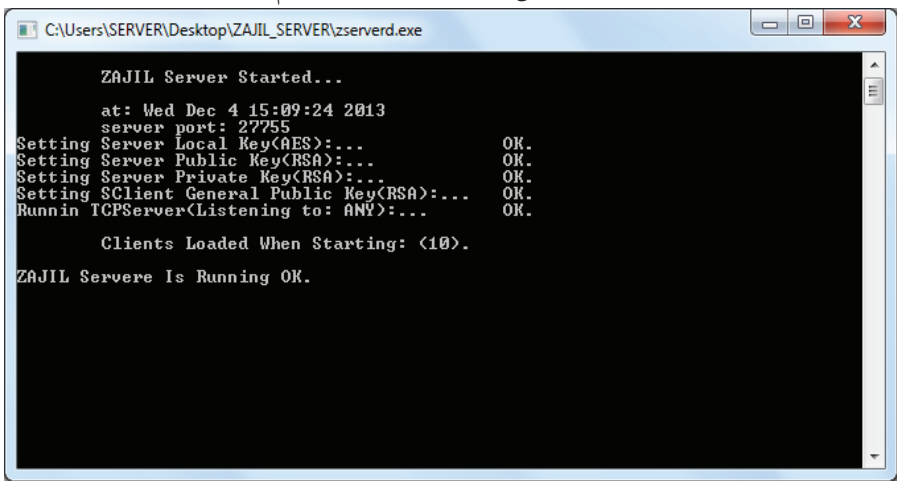
9.4 نوافذ النظام (واجهة المستخدم) :

في هذه الفقرة يتم استعراض نوافذ البرنامج بعد أن تم تطويره مع تقديم شرح مختصر لكل نافذة:

1. نافذة الملقم :

تظهر هذه النافذة – شكل (4.1) عند تشغيل الملقم، و تبين تاريخ بدء التشغيل كما تبين حالة الملقم بشكل عام – أي أنه قد تم تحميل مفاتيح الملقم بنجاح و أن الملقم مستعد لتلقى الاتصالات، كما يعرض اجمالي المستخدمين المسجلين و الذين تم تحميل بياناتهم بنجاح من قاعدة البيانات.

شكل (4.1) : نافذة الملقم



2. نافذة تمهيدية (Splash) :

تظهر هذه النافذة – شكل (4.2) – عند بدء تشغيل برنامج العميل لفترة قصيرة أثناء تحميل مكونات البرنامج.

شكل (4.2) : نافذة بدء تشغيل برنامج العميل

3. نافذة تسجيل الدخول:

أول نافذة تظهر للمستخدم ، شكل (4.3) – بعد النافذة السابقة – حتى يتمكن المستخدم من تسجيل الدخول.

شكل (4.3) : نافذة تسجيل الدخول

4. نافذة تسجيل حساب جديد :

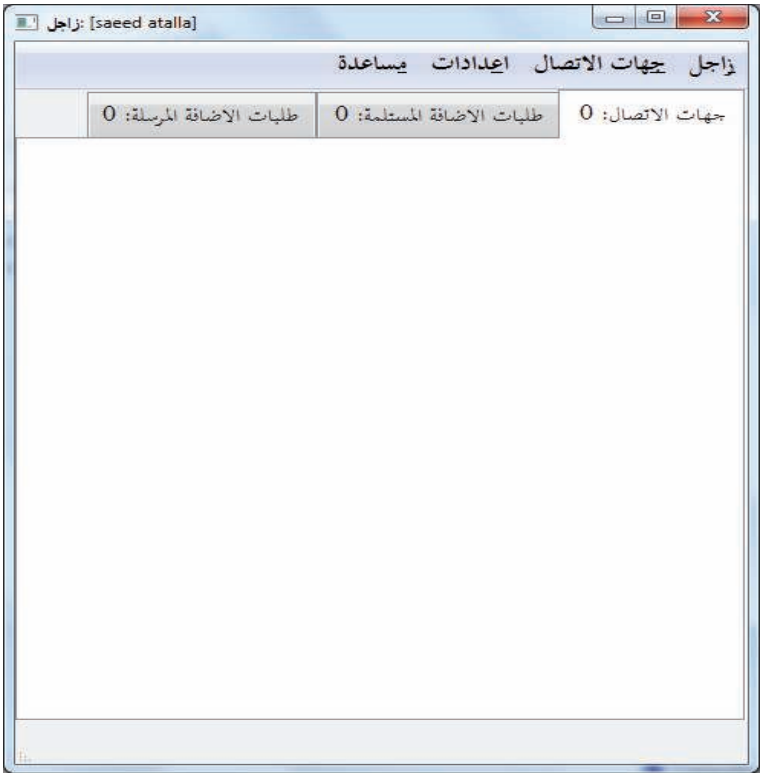
هذه النافذة – شكل (4.4) – يمكن الوصول لها قبل تسجيل الدخول من خلال النافذة السابقة، حيث يمكن من خلالها أن يقوم المستخدم بتسجيل حساب جديد له على النظام.

شكل (4.4) : نافذة تسجيل حساب جديد

5. النافذة الرئيسة :

النافذة الرئيسة للبرنامج – شكل (4.5) – و من خلالها يمكن الوصول لكل النوافذ الفرعية الأخرى، تظهر على هذه النافذة ثلاث قوائم: الأولى تضم أسماء جهات الاتصال (أي أصدقاء المستخدم) ؛ و إذا كان متصلا سيظهر باللون الأخضر و إلا باللون الرمادي ؛ أما القائمتين الأخرتين فتضمان قائمة طلبات الاضافة المستلمة من الآخرين و قائمة طلبات الاضافة التي أرسلها المستخدم لآخرين بالاضافة إلى قوائم علوية يمكن من خلالها تنفيذ مهام مختلفة.

شكل رقم (4.5) : النافذة الرئيسة لبرنامج العميل



6. نافذة تغيير كلمة السر :

من خلال هذه النافذة – شكل (4.6) يتمكن المستخدم من تغيير كلمة السر الخاصة به – إذا أراد ذلك.

شكل (4.6) : نافذة تغيير كلمة السر

7. نافذة التقارير :

النافذة – شكل (4.7) – تعرض خيارات كثيرة للمستخدم لاختيار التقرير الذي يرغب في عرضه بعد تحديد الفترة الزمنية المناسبة.

شكل (4.7) : نافذة التقارير

8. نافذة اعدادت الاتصال بالملقم :

تمكن هذه النافذة – شكل (4.8) – المستخدم من تهيئة اعدادات الاتصال بالملقم باختيار الاعدادات الافتراضية أو اعدادت أخرى.

شكل (4.8) : اعدادات الاتصال بالملقم

9. نافذة اضافة جهة اتصال :

في هذه النافذة – شكل (4.9) – يتمكن المستخدم من البحث عن مستخدمين آخرين للنظام لارسال طلبات اضافة لهم إذا أراد.

شكل (4.9) : نافذة البحث عن مستخدمين آخرين

10. نافذة مربع حوار :

النافذة – شكل (4.10) – نموذج لمربعات الحوار المختلفة التي تم استخدامها لابلاغ المستخدم بمعلومة مهمة أو سؤاله و نحو ذلك.

شكل : (4.10) : مربع حوار

11. نافذة حول البرنامج :

هذه النافذة – شكل (4.11) – تضم بعض المعلومات عن البرنامج.

شكل (4.11) :نافذة حول البرنامج

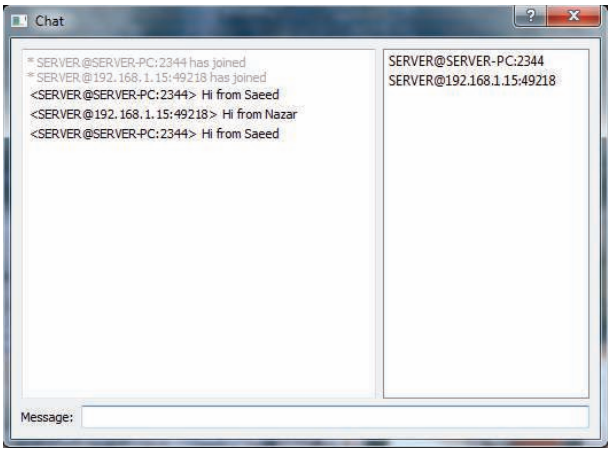
10.4 اختبار و تجربة البرنامج :

بعد الفراغ من كتابة البرنامج و تنقيحه قام الباحث باجراء اختبارين عليه ؛ للتأكد من تحقق عملية التأمين (أي تشفير البيانات و توقيعها رقميا) و ذلك كما يلي:

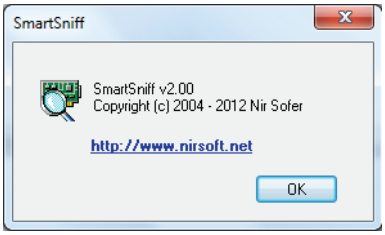
1. التجربة الأولي : التأكد من عملية التشفير :

تم استخدام نسخة مبدئية بسيطة من البرنامج (نموذج أولي Prortotype) – انظر شكل (4.12) – لا تتضمن خدمة التشفير ؛ تم تشغيل نسخة منه على جهاز الحاسوب، فيما تم تشغيل نسخة أخرى من نفس البرنامج على جهاز افتراضي (Virtual Machine) ؛ باستخدام محطة عمل (VMware Workstation , v.9.0.0) ؛ و تم اعداد شبكة محلية افتراضية للربط بين الجهازين الحقيقي و الافتراضي حتى يتم تبادل الرسائل بين نسختي البرنامج على الجهازين ؛ و ذلك بغرض استخدام أداة (برنامج خدمي) اسمه SmartSniff – انظر شكل (4.13) ؛ يقوم هذا البرنامج بالتقاط كل الحزم (packets) المارة عبر الشبكة و يعرضها على شكل تقرير مع معلومات تفصيلية – مثل عناوين الآي بي و طول الرسالة و وقت الارسال و نحو ذلك.

شكل (4.12) : نموذج أولي للبرنامج لا يقدم خدمة التشفير



شكل (4.13): معلومات عن أداة الشبكة SmartSniff

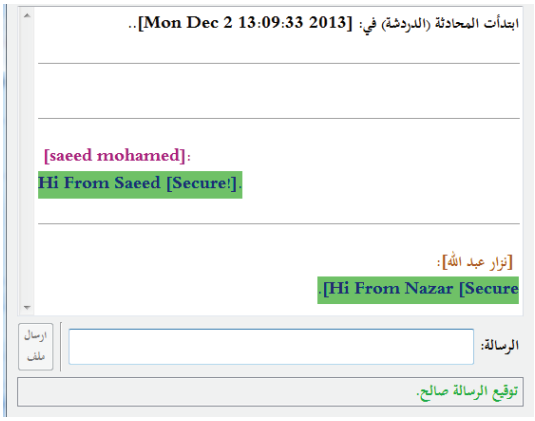


قام البرنامج الخدمي المذكور – و يستخدمه عادة مهندسي الشبكات لتحليل و اكتشاف المشاكل بالشبكات التي يعملون عليها – بالتقاط الرسائل المرسلة بين نستخي البرنامج المذكورتين ؛ كما في شكل (4.14) ؛ مما يثبت عمليا أن النسخة الأولية من البرنامج لا تقدم خدمة تشفير – و بالتالي يمكن لأي (هاكر) لديه الأدوات المناسبة أن يطلع على محتوى الرسائل التي يتم ارسالها بواسطة البرنامج من خلال الشبكة.

شكل (4.14) : جزء من محتوى رسالة غير مشفرة تم التقاطها

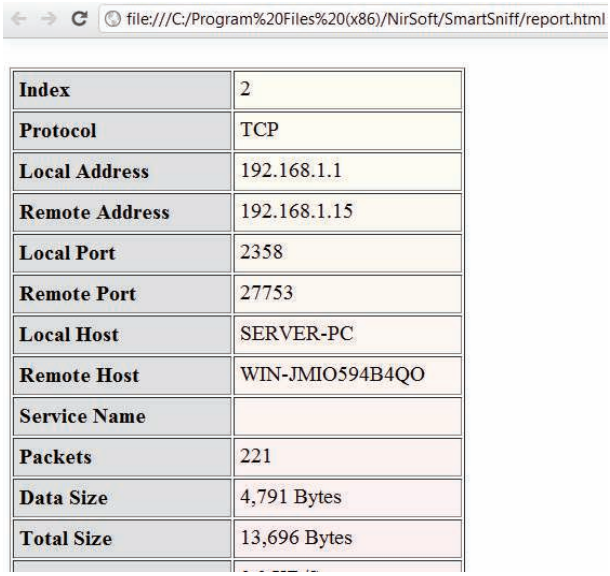
بعد ذلك تم تشغيل النسخة النهائية من البرنامج – و التي تقدم خدمة تأمين الرسائل – على نفس الشبكة ، انظر شكل (4.15) مع تشغيل أداة الشبكة المذكورة أعلاه و التي قامت بالتقاط الرسالة التي تم تشفيرها ، شكل (4.16) ، حيث ظهرت على شكل رموز ستعشرية (Hex)، و تم التحقق من أن هذه الرسالة فعلا تم ارسالها بواسطة البرنامج من خلال المعلومات التفصيلية التي يقدمها البرنامج الخدمي في تقريره ؛ شكل (4.17) و التي تبين البروتوكول المستخدم و عناوين آي بي لكلا الطرفين و المنافذ و وقت الارسال و غير ذلك من التفاصيل و التي تم التأكد من مطابقتها ؛ و بالتالي لن يتمكن الهاكر من الاطلاع على محتوى الرسالة – إلا لو عرف المفتاح بطبيعة الحال.

شكل (4.15) : نافذة حوار بين طرفي البرنامج



شكل (4.16) : جزء من محتوى رسالة مشفرة تم التقاطه

شكل (4.17) : جزء من تقرير أدة الشبكة حول الرسالة السابقة



Index	2
Protocol	TCP
Local Address	192.168.1.1
Remote Address	192.168.1.15
Local Port	2358
Remote Port	27753
Local Host	SERVER-PC
Remote Host	WIN-JMIO594B4QO
Service Name	
Packets	221
Data Size	4,791 Bytes
Total Size	13,696 Bytes

2. التجربة الثانية : التأكد من مقدرة البرنامج على اكتشاف عدم صحة توقيع رسالة مستلمة :

قام الباحث باجراء تعديل في شفرة البرنامج ، بحيث يتم تعديل بايت واحد فقط في الرسالة (أول بايت) بعد توقيعها و تشفيرها و قبل ارسالها ؛ بحيث يتم ارسالها مباشرة بعد التعديل ، و ذلك بافتراض أن الرسالة قد أرسلت سليمة و تم تعديل هذا البايت أثناء انتقالها عبر الشبكة – انظر شكل (4.18) و هو يوضح مقطعا من الشفرة حيث تم اضافة السطر المظلل رقم 83 ليقوم باستبدال حرف واحد فقط من الرسالة بعد تشفيرها بالحرف "A" ثم ارسال الرسالة.

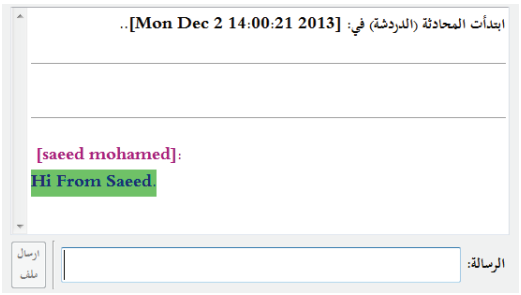
شكل (4.18) : مقطع من الشفرة تم تعديله لتغيير حرف في الرسالة بعد تشفيرها

conversation.cpp | Conversation::sendMessage(const QString, const QString)

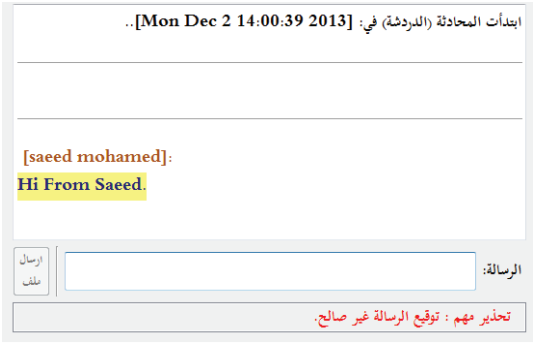
```
58      MSG = message;
59      MSG.append(MessageSeparatorToken);
60      MSG.append(m_mePtr->name());
61      MSG.append(MessageSeparatorToken);
62      if(isChattingStarted()) {...}
66      else {...}
71      cipher = encryptMSG(MSG);
72      if(cipher.isEmpty()) {...}
81      else
82      {
83              cipher.replace(0, 1, "A");
84              m_connection->sendMessage(cipher);
85      }
86  }
```

تم تشغيل البرنامج – بعد التعديل – و ارسال رسالة منه كما في شكل (4.19) ؛ و ينبغي أن تكون النسخة الأخرى من البرنامج لدى الطرف المستلم قادرة على اكتشاف حدوث التعديل بالرغم من أنه تم في بايت واحد فقط. و بالفعل قام البرنامج المستلم – و الذي لم يتم تعديله بأي شكل – باكتشاف حدوث خلل في الرسالة، شكل (4.20) ؛ و تنبيه المستخدم لوجود خلل في توقيع الرسالة ؛ و من الجدير بالذكر هنا أن الرسالة المستلمة لم يظهر بها أي تعديل و ذلك لأن البايت الذي تم تعديله في أول الرسالة هو في الواقع أول بايت في توقيع الرسالة الذي يرسل معها و يوضع في البداية أولا ثم الرسالة.

شكل (4.19) : نافذة الحوار لدى الطرف المرسل



شكل (4.20) : نافذة الحوار لدى الطرف المستلم



11.4 الخلاصة :

في هذا الباب تم مناقشة و استعراض بعض الجوانب الفنية المتعلقة بالتنفيذ العملي للنظام باختصار ؛ بالاضافة لاستعراض نوافذ البرنامج المختلفة بعد الفراغ من تطويره و تشغيله ثم تجربته و اختباره.

الخاتمة :

و هكذا نصل الى نهاية المطاف لهذا البحث المتواضع و الذي نرجو أن يكون إضافة و لو يسيرة للمكتبة العربية و للقارئ العربي في مجال التشفير و أمن المعلومات. و نسأل الله تعالى أن ينفعنا بما علمنا و أن ينفع بنا الاسلام و المسلمين وأن يعيننا على تعلم العلم وتبليغه إنه ولي ذلك والقادر عليه ؛ وأخيرا ما كان من خطأ فمن أنفسنا و الشيطان و ما كان من صواب فمن الله عز وجل.
وصلى اللهم وسلم على نبينا محمد وعلى آله وصحبه أجمعين. وآخر دعوانا أن **الحمد لله رب العالمين**.

النتائج :

1- تحقيق البرنامج لمواصفات المتطلبات – انظر الباب الثاني – و أهمها : تقديم مستوىً عال و قياسي من الأمان للرسائل المرسلة عبر الشبكة باستخدام خوارزميات التشفير القياسية و قواعد أمن المعلومات.
2- تأكيد أهمية المصادر المفتوحة في مجال تطوير البرمجيات ؛ خاصة مع حاجتنا في السودان لزيادة الاهتمام بهذا الحقل و محاولة ردم الفجوة الموجودة بيننا و بين كثير من الدول الأخرى في التوسع في توظيف علوم الحاسوب في مختلف التطبيقات ؛ خاصة التطبيقات المهمة و التي تحتاج لمستوى عال من الأمان.
3- أهمية البحث العلمي و دوره ايجاد حلول علمية و عملية لمشاكل واقعية باستخدام العلوم المتنوعة التي قام الباحثون بدراستها في الكلية ؛ و التي قد تسهم في تسهيل الحياة و تسخيرها لخدمة المجتمع بأقل التكاليف و بأيدي وطنية موثوق بها.

التوصيات و المقترحات:

1. تطوير مكتبة التشفير (CryptoEngine) لإضافة المزيد من الخوارزميات المهمة و الحديثة كخوارزمية الاختزال (الهاش) الآمن – الاصدارة الثالثة SHA3 التي تم اعتمادها مؤخرا.
2. اجراء مزيد من الاختبارات على المكتبة و البرنامج بواسطة أطراف أخرى مستقلة للتحقق من مستوى الأمان و غيره من المتطلبات.

3. تطوير نسخة من الملقم تعمل على نظام تشغيل لينكس (Linux) للاستفادة من مستوى الأمان الجيد الذي يوفره ؛ و تطوير نسخة من البرنامج (نسخة العميل) تعمل على نظم تشغيل أخرى مثل نظام أندرويد (Android) على الهواتف الذكية مع الالتزام بنفس مستوى الأمان.

4. اضافة مزيد من التحسينات و الوظائف المهمة مثل تحسين مستوى الأداء للملقم و تحسين الواجهة الرسومية لبرنامج العميل؛ و كذلك اضافة خدمات المكالمات الصوتية و مؤتمرات الفيديو، مع الالتزام بنفس المستوى من الأمان.

5. التوسع في دراسة الجانب النظري في علم التشفير و أمن المعلومات، و تلخيصه بشكل جيد حتى يتمكن القارئ العربي من الاطلاع على تفاصيل هذا الحقل المهم و الذي تقل فيه البحوث المكتوبة باللغة العربية.

Bibliography

[1] Ezust Alan and Ezust Paul, **Introduction to Design Patterns in C++ and Qt**, 2th Edition, Prentice Hall, 2011.

[2] McCullagh Declan, **How safe is instant messaging?**, visited in March2013, http://news.cnet.com/8301-13578_3-9962106-38.html

[3] Menezes, Oorschot, and Vanstone, **Handbook of Applied Cryptography**, 5th Edition, CRC Press, 1997.

[4] Milewski Bartosz, **C++ In Action**, RO Release.

[5] Molkentin Daniel, **The Book of Qt 4: The Art of Building Qt Applications**, 1st Edition, Open Source Press GmbH, 2006.

[6] National Institute of Standards and Technology, FIPS PUB 140-2, **SECURITY REQUIREMENTS FOR CRYPTOGRAPHIC MODULES**, March 2002.

[7] National Institute of Standards and Technology, FIPS PUB 180-4, **Secure Hash Standard (SHS)**, March 2012.

[8] National Institute of Standards and Technology, FIPS PUB 186-3, **Digital Signature Standard (DSS)**, Gaithersburg, June, 2009.

[9] National Institute of Standards and Technology, FIPS PUB 197, **Advanced Encryption Standard (AES)**, Nov. 2001.

[10] Ramnath Sarnath and Dathan Brahma, **Object-Oriented Analysis and Design**, Universi-ties Press and Springer.

[11] Schneier Bruce, **Applied Cryptography**, 2nd Edition.

[12] Stallings William, **Cryptography and Network Security**, 5th Edition, Prentice Hall, 2011.

[13] Wei Dai, **CryptoPP**, visited in :March 2013, http://www.cryptopp.com/wiki/index.html

أ. نماذج من شفرة نظام زاجل:

I. مكتبة CryptoEngine :

1. ملف celib.h:

```
// celib.h:
#ifndef CELIB_H
#define CELIB_H
#include "../CELIB/cedll.h"
#include "../CELIB/CryptoEngine_global.h"

namespace CELIB {

enum    OP_MODE {ECB = 1, CBC, CFB, OFB, CTR, GCM}; // Operation Modes.
enum    CIPHER_ALGORITHM {AES_ALGORITHM = 1, RSA_ALGORITHM, DSA_ALGORITHM};
enum    Padding { DefaultPadding = 1, NoPadding, PKCS7 };
//enum    KeyLength {MIN = 1, MAX, DEFAULT};
typedef CryptoPP::SHA256 SHA256;
typedef CryptoPP::SHA384 SHA384;
typedef CryptoPP::SHA512 SHA512;
typedef CryptoPP::SecByteBlock SByteArray;
typedef CryptoPP::Exception Exception;

class CELIB_EXPORT SymmetricCrypto : public QObject {
    Q_OBJECT
public:
    SymmetricCrypto();
    virtual const QByteArray encrypt(const QByteArray& plainText, bool * OK = 0) = 0;
    virtual const QByteArray decrypt(const QByteArray& cipherText, bool * OK = 0) = 0;
    virtual ~SymmetricCrypto();
    virtual QString className()const;
private:
    QString m_Name;
};

class CELIB_EXPORT PublicKeyCrypto : public QObject{
    Q_OBJECT
public:
    PublicKeyCrypto();
    virtual const QByteArray encrypt(const QByteArray& plainText, bool * OK = 0) = 0;
    virtual const QByteArray decrypt(const QByteArray& cipherText, bool * OK = 0) = 0;
    virtual const QByteArray sign(const QByteArray& msg, bool * OK = 0) = 0;
    virtual const bool Verify(const QByteArray& signature, const QByteArray &msg)= 0;
    virtual ~PublicKeyCrypto();
    virtual QString className()const;
private:
    QString m_Name;
};
}
#endif // CELIB_H
```

2. ملف aes.h :

```
//aes.h:
#ifndef AES_H
#define AES_H

#include <QObject>
#include "../CELIB/cedll.h"
#include "../CELIB/celib.h"
#include "../CELIB/SymmetricKey.h"
#include "../CELIB/CryptoEngine_global.h"
namespace CELIB {
class CELIB_EXPORT AES :public SymmetricCrypto{
    Q_OBJECT
public:
    AES(SymmetricKey &key, const OP_MODE mode);
    AES(SymmetricKey &key, const QByteArray &ivFromHex, const OP_MODE mode);
    AES(const AES& other);
    virtual ~AES();
    AES& operator=(const AES& other) ;
    bool operator==(const AES& other)const;
    //Encryption:
    const QByteArray encrypt(const QByteArray& plainText, bool * OK = 0);
    const QByteArray test(const QByteArray& pt);
    //Take source file path & destination file path as input:
    bool encrypt(QFile &sourceFullPath, QFile &destnFullPath);
    //Decryption:
    const QByteArray decrypt(const QByteArray& cipherText, bool * OK = 0);
    //Take source file path & destination file path as input:
    bool decrypt(QFile &sourceFullPath, QFile &destnFullPath);
    //key & iv:
    inline bool setKey(const SymmetricKey& key);
    inline bool setKey(const QByteArray& key); // key as Hex.
    inline bool setIV(const QByteArray& iv); // IV as Hex.
    inline void setOPMode(const OP_MODE mode);
    inline const SymmetricKey& currentKey()const;
    inline const QByteArray currentKeyHex() const; //return KEY as Hex.
    inline const QByteArray currentIVHex() const;
    virtual QString toString() const;
    QString currentOPMode() const;
    inline const byte& currentIV()const;
    virtual QString className() const;
signals:
    void error(QString err);

private:
    inline void encryptECB(const QByteArray& plain, QByteArray& cipher);
    inline void decryptECB(const QByteArray &cipher, QByteArray& plain);
    inline void encryptCBC(const QByteArray& plain, QByteArray& cipher);
    inline void decryptCBC(const QByteArray &cipher, QByteArray& plain);
    inline void encryptCFB(const QByteArray& plain, QByteArray& cipher);
    inline void decryptCFB(const QByteArray &cipher, QByteArray& plain);
    inline void encryptOFB(const QByteArray& plain, QByteArray& cipher);
    inline void decryptOFB(const QByteArray &cipher, QByteArray& plain);
    inline void encryptCTR(const QByteArray& plain, QByteArray& cipher);
    inline void decryptCTR(const QByteArray &cipher, QByteArray& plain);
    inline void encryptGCM(const QByteArray& plain, QByteArray& cipher);
    inline void decryptGCM(const QByteArray &cipher, QByteArray& plain, bool * integrity

    QString m_Name;
    SymmetricKey& m_Key;
    byte m_IV[CryptoPP::AES::BLOCKSIZE];
    OP_MODE m_Mode;
```

II. نسخة الملقم:

3. ملف: client.h :

```
#ifndef DCLIENT_H
#define DCLIENT_H

#include <QObject>
#include <QHostAddress>
#include <QStringList>
#include "CELIB/INCLUDE/CELIB/rsa.h"
#include "areqpending.h"
namespace ClientMGR
{
class Client : public QObject
{
    Q_OBJECT
public:
    enum STATUS{
        ONLINE,
        OFFLINE
    };
    enum GENDER{
        MALE,
        FEMAIL
    };
    explicit Client(long cid, QString name, QByteArray passwd
                    , QString email, QString gender
                    , QString location, QObject *parent = 0);
    virtual ~Client();
    const long cid()const;
    const QString name()const;
    const bool isPasswdCorrect(const QByteArray& passwd)const;
    const bool isPubKeyCorrect(const CELIB::RSApublicKey* othrPubKey);
    const QString email()const;
    const QString gender()const ;
    const QString location()const ;
    const QString status()const;
    const bool isOnline()const;
    const QHostAddress myIPAddress()const;
    const int myPort()const;
    const CELIB::RSApublicKey publicKey()const;
    const QByteArray publicKeyHex()const;
    const QString contactsList()const;
    const QString ReqPendingFromOthersToMe()const;
    const QString ReqPendingFromMeToOthers()const;
    const QByteArray passwd()const;
    virtual QString toString()const;
    const QString toString2()const;//used with sign in request to send contact data.

public slots:
    void setIPAddres(const QHostAddress ip);
    void setPort(const int port);
    bool setPublicKey(const CELIB::RSApublicKey* pubKey);
    void removePublicKey(); //Only used when encryption failed!.
    void setStatus(const STATUS status);
    void setPasswd(QByteArray &passwd);
    void addReqPendingFromMeToOthers(const long cid);
    void removeReqPendingFromMeToOthers(const long cid);
    void addReqPendingFromOthersToMe(const long cid);
    void removeReqPendingFromOthersToMe(const long cid);
    void addContact(const long cid);
    void removeContact(const long cid);
    void setContacts(const QString contacts);
    void setReqPendingFromMeToOthers(const QString reqp2othrs);
    void setReqPendingFromOthersToMe(const QString reqp2me);

private:
    GENDER toGENDER(const QString gender);

    long m_cid;
    QString m_name;
    QByteArray m_passwd;
    QString m_email;
    GENDER m_gender;
    QString m_location;
    STATUS m_status;
    QHostAddress m_ip;
    int m_port;
    QStringList m_contacts;//contains contacts CIDs.
    CELIB::RSApublicKey* m_pubKey;
    AReqPending m_reqPendingFromMeToOthers;// Holding CIDs
    AReqPending m_reqPendingFromOthersToMe;// Holding CIDs
```

4. ملف server.cpp:

```
//server.cpp:
#include <QtNetwork>

#include "connection.h"
#include "server.h"
namespace ConnectionMGR
{
Server::Server(QObject *parent)
    : QTcpServer(parent)
{
    listen(QHostAddress::Any, 27755);
}
QString Server::ipAddress() const
{
    QString ip;
    QList<QHostAddress> ipAddressesList = QNetworkInterface::allAddresses();
    // use the first non-localhost IPv4 address
    for (int i = 0; i < ipAddressesList.size(); ++i) {
       if (ipAddressesList.at(i) != QHostAddress::LocalHost &&
            ipAddressesList.at(i).toIPv4Address()) {
            ip = ipAddressesList.at(i).toString();
            break;
        }
    }
    // if we did not find one, use IPv4 localhost
    if (ip.isEmpty())
        ip = QHostAddress(QHostAddress::LocalHost).toString();
    return ip;
}
void Server::incomingConnection(int socketDescriptor)
{
    emit newConnection(socketDescriptor);
}
}
```

III. نسخة العميل:

5. ملف: contactlist.h :

```
#ifndef CONTACTLIST_H
#define CONTACTLIST_H

#include <QLinkedList>
#include "contact.h"
namespace ContactMGR {
class ContactList : public QLinkedList<Contact*>
{
public:
    explicit ContactList();
    virtual ~ContactList();
    bool containsEmail(const QString& email)const;
    bool containsName(const QString& name)const;
    bool containsContactPtr(Contact* const contPtr);
    QString contactName(const QString& email)const;
    QString contactEmail(const QString& name)const;
    Contact* contactPtr(const QString& name)const;
    void append(Contact* const contPtr);

public slots:

private:
    ContactList(const ContactList&);
    ContactList& operator=(const ContactList&);
};
}
#endif // CONTACTLIST_H
```

6. ملف: main.cpp:

```
#include <QApplication>
#include <QTranslator>
#include <QMessageBox>
#include <QFontDatabase>
#include "mainwindow.h"
#include "dboperation.h"
#include "dardashat.h"

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    QTextCodec::setCodecForTr(QTextCodec::codecForName("utf8"));
    QFontDatabase::addApplicationFont("PTBLDHAD.TTF");
    QFontDatabase::addApplicationFont("TRADBDO.TTF");
    QFontDatabase::addApplicationFont("TRADO.TTF");
    app.setApplicationName("ZAJIL");
    app.setApplicationVersion("1.0.1-BETA");
    app.setOrganizationDomain("sudasoft.com");
    app.setOrganizationName("SUDASOFT SOLUTIONS LTD.");
    app.setLayoutDirection(Qt::RightToLeft);
    QTranslator translator;
    translator.load("drdshTr_ar");
    app.installTranslator(&translator);
    QFile dbaseFile(QString("dBase.db3"));
    if(!dbaseFile.exists())
    {
        QMessageBox msg;
        msg.setIcon(QMessageBox::Critical);
        msg.setText(QMessageBox::tr("Fatal Error:"));
        msg.setInformativeText(QMessageBox::tr("Database file Does Not exist, Unable to run."));
        msg.setDetailedText(QMessageBox::tr("Can Not find database file ["+dbaseFile.fileName()
                                            .toUtf8()+"], please be sure that file is exist "
                                            "then run the application."));
        msg.setDefaultButton(QMessageBox::Ok);
        msg.exec();
        exit(EXIT_FAILURE);
    }
    dBaseMGR::dBconnect dbc(QString("QSQLITE"), dbaseFile.fileName());
    dBaseMGR::dBOperation dbo(dbc, QString(""),
                              QString(""), QString("localhost"));
    Dardashat drdsht(dbo);
    MainWindow mw(drdsht);
    mw.init();// show splash ...
    if(mw.signedIn())//if user signed in successfuly, show main window.
    {
        mw.setContactsLists();
        mw.show();
    }
    return app.exec();
}
```