دراسة استخدام البروتوكول DTLS مع البروتوكول CoAP في النقل الآمن للصور في نظام إنترنت الأشياء المعتمد على السحابة

د. رضوان دنده *

د. أحمد محمود أحمد **

رشا غدير ***

(تاريخ الإيداع 29 / 3 / 2021. قُبِل للنشر في 4/ 8 / 2021)

□ ملخّص □

تشكل إنترنت الأشياء (IoT) ثورة حقيقية في عالم الإنترنت الآن، لاسيما أنها تأخذ بعداً جديداً يعتمد على السحابة. تعتبر بروتوكولات نقل البيانات عاملاً أساسياً للتواصل مابين هذه الأشياء والسحابة لاسيما في سيناريوهات الزمن الحقيقي. يعتبر بروتوكول التطبيقات المقيدة CoAP من البروتوكولات المهمة التي تستخدم لنقل البيانات مابين الأشياء والسحابة، لكنه يعاني من عدم توفيره لقناة آمنة خلال عملية النقل. يعتبر موضوع الحماية من أهم المواضيع التي يجب أخذها بعين الاعتبار خصوصاً في بعض التطبيقات التي تكون فيها البيانات المرسلة من الأشياء على اختلاف أنواعها حساسة وتتطلب الحماية، كمثال على تلك التطبيقات، تلك التي تتعامل مع التعرف على الوجوه وتعقبها.

نقدم في هذا البحث مقترحاً لحماية البروتوكول CoAP باستخدام بروتوكول أمن رزم بيانات طبقة النقل DTLS في نظام IoT مصمم لتمييز الوجوه مبني باستخدام المكتبة OpenCV المقدمة من قبل السحابة. لتحقيق هدفنا، تم في البداية تحليل البنية المقترحة من الناحية الأمنية، ليتم بعدها دراسة عبء استخدام البروتوكول DTLS، حيث تم تحقيق ذلك من خلال المقارنة بين نقل الصور بوجود حماية وبعدمها، ليتبين لنا أن استخدام بروتوكول DTLS يفرض عبء على الأشياء، حيث يزيد استهلاك البطارية بنسبة 12% كما أنه يزيد زمن استخدام المعالج بمقدار الضعف تقريباً.

الكلمات المفتاحية: إنترنت الأشياء، الحوسبة السحابية، بروتوكول التطبيقات المقيدة، حماية، أمن رزم بيانات طبقة النقل، تمييز الوجه، OpenCV.

* أستاذ، قسم النظم والشبكات الحاسوبية، كلية الهندسة المعلوماتية، جامعة تشرين، اللاذقية، سورية.
E-mail: radwan.dandah@tishreen.edu.sy

** مدرس، قسم النظم والشبكات الحاسوبية، كلية الهندسة المعلوماتية، جامعة تشرين، اللاذقية، سورية.
E-mail: ahmad.m.ahmad@tishreen.edu.sy

*** طالبة دراسات عليا (دكتوراه)، قسم النظم والشبكات الحاسوبية، كلية الهندسة المعلوماتية، جامعة تشرين، اللاذقية، سورية.
E-mail: r.ghadeer@tishreen.edu.sy

A Study of Using DTLS and CoAP protocols in the secure transfer of images in a cloud-based IoT system

Dr. Radwan Dandah [*]
Dr. Ahmad Mahmoud Ahmad [**]
Rasha Ghadeer [***]

(Received 29 / 3 / 2021. Accepted 4 / 8 / 2021)

□ ABSTRACT □

Nowadays, The Internet of Things (IoT) is considered to be a real revolution in the world of the Internet, especially with the appearance of cloud technology. Data transfer protocols are considered a key factor for communication between these things and the cloud, especially in real-time scenarios. Constrained Application Protocol (CoAP) is an important protocol used to transfer data between things and the cloud, but it suffers from not achieving any secure channel during the transmission process. Security is considered as one of the most important topics that must be taken into consideration, especially in some applications where the data sent from objects of different types are sensitive and need to be protected, as an example of those applications, those that deal with face recognition and tracking.
This paper provides a proposal for securing CoAP protocol using the Datagram Transport Layer Security (DTLS) protocol in an IoT system designed to recognize faces based on the OpenCV library provided by the cloud. To achieve our goal, the proposed architecture was initially analyzed from the security point of view, after that we study the burden of using the DTLS protocol, as this was achieved by comparing the transmission of images with the presence and absence of protection, to show that the use of the DTLS protocol imposes a burden on things, as the power consumption increases by 12% and the processor usage time increases by almost twice.

Keywords: Internet of Things, Cloud Computing, Constrained Application Protocol (CoAP), Security, Datagram Transport Layer Security (DTLS), Face Recognizer, OpenCV.

[*] **Professor, Department of Computer Systems and Networks, Faculty of Informatics Engineering, Tishreen University, Lattakia, Syria. E-mail: radwan.dandah@tishreen.edu.sy**
[**] Assistant Professor, Department of Computer Systems and Networks, Faculty of Informatics Engineering, Tishreen University, Lattakia, Syria. E-mail: ahmad.m.ahmad@tishreen.edu.sy
[***] PhD Student, Department of Computer Systems and Networks, Faculty of Informatics Engineering, Tishreen University, Lattakia, Syria. E-mail: r.ghadeer@tishreen.edu.sy

مقدمة:

شهد الانترنت على مدى العقود الماضية نمواً واسعاً وسريعاً، بدءاً من خدمة مليارات الأشخاص، وصولاً إلى ربط مليارات من الحواسيب وأجهزة الموبايل مع بعضها البعض. جاءت الـ IoT لتكون الخطوة التالية من هذا التطور، حيث أنها امتداد جذري من الانترنت الحالي، والذي يسمح بربط الأشياء مع بعضها البعض، مما يسمح بانتقالها من التقليدية إلى الذكية، وهذا أحدث ثورة في فائدتها و تطبيقاتها. إن من أهم الأمثلة على IoT هي: المدن الذكية، المنازل الذكية، والسيارات الذكية .. وهذه التطبيقات ليست سوى عدد قليل من التطبيقات الممكنة لتقنية انترنت الأشياء.

إن إنترنت الأشياء أو اختصاراً IoT عبارة عن مجموعة من الأشياء التي تتواصل مع بعضها البعض لتقديم خدمات ذكية للإنسان. أما الشيء (الغرض) في إنترنت الأشياء فهو أي شيء يمكنه الاتصال بالانترنت. استهدفت العديد من الجهود البحثية تحديات البيئات المقيدة من حيث قدرات المعالجة والتخزين واستهلاك البطارية مع موارد محدودة، حيث أن عقد IoT تقوم بتبادل رسائل تحكم ورسائل بيانات بين الأشياء في الزمن الحقيقي. إن امكانيات أجهزة IoT محدودة جداً وبالتالي هذه الأجهزة أو الحساسات غير قادرة على معالجة البيانات[1]، وبالتالي نحن بحاجة إلى طرف آخر يأخذ على عاتقه عبء معالجة البيانات، والحل الأمثل هو باستخدام الخدمات المقدمة من قبل السحابة. تقوم السحابة بالاستفادة من إنترنت الأشياء عن طريق توسيع مجال تعاملها إلى أشياء حقيقية موزعة بطريقة ديناميكية، بالإضافة إلى مشاركتها في تقديم الخدمة في عدد كبير من سيناريوهات الزمن الحقيقي، من هنا كان السيناريو النموذجي هو استخدام خدمات السحابة لإدارة هذا العدد الكبير من الأجهزة، ومعالجة البيانات الخاصة بهم، وتنسيق النتائج [2].

هناك العديد من التحديات التي تواجه إنترنت الأشياء والتي يشكل الأمن العنصر الرئيسي فيها، بالإضافة إلى القيود المتعلقة بالموارد. عند اتصال الشيء إلى الإنترنت فإنه يصبح أكثر عرضة لخطر التهديدات والهجمات الخبيثة، حيث يمكن للأشخاص أو الأجهزة الأخرى نشر محتويات ضارة على هذه الشبكات، وسرقة البيانات من هذه الأجهزة، واستخدامها بشكل غير قانوني. لذا فإن تحقيق الجوانب الأمنية بالتوازي مع الحلول الشبكية هو أمر ضروري لتواصل أكثر أمناً. أما فيما يتعلق بالموارد، فإن هذه الأجهزة تحتوي على ذاكرة قليلة، وقدرات معالجة محدودة، وعرض نطاق اتصالات قليل، ومعظمها مدعوم بالبطاريات، أي أنها تعاني من محدودية في قدرات الإرسال وقيود فيما يتعلق باستهلاك الطاقة في معظم مكوناتها، والذي يؤدي إلى التنفيذ المحدود للمخططات الأمنية المعقدة. من هنا برز تحدي مهم و هو تطبيق تقنيات حماية على البروتوكولات التي تستخدم في عملية التواصل ما بين الأشياء في IoT والحوسبة السحابية. لعل من أشهر بروتوكولات نقل البيانات هو بروتوكول التطبيقات المقيدة CoAP [3] . قام الباحثون بنشر ورقة بحثية [4]، تعتمد فكرتها على تطوير بروتوكول التطبيقات المقيدة CoAP ليناسب المعمارية الجديدة لوجود السحابة في بنية إنترنت الأشياء. قدم البحث Californium، والذي هو عبارة عن تحقيق لبروتوكول CoAP بلغة الجافا، مخصص لبيئات إنترنت الأشياء مع الحوسبة السحابية. تم تقييم أداء هذا البروتوكول باستخدام الأداة CoAPBench بإرسال رسائل نصية للمخدم. البروتوكول المحسن أظهر 64% تحسين بالإنتاجية (Throughput) عن مخدمات HTTP، وهذا يؤكد أهمية استخدام بروتوكول CoAP في الأنظمة التي تتعامل بها إنترنت الأشياء مع الحوسبة السحابية و خاصة في تطبيقات الزمن الحقيقي[4] . لكن بروتوكول CoAP لا يزود بأي خدمات أمنية، تتمثل إحدى الطرق الممكنة لتوفير الأمان لهذا البروتوكول في استخدام بروتوكول أمن رزم بيانات طبقة النقل DTLS الذي يقدم وظيفة المصادقة بالإضافة إلى حماية المعلومات الحساسة خلال عملية النقل. قام الباحث في [5] بالمقارنة بين بروتوكول التطبيقات المقيدة CoAP من دون حماية، وبين استخدام بروتوكول DTLS مع بروتوكول التطبيقات

المقيدة كنوع من الحماية، تم استخدام المحاكي Cooja على نظام التشغيل Contiki-OS. في حالة CoAP من دون DTLS، تم استخدام الحساس Wismote wireless sensor مع وجود زبون CoAP والذي هو Copper (آلية يتم اضافتها إلى أي متصفح ليتمكن من إرسال و استقبال رسائل بروتوكول CoAP) ومخدم CoAP الذي يقوم بإستقبال الطلب وارسال استجابة. أما في حال CoAP مع استخدام بروتوكول DTLS لتأمين الحماية، تم استخدام عقدتين في محاكي Cooja واحدة كزبون والثانية كمخدم مع استخدام مكتبات TinyDTLS و CoAP مفتوحة المصدر.
البيانات المرسلة بين الزبون والمخدم من نوع نص، تمت المقارنة وفقاً لمعياري استهلاك طاقة البطارية بالإضافة إلى معيار استخدام الذاكرة، أظهرت نتائج المحاكاة زيادة في استهلاك الذاكرة عند استخدام DTLS كطريقة لتأمين الحماية، حيث ازداد استخدام الذاكرة RAM بنسبة 30%، كما ازداد استخدام الذاكرة بنسبة 59%، كما أنها أظهرت أن معدل زيادة استهلاك الطاقة بعد إضافة الحماية قد ازدادت بحوالي 10%.
قمنا في هذا البحث ببناء سيناريو كامل لمجموعة من عقد إنترنت الأشياء (كاميرات) تقوم بإرسال بياناتها (صور) باستخدام بروتوكول CoAP، ويتم تجميع هذه البيانات في سحابة والاستفادة من خدماتها في معالجة البيانات المجمعة، حيث أن الخدمات المقدمة هي خدمتي تحديد الوجوه وتمييزها. تم تدريب السحابة باستخدام مجموعة من التوابع المضمنة داخل مكتبة OpenCV على مجموعة من الصور التي تحتوي على عدة وضعيات لأشخاص محددين مطلوبين، لينتج لدينا مودل محددة نستخدمها لحساب قيمة تدل على مدى تطابق صور الوجوه المارة مع الوجوه المدرب عليها السحابة. عند ورود أي صورة جديدة من الكاميرات، يتم تحديد اذا كانت هذه الصورة تحتوي على وجه أم لا بواسطة خدمة تحديد الوجوه، اذا احتوت على وجه يتم حساب قيمة لهذا الوجه بالاعتماد على المودل السابقة، اذا كانت هذه القيمة كبيرة فهذا يعني أن الوجه قد تطابق إلى حد كبير مع أحد الوجوه المدرب عليها السحابة مسبقاً، عندئذ تقوم السحابة بإرسال إشعار إلى المنبه الذي يقوم بدوره بإصدار صوت معين. قمنا بتطبيق نوع من الحماية على بروتوكول CoAP باستخدام بروتوكول DTLS على نوع بيانات صور، حيث يتم تشفير جميع الصور المرسلة قبل ارسالها، لتقوم السحابة بفك التشفير و استخدام الصور في عملية تمييز الوجوه. قمنا بمقارنة أداء بروتوكول CoAP من دون حماية مع أدائه عند إضافة بروتوكول DTLS كنوع من الحماية وذلك من الناحية الأمنية، بالإضافة إلى إجراء نفس المقارنة من ناحية العبء على الأشياء آخذين بعين الاعتبار البارامترين التاليين: (استهلاك طاقة البطارية ، زمن استخدام المعالج).

أهمية البحث وأهدافه:

يزداد يوماً بعد يوم عدد الأشياء المتصلة بالانترنت وتزداد معها أهمية التطبيقات المقدمة من قبل إنترنت الأشياء مترافقة مع حساسية البيانات المرسلة من قبل الأشياء إلى السحابة، من هنا فإننا بحاجة إلى تطبيق آليات حماية معينة على البيانات المرسلة لضمان سريتها وسلامة وصولها إلى السحابة. تأتي أهمية هذا البحث في تقديم حل أمني لتراسل البيانات باستخدام بروتوكول CoAP يعتمد على بروتوكول DTLS، خاصة أن بروتوكول CoAP يعتمد على بروتوكول UDP الذي لا يوفر أي نوع من أنواع الحماية. من هنا يهدف هذا البحث إلى تطبيق الحماية باستخدام البروتوكول السابق وتقييم التأثير الذي يفرضه تطبيق مثل هذا البروتوكول، وخاصة أن البنية المستخدمة في هذا البحث

تستخدم الصور كنوع حمولة مرسل من قبل الأشياء إلى السحابة، على عكس الدراسات السابقة التي تستخدم النص كنوع حمولة.

طرائق البحث ومواده:

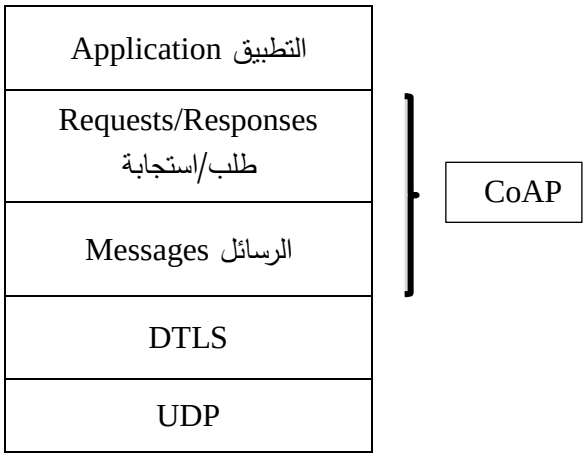
يعتمد البحث الطرق الوصفية والتجريبية والتحليلية، حيث يبدأ البحث بعرض لبروتوكول CoAP يليه شرح لبروتوكول DTLS مع توضيح بنية عمله، ومن ثم الشرح للعديد من البنى التي تم استخدامها في البنية العملية كإطاري العمل Californium و Scandium، بالإضافة إلى شرح للخدمة المقدمة من قبل شركة Google (Firebase Cloud Messaging)، ليتم بعدها شرح المكتبة البرمجية المفتوحة المصدر للرؤية الحاسوبية Open Source Computer Vision Library (OpenCV)، ومن ثم يتم شرح البنية العملية المنجزة. نتبع بعد ذلك الدراسة التحليلية لتقييم البنية المقترحة أمنياً وننهي البحث بإجراء تجارب لتقييم أثر استخدام البروتوكول DTLS مع البروتوكول CoAP من ناحية العبء على الأشياء.

1- بروتوكول التطبيقات المقيدة (Constrained Application Protocol):

بروتوكول CoAP عبارة عن بروتوكول ويب تم تصميمه لأنظمة الزمن الحقيقي المبنية على استخدام أجهزة IoT صغيرة الحجم، ذات التكلفة قليلة والمحدودة الموارد. يمكن لهذا البروتوكول من ارسال بيانات الأشياء بالإضافة إلى رسائل التحكم، كما يؤمن عمليات تبادل للرسائل غير المتزامنة باستخدام بروتوكول رزم بيانات المستخدم UDP، يعتمد CoAP على نموذج مخدم/زبون مثل بروتوكول HTTP، ويمثل عملية الارسال والاستقبال بنفس الطريقة. يتم تعريف مجموعة من الموارد عن طريق ربطها بمعرف يسمى معرف الموارد المحدد Uniform Resource Identifiers (URIs) باستخدام طرائق النقل مثل GET، POST ، PUT، DELETE أي (إحضار – إرسال – تعديل – حذف) مورد من مخدم، كما أنه يتم الرد بإحدى أرقام الاستجابة والذي يعبر عن الاستجابة الصحيحة أو الخاطئة. على عكس بروتوكول HTTP، يقوم بروتوكول CoAP بتبادل الرسائل بطريقة غير مباشرة عبر UDP، لذلك ينبغي عليه تحقيق كل الخدمات المقدمة من قبل TCP، لاسيما الأمنية منها [6]. يوجد عدة تحقيقات لهذا البروتوكول منها Californium و nCoAP بلغة الجافا ، libcoap بلغة C ، CoAP.NET بلغة #C، وأيضاً aiocoap ، CoAPthon بلغة Python وغيرها الكثير [7]. تم استخدام Californium في البنية العملية المنجزة في هذا البحث وتم اختياره لأن اللغة المستخدمة في التطبيق العملي هي الجافا كما أنه مخصص لبيئات إنترنت الأشياء مع الحوسبة السحابية، حيث أن Californium تتميز ببنية قابلة للتطوير، وتتفوق على مخدمات HTTP عالية الأداء[4]، وبالتالي هو الأنسب للاستخدام [8]، بالإضافة إلى وجود Scandium وهو تطبيق مفتوح المصدر لبروتوكول DTLS على بروتوكول CoAP [9].

2- بروتوكول أمن رزم بيانات طبقة النقل (Datagram Transport Layer Security DTLS) :

تم اقتراح بروتوكول DTLS 1.2 من قبل IETF بالمستند رقم (RFC 6347)، وهو يستخدم لتأمين مسار الشبكة، ويشبه DTLS بروتوكول أمن طبقة النقل (Transport Layer Security) TLS مع تعديلات معينة لنستطيع استخدامها في البيئات التي تتعامل مع بروتوكولات نقل غير موثوقة مثل UDP، من هذه التعديلات: معالجة فقدان الرسالة، إعادة ترتيب الرسالة، تعديل حجم الرسالة [10]. يمكن استخدام بروتوكول DTLS لتأمين بروتوكول CoAP، حيث يضمن تحقيق الأمن الشامل للتطبيقات المختلفة عن طريق العمل بين طبقة التطبيقات وطبقة النقل كما في الشكل (1).

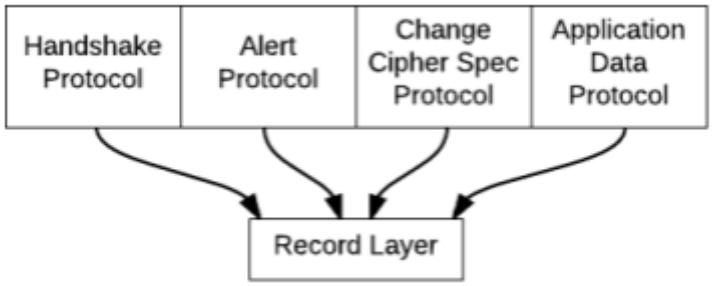


الشكل (1) : طبقات بروتوكول CoAP المحمي ببروتوكول DTLS

يتألف البروتوكول DTLS من طبقتين موضحتين في الشكل (2) وهما[10] :

الطبقة الأولى وتدعى طبقة المصافحة وتتألف من أربع بروتوكولات فرعية: المصافحة Handshake ، تغيير مواصفات التشفير change cipher spec ، التنبيه Alert ، بروتوكولات التطبيق Application.

الطبقة الثانية الأعلى و تدعى طبقة التسجيل حيث تتفاعل مع الطبقة الدنيا وتحتوي على بروتوكول التسجيل record فقط.



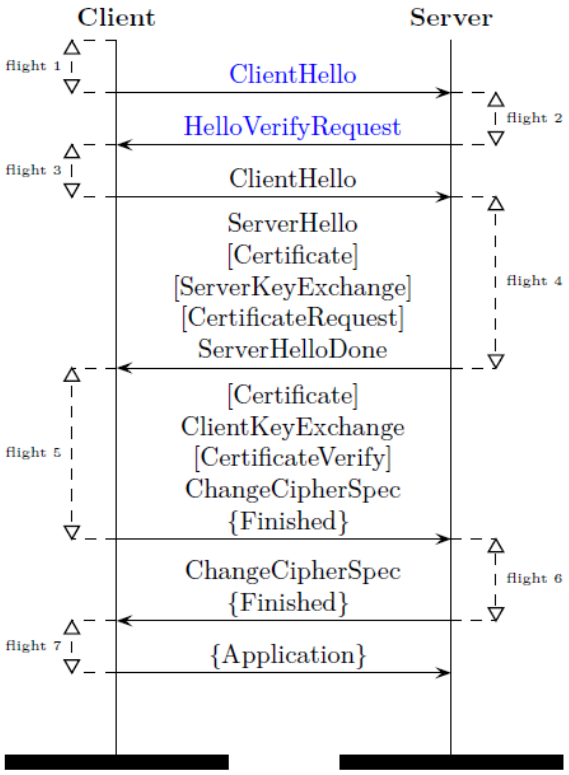
الشكل (2) : البروتوكولات الجزئية لبروتوكول DTLS

نعرض فيما يلي البروتوكولات المكونة للطبقتين الأولى والثانية.

2-1- بروتوكول المصافحة:

وهو عبارة عن سلسلة من الرسائل المتبادلة بين الزبون والمخدم قبل أن يتم نقل أي بيانات. هذا البروتوكول مسؤول عن المصادقة والتفاوض بشأن متحولات الأمان لإنشاء جلسات آمنة قابلة للاستئناف. يتعين على الطرفين المتصلين التأكد من أنهما يدعمان نفس الخوارزميات ويتفقان على استخدامها. كما يحتاج كلاهما أيضاً إلى إنشاء مفاتيح مطابقة للاستخدام مع الخوارزميات المحددة. تسمى هذه العملية بالتفاوض على الجلسة، وبعدها ينتج لدينا جلسة خاصة آمنة بين الطرفين المتصلين. الجدير بالذكر أنه يجب ارسال رسائل المصافحة بترتيب محدد في بروتوكول TLS، وهذا لايتوافق مع ضياع الرسالة وإعادة ترتيبها في حالة UDP. بينما يمكن تطبيق البروتوكول DTLS عبر بروتوكولات طبقة النقل غير الموثوق بها، وبالتالي يجب أن يوفر بروتوكول DTLS موثوقية لرسائل المصافحة، وهذا يحدث عن

طريق إضافة آليتين: مؤقت لإعادة الارسال للتعامل مع فقدان الرسالة، ورقم تسلسلي للرسالة ضمن عملية المصافحة. يسمح رقم التسلسل المحدد هذا للمستقبل بتحديد الترتيب الصحيح لرسائل المصافحة بسرعة. يتم تجميع رسائل DTLS في مجموعات أو رحلات flights، حيث يمكن أن تحتوي كل رحلة على عدد من الرسائل. يوضح الشكل (3) عملية مصافحة DTLS [11] .



الشكل (3) : عملية المصافحة لبروتوكول DTLS، الرسائل المشفرة داخل أقواس كبيرة، والرسائل الاختيارية بداخل أقواس مربعة .

تبدأ عملية المصافحة في الرحلة 1 بإرسال الزبون رسالة ClientHello إلى المخدم، تحتوي هذه الرسالة على مصفوفة بايتات عشوائية (رقم عشوائي)، تستخدم لاحقاً في مرحلة إنشاء المفتاح، وقائمة بخوارزميات التشفير المناسبة وأساليب الضغط المدعومة. كما تحتوي على أعلى رقم لنسخة بروتوكول DTLS التي يدعمها الزبون. إذا أراد جانب المخدم إعادة التفاوض على إعدادات أو بارمترات الاتصال، فإنه يقوم في الرحلة 2 بإرسال رسالة HelloVerifyRequest، تحتوي هذه الرسالة على ملف تعريف ارتباط عديم الحالة (cookie)، والتي يجب على الزبون تضمينه كلاحقة في رسالة ClientHello الثانية. يقوم المخدم بعد ذلك بالتحقق من هذا الملف والتأكد هل هو صحيح ليستمر بعملية المصافحة. تجبر هذه العملية المهاجم/الزبون أن يكون قادر على استلام الملف، والتي تجعل من بعض هجمات الحرمان من الخدمة أمراً صعباً[11]. بعد التأكد من ذلك يقوم المخدم في الرحلة 4 بإرسال رسالة ServerHello تحتوي على مجموعة من البايتات العشوائية (رقم عشوائي آخر)، تستخدم لاحقاً في مرحلة إنشاء المفتاح، كما تحتوي على رقم نسخة بروتوكول DTLS التي يدعمها المخدم، ومجموعة التشفير المختارة من قبل المخدم. تحمل شهادة المخدم في الشهادة Certificate، والتي تحتوي على مفتاحه العام. يرسل المخدم في رسالة

ServerKeyExchange مفتاحاً عاماً سريع الزوال يتم توقيعه باستخدام المفتاح الخاص لشهادة المخدم. هذا التوقيع يغطي أيضاً كلا الرقمين العشوائيين (الرقم المرسل من الزبون والرقم الآخر من المخدم). يطلب في الرسالة CertificateRequest من الزبون المصادقة على المخدم. هذه الرسالة اختيارية، ولاتستخدم إلا عندما يتم إعداد المخدم ليقوم بمصادقة الزبائن عن طريق الشهادات. أما رسالة ServerHelloDone تشير إلى أنه لا توجد رسائل أخرى واردة.

في الرحلة 5 يقوم الزبون بالاستجابة بقائمة من الرسائل، رسالتي Certificate و CertificateVerify رسالتان اختياريتان، ولايتم ارسالهما إلى المخدم إلا عندما يطلب المخدم المصادقة من الزبون. وهي تحتوي على التوالي شهادة الزبون والتوقيع المحسوب على جميع الرسائل السابقة باستخدام المفتاح الخاص طويل المدى للزبون، هذه الرسالة توضح أن المفتاح الخاص بالمخدم قد طابق المفتاح العام للزبون. كما يرسل الزبون في رسالة ClientKeyExchange مشاركته في المفتاح العام. ثم يستخدم كلا الطرفين المعلومات المتبادلة لاشتقاق مفاتيح متماثلة يتم استخدامها في بقية البروتوكول. يرسل الزبون رسالة ChangeCipherSpec للإشارة إلى أنه سيستخدم المفاتيح التي تم التفاوض عليها من الآن في طبقة التسجيل، كما أنها تظهر أن الزبون قام بتشفير جميع الرسائل بنجاح باستخدام المفاتيح مع خوارزمية التشفير. وأخيراً يرسل رسالة Finished تحتوي على النسخة المشفرة بالمفاتيح الجديدة لجميع رسائل المصافحة السابقة. بعد ذلك في مجموعة الرحلة 6 يستجيب المخدم برسائل ChangeCipherSpec و Finished الخاصة به. والتي تنتهي بإرسالها عملية المصافحة. بعد ذلك، يمكن لكل من الزبون والمخدم تبادل بيانات التطبيق الموثوقة والمشفرة فيما بينهما[10]،[11] .

عملية المصافحة في بروتوكول DTLS مهمة جداً لبدأ اتصال آمن بين الزبون والمخدم. اذا فشلت العملية، لايمكن للزبون والمخدم التواصل مع بعضهم البعض.

2-2- بروتوكول تغيير مواصفات التشفير: يتم استخدام هذا البروتوكول في الطبقة العليا من DTLS أثناء عملية المصافحة للإشارة إلى ضرورة حماية جميع الرسائل اللاحقة باستخدام مواصفات التشفير والمفاتيح التي تم التفاوض عليها أثناء عملية المصافحة.

2-3- بروتوكول التنبيه: يستخدم هذا البروتوكول لنقل معلومات حول رسائل الأخطاء أو التحذيرات بين النظيرين (مخدم وزبون).

2-4- بروتوكول التطبيق: يشير إلى البروتوكول الموجود في طبقة التطبيق على سبيل المثال CoAP.

2-5- بروتوكول تسجيل DTLS: يستخدم للتعامل مع نقل البيانات و تطبيق آليات الأمان. يحمي بيانات التطبيق باستخدام معلومات الأمان والمفاتيح المولدة أثناء عملية المصافحة. يتم تجزئة الرسائل الصادرة وضغطها وتشفيرها في طبقة التسجيل هذه والعكس صحيح بالنسبة للرسائل الواردة [10].

3- البنية Californium :

هو تحقيق مفتوح المصدر لبروتوكول CoAP موجود على GitHub [8]، مكتوب بلغة الجافا، يوفر خدمات خلفية للتواصل مع جميع أنواع أجهزة IoT سواء كانت محدودة أو قوية الموارد، كما يوفر واجهة برمجة تطبيقات ملائمة لخدمات الويب التي تدعم جميع ميزات بروتوكول CoAP [12].

يستخدم تحقيق Californium لبناء تطبيقات انترنت الأشياء الخاصة. تحتوي هذه المكتبة على خمس حزم فرعية و هي: californium-core، californium-osgi، californium-proxy، element-connector ، scandium-core، حيث يحتوي Californium-core على تحقيق بلغة الجافا للوظائف الأساسية لبروتوكول CoAP. ويتم استخدامه للتهيئة عند إنشاء مخدم أو زبون [13].

نعرض فيما يلي المكونين الأساسيين لحزمة Californium-core المستخدمين في الجزء العملي:

1- مخدم CoAP: هو عبارة عن صف CoapServer موجود ضمن حزمة Californium-core مهمته تحقيق كل المهام المطلوبة من مخدم CoAP للاستفادة منها في بناء تطبيقات تعتمد على بروتوكول CoAP بلغة الجافا. يحتوي هذا الصف على ضمانات للشبكة، معلومات عن منفذ الاستماع ومعلومات نقطة النهاية Endpoints، حيث يمكنه الاستماع إلى أكثر من منفذ من نقاط النهاية هذه. منفذ CoAP الافتراضي هو 5683 [8]،[13].

2- زبون CoAP: هو عبارة عن صف CoapClient يحتوي على جميع التحقيقات التي تخص الزبون في حزمة californium-core، على عكس مخدم CoAP، لايحتوي هذا الصف على برنامج لتوصيل الرسائل، بل انه يحتوي على قيمة مؤقت للتحكم بحالة الرسائل المرسلة والمستقبلة. يستطيع الزبون ارسال طلب إلى مورد واحد فقط موجود على المخدم وبالتالي الزبون لايدعم حالة الارسال المتعدد، من الممكن أن يكون هذا الطلب (إحضار – إرسال – تعديل – حذف) لمورد موجود في مخدم CoAP [8]،[13] .

4- البنية Scandium (Sc):

هو تحقيق لبروتوكول DTLS 1.2 مفتوح المصدر مكتوب بلغة الجافا لتأمين حماية لبيئة Californium التي تستخدم بروتوكول CoAP. تطبق Scandium واجهة element-connector التي توفر واجهة برمجة تطبيقات لإرسال واستقبال أجزاء البيانات الأولية (البايتات). كما يمكن استخدام Scandium كمكتبة توفر طبقة نقل آمنة قائمة على UDP لأي نوع من التطبيقات الموجودة فوقها. كما أنها مفتوحة المصدر ومتاحة داخل مستودع Eclipse على Github [9]،[12].

عندما نضيف بروتوكول DTLS إلى مخدم CoAP أو زبون CoAP يجب علينا أن نعرف مجموعة من إعدادات الحماية على هذه العقد. يوجد أربع إعدادات مهمة لتنفيذ وتحقيق بروتوكول DTLS فوق بروتوكول CoAP وهي:

1- مخزن Psk (Psk Store): يعين مفتاح مشترك يستخدم في عملية المصادقة، يحتوي على هوية وعلى معلومات عن المفتاح المستخدم في عملية التشفير، تخزن هذه المعلومات بشكل تنسيق نصي، وهذه المعلومات هي نفسها لكل من المخدم والزبون أثناء عملية الاتصال ببعضهما البعض.

2- مخزن الثقة (Trust Store): يتضمن مخزن الثقة شهادات الجذر بصيغة X.509 (معيار لشهادات المفتاح العام)، يحتوي على مفتاح عام و هوية.

3- الهوية (Identity): توفر الهوية مفتاحاً خاصاً وتعريفاً للشهادة المستخدمة مع الموصل Connector. تستخدم هذه القيمة لإثبات هوية العقد، المخدم للزبون والزبون للمخدم. لذلك يجب أن تكون القيم متطابقة لضمان هذه العملية على كلتا العقدتين.

4- مجموعات التشفير المدعومة (Supported Cipher Suites): تتضمن قائمة بمجموعات التشفير المدعومة. ليتم حماية الاتصال بين الزبون والمخدم يجب على الزبون والمخدم دعم مجموعة تشفير متماثلة واحدة على الأقل ليتم الاتفاق عليها.

يوجد في نموذج scandium-core صفوف مساعدة لتكوين خصائص الأمان، أحد هذه الصفوف يدعى مخزن المفاتيح KeyStore، والذي يوفر عملية تخزين مفاتيح التشفير والشهادات. يتم استخدام هذه المفاتيح والشهادات المخزنة لتعيين هوية في إعدادات الأمان. كما يستخدم هذا الصف المساعد لتخزين الشهادات وقراءتها من مخزن الثقة، وحفظ معلومات عن مجموعات التشفير المدعومة، أيضاً معلومات عن إعددات أو معرفات الهوية .

بعد القيام بكل الاعدادات السابقة في الكائن المنشئ، يتم اضافة هذا الباني إلى موصل DTLS والذي يدعى DTLConnector، حيث يتم انشاء كائن لاستخدامه في عقدة المخدم وفي عقدة الزبون. في النهاية يكون المخدم والزبون جاهزين للاتصال الآمن بعد الانتهاء من الاعدادات السابقة [13].

5- خدمة Firebase Cloud Messaging (FCM):

وهي عبارة عن خدمة مقدمة من شركة Google مطورة من الخدمة Google Cloud Messaging تتعامل مع إرسال الرسائل وتوجيهها وتخزينها بين تطبيقات المخدم وتطبيقات الزبون المتواجدة على الجهاز المحمول. إن FCM عبارة عن وسيط ما بين مرسلي الرسائل والزبائن. تطبيق الزبون هو تطبيق يعمل على جهاز يدعم FCM ، أما تطبيق المخدم هو عبارة عن مخدم يتواصل مع تطبيق الزبون عن طريق FCM. باستخدام FCM يمكن لتطبيق المخدم أن يقوم بإرسال الرسائل أو الاشعارات إلى جهاز واحد، أو مجموعة من الأجهزة، أو إلى عدد من الأجهزة التي تشترك أو تهتم بموضوع محدد. أما تطبيق الزبون فإنه يقوم باستقبال الرسائل من تطبيق المخدم أي تلقي اشعارات عن بعد [14].

6- المكتبة البرمجية المفتوحة المصدر للرؤية الحاسوبية (OpenCV):

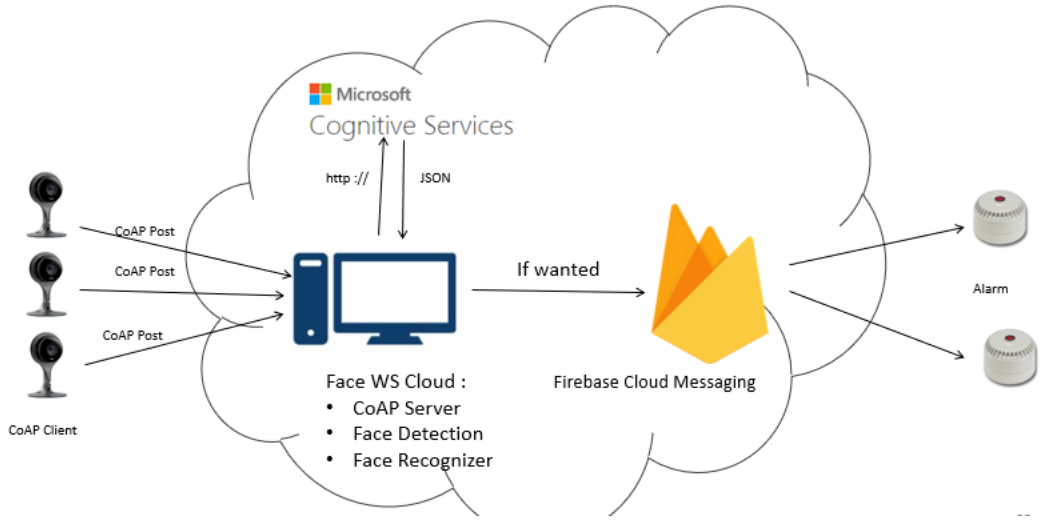
إن كل العمليات على الصور هي مجرد عمليات رياضية بحتة، ولكن ليس من المعقول أن يقوم المبرمجون بعمل كل هذه العمليات في كل مره يستخدمون فيها الصور. ولذلك تم انشاء المكتبة البرمجية المفتوحة للرؤية الحاسوبية (OpenCV) ، والتي تهدف بشكل أساسي لتطوير الرؤية الحاسوبية، بحيث تحتوي على دوال تقوم بعمل معظم عمليات الصور التي يحتاجها المبرمجون كما تحتوي على دوال تقوم بتحديد أجسام معينة في الصورة مثل الخطوط والاشكال الهندسية، ودوال اخرى لتحديد الألوان، كما يمكن استدعاء ملفات XML خاصة بتحديد وجه الانسان وعينيه وشعره ويده وجسمه ككل، ويمكن استخدامها في التعرف على الشخص، كما يمكن عمل مصنف خاص بأي جسم (التقاط صور عديده لنفس الجسم من كل الاتجاهات حتى تتعرف عليه الكاميرا من كل الاتجاهات). طورت من قبل شركة (Intel) ، وهي مجانية من أجل الاستخدام الأكاديمي و التجاري، تملك العديد من التحقيقات بعدة لغات (C++، C، java،Python)، كما أنها تستخدم على جميع الأنظمة الحاسوبية منها (ويندوز ولينوكس و أندرويد)، وهي تركز بشكل أساسي على معالجة الصور في الزمن الحقيقي[15].

7- الجزء العملي:

تم الاستفادة من جزء من البنية العملية لإنترنت الأشياء مع الحوسبة السحابية المنجزة مسبقاً والموجودة في المرجع [16] والموضحة في الشكل (4)، حيث تم اجراء جميع التعديلات بحيث نطبق حماية على بروتوكول CoAP باستخدام بروتوكول DTLS ليتشكل لدينا سيناريو جديد آمن.

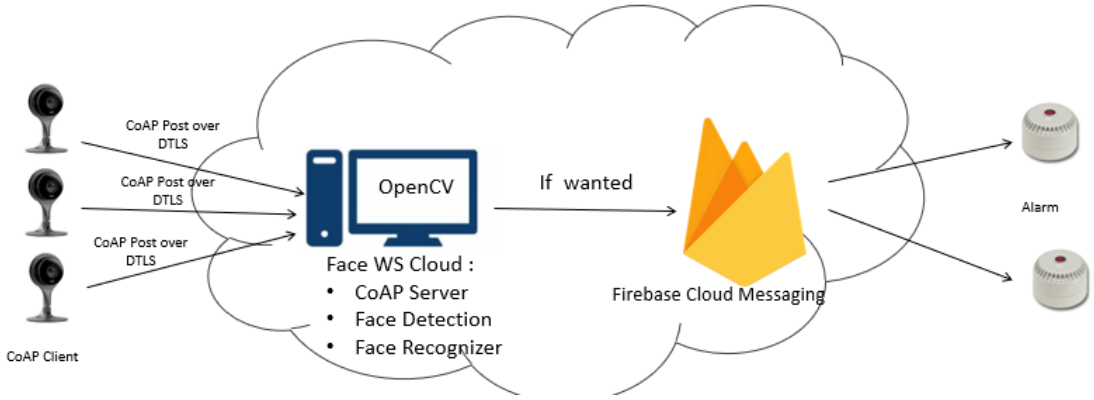
يتم في البنية الموضحة في الشكل (4) التقاط مجموعة من الصور خلال فترات زمنية معينة وارسالها باستخدام بروتوكول CoAP إلى السحابة التي تقوم بدورها بتقديم خدمتين وهما خدمة اكتشاف الوجوه (Face Detection) وخدمة تمييز الوجوه (Face Recognizer) بالاستعانة بالخدمة المقدمة من شركة مايكروسوفت والتي تدعى خدمات

مايكروسوفت المعرفية أو الإدراكية Microsoft Cognitive Services. تقوم السحابة المنجزة والتي تدعى FaceWS بإرسال الصورة باستخدام بروتوكول HTTP (وذلك لأنها لا تدعم بروتوكول CoAP) إلى مايكروسوفت، والتي تقوم بدورها باستقبال الصورة واعادة النتائج بصيغة JSON إلى السحابة، حيث يتم تحديد إذا وجد وجه في الصورة أم لأ، كما يتم مقارنة الوجوه الموجودة بمجموعة من الوجوه المدربة مسبقاً والمحددة لخدمة مايكروسوفت. تقوم السحابة بمقارنة النتائج مع الأشخاص المطلوبة، في حال تحقق أي منها تقوم بإرسال إشعار إلى FCM ليقوم بدوره بإصدار صافرة إنذار للدلالة على وجود شخص ما مطلوب قد تم التقاطه في الكاميرا.



الشكل (4) : بنية IoT Cloud

يوضح الشكل (5) البنية الجديدة والتي توفر نوع من الحماية بالاعتماد على بروتوكول DTLS. تعتمد هذه البنية على البنية السابقة في الأساسيات لكنها تختلف عنها في عملية ارسال واستقبال البيانات. يتم في البنية السابقة ارسال واستقبال البيانات دون إضافة أي نوع من الحماية، أما في هذا السيناريو فيتم الاستعانة ببروتوكول DTLS لتطبيق الحماية على الصور المرسلة من قبل الكاميرات والتي تصل إلى السحابة. كما أنه تم الاستغناء عن خدمات مايكروسوفت المعرفية واستبدالها بسحابة خاصة بنا معتمدة على مكتبة OpenCV تستخدم في عملية المطابقة، حيث يتم معالجة الطلب بشكل محلي من دون ارسال واستقبال الصورة عبر الإنترنت إلى مايكرسوفت، لتكون نتائج المقارنات غير متأثرة بضعف جودة الإنترنت. يتم معالجة النتيجة التي تعيدها OpenCV في السحابة ، التي تقوم بمقارنة النتائج مع المطلوب، في حال تحقق أي منها تقوم بإرسال إشعار إلى FCM ليقوم بدوره بإصدار صافرة إنذار.



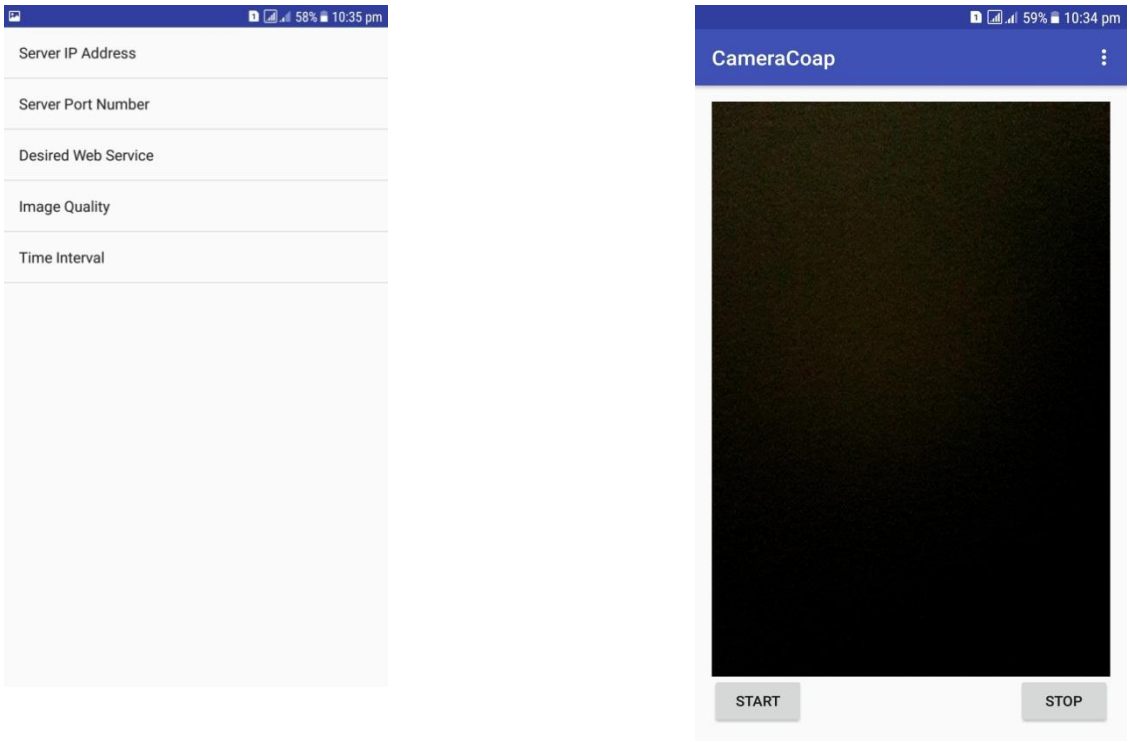
الشكل (5) : بنية IoT Cloud الجديدة

وفيما يلي شرح مفصل للبنية الجديدة ابتداء بالزبائن ثم شرح للسحابة المنجزة وانتهاء بجهاز الانذار:

7-1- الزبائن CoAP Client:

بسبب عدم امتلاك كاميرا، تم الاستفادة من الكاميرا المقدمة من قبل الهاتف المحمول دون الاستفادة من أي خدمات أخرى موجودة فيه، وتم بناء تطبيق على نظام Android يوفر المطلوب، سمي CameraCoAP باستخدام برنامج Android Studio، حيث تم تضمين مكتبة بروتوكول CoAP المخصصة للغة الجافا والتي تدعى Californium، والاستعانة بمكتبة Scandium لتحقيق بروتوكول DTLS على بروتوكول CoAP.

يحتوي التطبيق على الواجهة كما في الشكل (6)، حيث يوجد في الواجهة زرين، زر التشغيل Start وزر الايقاف Stop ، ويوجد زر للإعدادات Setting.



الشكل (6) : تطبيق CameraCoAP

عند الضغط على زر Start يبدأ الزبون بالقيام بعملية المصافحة مع المخدم المحدد بالـ URI لتكوين قناة اتصال آمنة، كما يبدأ بالتقاط الصور كل فترة زمنية محددة بقيمة المؤقت، بعد ذلك يقوم بتوليد نيسب مهمته توليد زبون CoAP Client يقوم بإرسال طلب Request باستخدام بروتوكول CoAP إلى مخدم CoAP Server له الشكل التالي:

POST URI Payload

الطلب POST يعني أنه سيتم اضافة مورد محدد بالحمولة إلى المخدم .

Uniform Resource Identifier (URI) أو معرف الموارد الموحد له الشكل: (coap://ServerIpAddress:ServerPortNumber/image) ويحدد عنوان المخدم و رقم المنفذ (الافتراضي هو 5684 في حال استخدام البروتوكول DTLS).

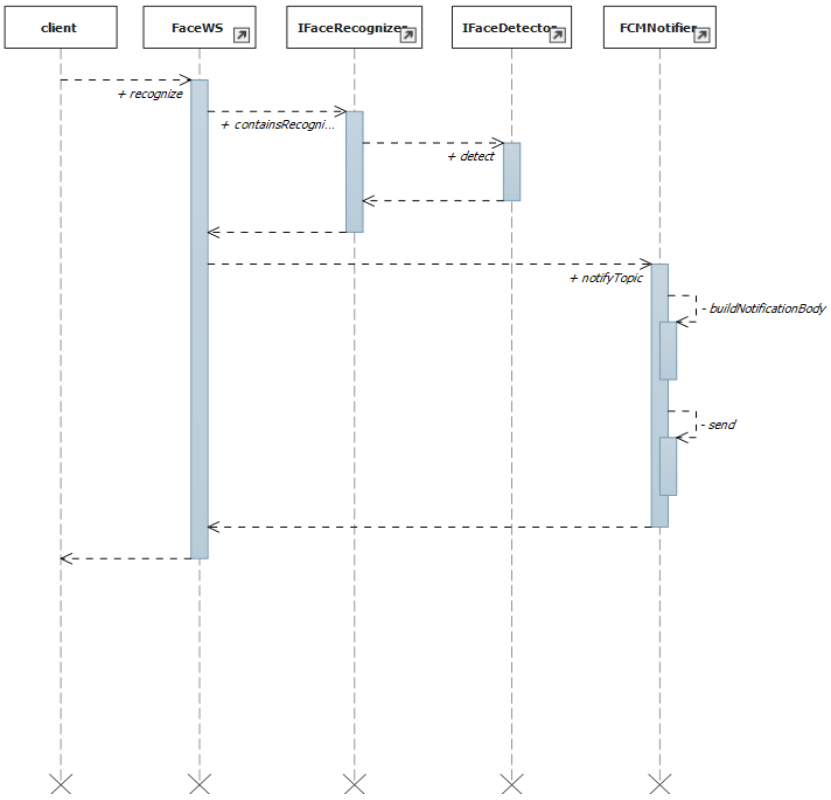
الحمولة Payload عبارة عن الصورة التي تم التقاطها بعد تحويلها إلى سلسلة من البايتات.

تم تعديل بعض القيم الأساسية الموجودة في التحقيق الحالي Scandium ليستطيع دعم الصور في عملية الارسال والاستقبال، عن طريق تعديل قيم حجم البيانات المسموح ارسالها والمسموح استقبالها، كما تم تطبيق عدة نياسب لكي يتم ارسال الصور بالشكل المطلوب حيث أن حجمها سيكون أكبر من حجم النص، خاصة أن التحقيق المتوفر مصمم من أجل نص فقط.

7-2- **السحابة FaceWS**: تم انجاز السحابة باستخدام لغة java على بيئة التطويرEclipse Neon. تقوم السحابة في البداية باستقبال الصور من تطبيق الموبايل ومن ثم حفظها بمجلد على القرص D ليتم معالجتها بشكل محلي باستخدام مكتبة OpenCV.

يوجد في السحابة مخدم CoAP مضمنة به إطار العمل Californium[8]، حيث تمت تهيئته بالقيم المناسبة ومن ثم اضافة مورد إلى المخدّم يدعى image، هذا المورد يوفر الطرائقGET، POST، PUT ، DELETE (إحضار مورد من المخدم ، إرسال مورد إلى المخدم ، حذف مورد من المخدم ، تعديل مورد في المخدم) للزبائن التي تقوم بإرسال الطلبات.

تم كتابة الكود الخاص بالمخدم الذي يستخدم بروتوكول DTLS كنوع من الحماية بالاستفادة من Scandium[9]، يقوم المخدم باستقبال الصور من الزبائن عبر المنفذ 5684 و يستخدم الخوارزمية ECC [17]ومفاتيح خاصة متفق عليها بين الزبون والمخدم، حيث تم استخدام مفاتيح متوفرة من قبل Scandium بغرض التجربة والدراسات. تم في تطبيق الأندرويد CameraCoAP السابق استخدام الطريقة POST مع المورد image، والتي تقوم بارسال سلسلة من البايتات المشفرة كحمولة مع الطريقة POST، والتي تعبر عن الصورة وتمرير الصورة إلى خدمة تمييز الوجوه. المخدم يستقبل الصورة ويقوم بفك تشفير الحمولة بالاعتماد على خوارزميات التشفير و المفاتيح التي تم الاتفاق عليها في عملية المصافحة الخاصة ببروتوكول DTLS ومن ثم يقوم بتمرير الصورة لتطبيق خدمة تمييز الوجوه المطلوبة. الخدمات التي تقدمها السحابة هي خدمة تحديد الوجوه وتمييزها، حيث تم تحقيق كل من الواجهات عن طريق مكتبة OpenCV المحققة في الصفوف OpenCVFaceDetector، OpenCVFaceRecognizer. يمثل الشكل (7) المخطط التسلسلي (Sequence Diagram) لخدمة Face Recognize التي توفرها السحابة .



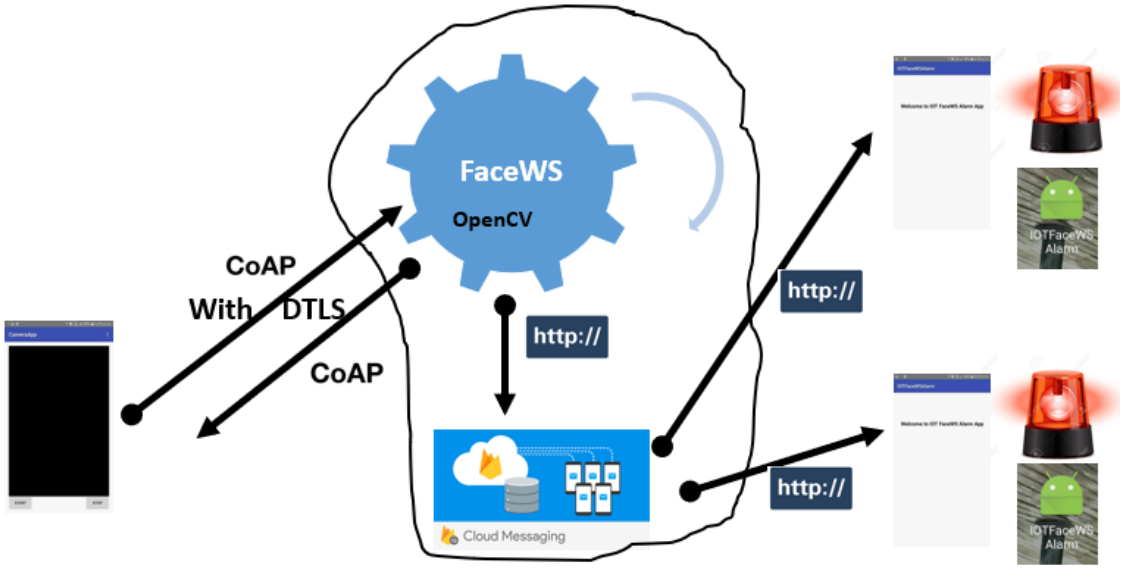
الشكل (7) : المخطط التسلسلي لخدمة Face Recognize

7-3- جهاز الانذار:

بسبب عدم امتلاك جهاز انذار لاستخدامه في التطبيق العملي، تم بناء تطبيق أندرويد يدعى بـIOTFaceWSAlarm تظهر واجهته في الشكل (8). عند تنصيب التطبيق على الجهاز المحمول يقوم بتسجيل نفسه في قاعدة بيانات تدعى Alarm منشأة باستخدام تقنية Firebase Cloud Messaging. تقوم السحابة بعد معالجة الصورة بتحديد و تمييز الوجوه، ومن ثم ووفقاً للخدمة المطلوبة وعند توفر الشروط المناسبة ترسل الى FCM طلب لإرسال الاشعارات إلى كل الأجهزة المسجلة والتي تقوم بدورها بإصدار صوت صافرة إنذار.

الشكل (8) : واجهة تطبيق IOTFaceWSAlarm

في النهاية يتكون لدينا البنية الجديدة موضحة في الشكل (9):



الشكل (9) : بنية IoTCloud مع حماية

النتائج والمناقشة:

سنقوم بتقييم الأداء من الناحية الأمنية ومن ناحية العبء على العقد.

1- تقييم البنية من الناحية الأمنية:

من الممكن أن يتعرض بروتوكول CoAP لأنماط متعددة من الهجمات موضحة في RFC رقم (7252) [6]. أحد هذه الهجمات هي هجمة الرجل في المنتصف man-in-the-middle، حيث يقوم المهاجم باعتراض المحادثة بين الزبون والمخدم، بحيث تظهر المحادثة وكأنها تجري بينهما مباشرةً، ولكن في الحقيقة يتم التحكم بها من قبل الشخص المهاجم، حيث يمكنه عرض أو إضافة أو إزالة أو تعديل أو استبدال الرسائل التي يتم تبادلها في هذه المحادثة. ولكنه لن يتمكن من تحقيق هدفه وذلك بسبب أن مجموعات التشفير المدعومة في بروتوكول DTLS لا تدعم الجلسات المجهولة. حيث إذا تمت مصادقة المخدم باستخدام بنية أساسية للمفتاح العام، لن يكون لدى المهاجم إمكانية لتزوير رسالة ServerKeyExchange الخاصة بالمخدم والتي تم توقيعها بالمفتاح الخاص المناسب خلال عملية المصافحة. من الممكن أن يقوم المهاجم بتطبيق الهجوم من نوع الحرمان من الخدمة DoS من خلال إرسال سلسلة من طلبات بدء المصافحة بهدف إغراق النظام، مما يجعل من الصعب على الزبائن الوصول للمخدم، وبالتالي استهلاك موارد زائدة على المخدم و إضافة عبء إضافي عليه. أو يمكن أن يقوم المهاجم بإرسال رسالة بدء الاتصال بمصدر مزيف. ولكنه لن يتمكن من تحقيق هدفه وذلك بسبب أن بروتوكول DTLS يستخدم تقنية ملفات تعريف الارتباط عديمة الحالة Cookie لحماية النظام من هجمات DoS. عندما يرسل الزبون رسالة ClientHello إلى المخدم، الذي بدوره يستجيب بإرسال رسالة HelloVerifyRequest تحتوي على ملف تعريف الارتباط. ثم يستجيب الزبون مرة أخرى بإرساله رسالة ClientHello مضافاً إليها ملف تعريف الارتباط الذي استقبله في رسالة HelloVerifyRequest. يتحقق المخدم من ملف تعريف الارتباط ويستمر في عملية المصافحة فقط إذا كان ملف تعريف الارتباط صالحاً.

يمكن إيقاف هجمات DoS بعناوين IP المزيفة باستخدام هذه الآلية لأنها تفرض على المهاجم تلقي ملفات تعريف الارتباط، ولكن لاتزال هذه الطريقة لا تضمن أي دفاع ضد هجوم DoS عند استخدام المهاجم لعنوان IP صالح.

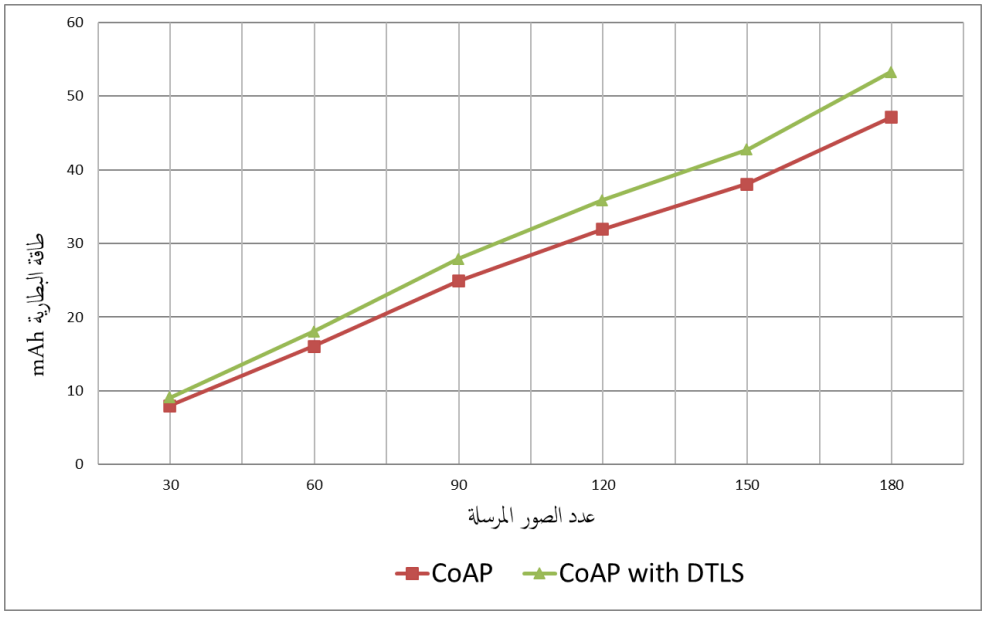
الجدير بالذكر أن استخدام الخوارزمية ECC في البروتوكول يعتبراً خياراً رائعاً وذلك لأنها لا تستهلك موارد حسابية مقارنة بالخوارزميات المعروفة مثل RSA، حيث تستخدم أطوال مفاتيح أصغر بكثير مع توفيرها نفس المستوى من الحماية الأمنية.

2- تقييم البنية من ناحية العبء على الأشياء:

تم المقارنة بين البنية العملية الأولى و البنية العملية الثانية الموضحتين في الشكلين (4) و(5)، حيث تم في كل بنية:

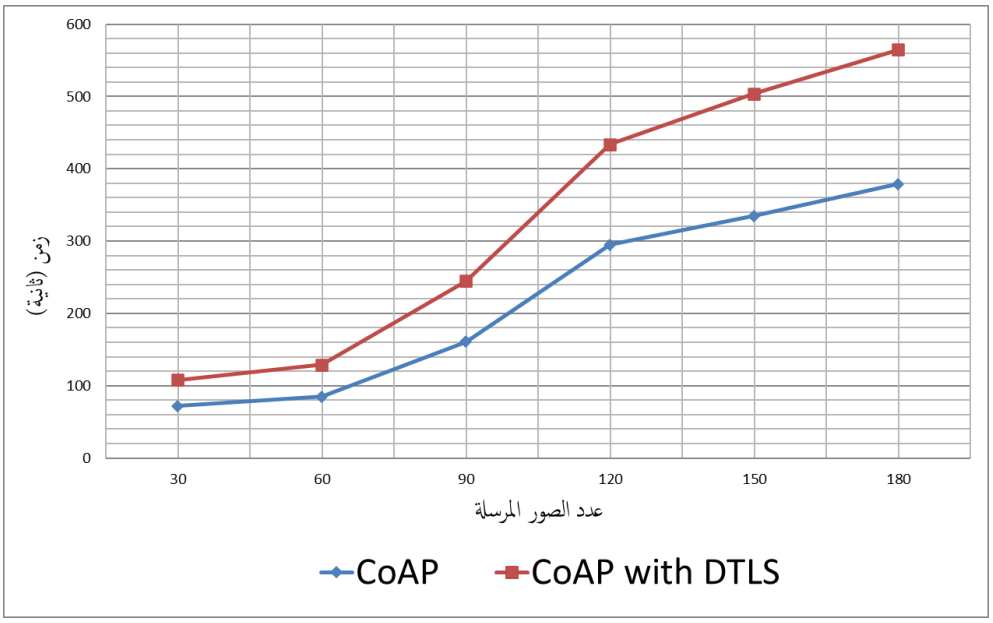
- ارسال عدد محدد من الصور (30 – 60 – 90 – 120 – 150 – 180) من الكاميرا إلى السحابة التي تقوم بالمهمة المطلوبة.
- قياس استهلاك البطارية وزمن استخدام المعالج في كل من السيناريوهات السابقة بالاعتماد على استهلاك التطبيق في إعدادات الموبايل وحفظ النتائج.

مخطط استهلاك البطارية من جهة الزبون : يعتبر استهلاك الطاقة للأجهزة المقيدة في انترنت الأشياء مصدر قلق، لأن معظم الأجهزة المقيدة لديها موارد طاقة محدودة وبالتالي يجب دراسة العبء الذي يفرضه تطبيق الحماية على بروتوكولات التواصل، خاصة فيما يتعلق بموضوع استهلاك الطاقة. نلاحظ في الشكل (10) والذي يظهر استهلاك البطارية من جهة الزبون في حال استخدام بروتوكول CoAP مع حماية من جهة و بدونها من جهة أخرى. يوجد زيادة في استهلاك البطارية عند استخدام بروتوكول DTLS كطريقة لتأمين الحماية بمقدار 12% تقريباً. يعود سبب هذه الزيادة إلى الجهد الذي يبزله الزبون في عملية المصافحة بالإضافة إلى الجهد المبزول في عملية تشفير الحمولة قبل ارسالها إلى المخدم وهذا يؤدي بدوره إلى استهلاك من طاقة البطارية.



الشكل (10) : مخطط استهلاك البطارية من جهة الزبون

مخطط زمن استخدام المعالج : وهو موضح في الشكل (11)، حيث نلاحظ أنه يوجد زيادة ملحوظة في زمن استخدام المعالج تصل إلى الضعف تقريباً، يعود ذلك إلى الجهد المبذول في عملية تشفير كل الصور ذات الحجم الكبير نسبياً قبل أن يتم ارسالها إلى المخدم.



الشكل (11) : مخطط زمن استخدام المعالج من جهة الزبون

على الرغم من زيادة نسبة استهلاك البطارية وزيادة زمن استخدام المعالج بزيادة عدد الصور المرسلة إلا أن نسب الزيادة تعتبر مقبولة نسبياً كوننا نحقق في البنية المقترحة جانب أكثر أهمية وهو الجانب الأمني. وعند مقارنة نسبة استهلاك الطاقة مع الدراسة المرجعية [5] التي تستخدم نوع حمولة نص نجد أن زيادة تقريباً 2% في استهلاك الطاقة بين الدراستين. وهذه الزيادة تعتبر مقبولة مقارنة بالفرق بين حجم الحمولة في كلتا الحالتين.

الاستنتاجات والتوصيات:

قمنا في هذا البحث بتطبيق حماية على بروتوكول التطبيقات المقيدة CoAP في نظام لانترنت الأشياء يستخدم لتمييز الوجوه معتمد على السحابة، باستخدام بروتوكول DTLS مع نوع حمولة صور، كما تم مقارنة السيناريو المنجز مع حماية مع نفس السيناريو من دون حماية. نلاحظ أن استخدام بروتوكول DTLS كنوع من الحماية لبروتوكول CoAP يوفر المطلوب ويقدم حماية ضد عدد من الهجمات، ولكنه يجعل الحمل أكبر على الأشياء (يستهلك من طاقة البطارية و يزيد زمن استخدام المعالج)، حيث يزيد استهلاك البطارية بنسبة 12% تقريباً كما أنه يزيد زمن استخدام المعالج بمقدار الضعف. يعود سبب الزيادة إلى الوقت اللازم للقيام بعملية المصافحة قبل البدأ بعملية الارسال والاستقبال بالاضافة إلى الوقت اللازم لتشفير الرسائل المرسلة. وبالتالي يمكننا القول أن بروتوكول DTLS يقدم المطلوب من الناحية الأمنية لكنه يفرض عبء على العقد. بكافة الأحوال فإن تطبيق أي حل أمني يستوجب زيادة في استهلاك الموارد، نسعى دائماً لإيجاد حلول أمنية تقدم مستويات عالية من الحماية مع عدم زيادة ملحوظة في استهلاك الموارد،

نعتبر الزيادة المسجلة في الموارد في البنية المقترحة من قبلنا قليلة الأهمية لأن طبيعة التطبيق تقتضي استخدام كاميرات من المفترض وجودها في أماكن ثابتة بتغذية دائمة، ولكن تبرز أهمية استهلاك الموارد في تطبيقات أخرى.

نسعى مستقبلاً وبالاستفادة من البنية المنجزة، بتطبيق أشكال جديدة من الحماية ومقارنتها مع الحل المقترح بطرق مختلفة للحصول على البروتوكول الذي يقدم الحماية المطلوبة لبروتوكول CoAP مع عبء أقل على العقد المقيدة.

References:

[1] ROSE, K. ; ELDRIDGE, S. ; CHAPIN, L. *The internet of things: An overview.*The Internet Society (ISOC), Oct. 2015, (pp. 1-50).

[2] BOTTA, A. ; DE DONATO, W. ; PERSICO, V. ; PESCAPE, A. *On the integration of cloud computing and internet of things*. In Future Internet of Things and Cloud (FiCloud), 2014 International Conference on Barcelona, Spain, IEEE , Aug. 2014, (pp. 23-30).

[3] JOSHI, M. ; KAUR, B.P. , *CoAP Protocol for Constrained Networks.* Published in IJ Wireless and Microwave Technologies, vol.5, no.6, NOV. 2015, (pp.1-10).

[4] Kovatsch, M., Lanter, M. and Shelby, Z., *Californium: Scalable cloud services for the internet of things with coap. In Internet of Things (IOT)*, 2014 International Conference on the IEEE, 2014, October, (pp. 1-6). .

[5] Alamri, MH. *Securing the Constrained Application Protocol (CoAP) for the Internet of Things (IoT)*, (2017).

[6] SHELBY, Z. ; HARTKE, K. ; BORMANN, C. *The constrained application protocol (CoAP)*. Internet Engineering Task Force (IETF), ISSN: 2070-1721, June. 2014, < https://tools.ietf.org/html/rfc7252>.

[7] RFC 7252 *Constrained Application Protocol Implementations* , Feb.2021. <https://coap.technology/impls.html>.

[8] *Eclipse Californium*, Feb. 2021. <https://github.com/eclipse/californium>.

[9] *Eclipse Scandium*, Feb. 2021. <https://github.com/eclipse/californium/tree/master/scandium-core>.

[10] Eric Rescorla and Nagendra Modadugu, *Datagram transport layer security version 1.2*". 2012.

[11] Fiterau-Brostean, P., Jonsson, B., Merget, R., de Ruiter, J., Sagonas, K., & Somorovsky, J. *Analysis of {DTLS} Implementations Using Protocol State Fuzzing*. In 29th {USENIX} Security Symposium ({USENIX} Security 20),2020, (pp. 2523-2540).

[12*] The Eclipse Foundation*, 23Jan. 2021. <https://eclipse.org/californium/>.

[13] GÜRKAN, Ali Tunca. *Security analysis of coap and dtls protocols for internet of things applications*. 2019. Master's Thesis. Işık Üniversitesi.

[14] Google Firebase Cloud Messaging, 15Jan. 2021. <https://firebase.google.com/docs/cloud-messaging/>.

[15] OpenCV Documentation, 15Feb. 2021. < http://opencv.org/>

[16] Dandeh.R, Ghadeer.R. *Building a cloud-based Internet of Things system using the CoAP protocol*, Tishreen University Journal -Engineering Sciences Series. 2017 Vol. 39 No. 4.

[17] Nir, Yoav, Simon Josefsson, and Manuel Pégourié-Gonnard, "*Elliptic curve cryptography (ecc) cipher suites for transport layer security (tls) versions 1.2 and earlier.*" Internet Requests for Comments, RFC Editor, RFC 8422 (2018).