AUDITeD: análisis y arquitectura de Testigo Digital para Dispositivos Android

Silvia Cuenca
Ebury
29005 Málaga (España)
silvia.cuenca@ebury.com

Ana Nieto
NICS Lab, Universidad de Málaga
Edificio Ada Byron, 29010 Málaga (España)
nieto@lcc.uma.es

***Resumen*—La informática forense está en constante evolución para adaptarse a las nuevas tecnologías y formas de comunicación. Como parte de los nuevos paradigmas en torno a 2012 surgieron los primeros trabajos sobre IoT-Forensics, y en 2015 se propuso una solución para permitir a los dispositivos personales ser cooperantes en la gestión de las evidencias electrónicas. En aquel entonces los dispositivos aún no disponían de toda la funcionalidad requerida para hacer pruebas sobre dicho concepto, algo que cambió en 2019 con la inclusión de nuevas funcionalidades en torno al *secure element* en los teléfonos Android de última generación. Surge así la posibilidad de probar dicho enfoque en esta plataforma en aras de analizar la adecuación de los trabajos iniciales sobre este campo, y los requisitos que aún deben satisfacerse para permitir el despliegue total de los testigos digitales. En este trabajo se definirá una arquitectura para que los dispositivos Android puedan actuar como testigos digitales (AUDITeD), haciendo énfasis en las limitaciones actuales y por tanto en futuras vías de investigación que permitan ayudar a mejorar las soluciones existentes.**

***Index Terms*—IoT-Forensics, testigo digital, elemento seguro, evidencia digital, digital forense, ejecución segura.**

I. Introducción

Los dispositivos personales son preciadas fuentes de datos acerca de un individuo y su entorno. La dependencia de los usuarios hacia sus dispositivos unido a que están equipados con sensores capaces de recabar datos del entorno, los hacen muy útiles en las investigaciones digitales [1]. Las primeras referencias de estudios de informática forense sobre dispositivos móviles datan de lo que se conoce como la época dorada de la informática forense (finales de los 90 principios del 2000) [2]. Desde entonces, las herramientas forenses especializadas han continuado su evolución permitiendo la extracción de los datos de diferentes versiones de dispositivos, inclusive aquellas con arquitectura de seguridad diseñada para proteger los datos del usuario del terminal.

Las fases de las que suele constar una investigación digital forense se encuentran recogidas en los estándares ISO/IEC 27037:2012 (identificación, colección, adquisición, preservación) e ISO/IEC 27042:2015 (investigación, análisis, interpretación e informe). Tanto en estos estándares como en otros relacionados (p.ej. ISO/IEC 27043:2015 e ISO/IEC 30121:2015) los dispositivos personales forman parte de las fuentes de obtención de evidencias digitales, contenedores de los que extraer datos que posteriormente serán analizados.

Hasta ahora esta forma de proceder se ha demostrado suficiente para resolver los casos que se han presentado, pero como siempre la informática forense continuará evolucionando para afrontar nuevos desafíos derivados de la inclusión de nuevos paradigmas tecnológicos. Por ejemplo, en estudios anteriores se ha puesto de manifiesto que la Internet de las Cosas (IoT, por sus siglas en inglés) presenta nuevos retos que solventar para la informática forense tradicional [3]. El foco, que antes se situaba en los dispositivos como contenedores de datos cambia para situarlos en una posición de cooperadores del entorno, a fin de recabar evidencias digitales que pueden pasar desapercibidas para un usuario, pero no para un dispositivo. Esta es la idea que hay detrás del enfoque de *testigo digital* definido previamente en [4].

La contribución principal de este artículo es analizar si actualmente el concepto de testigo digital es implementable en sistemas Android, proponiendo una arquitectura (AUDITeD) que permita realizar una prueba de concepto. Uno de los inconvenientes de los que se parte es la carencia de dispositivos que cuenten con todos los requisitos del testigo digital (p.ej. seguridad embebida con opciones de desarrollo para terceros, similar al *trusted platform module* (TPM)). Este problema se solventó en parte gracias a la puesta en el mercado del dispositivo Android Google Pixel 3, que emplearemos para la demostración del prototipo desarrollado en base a la arquitectura propuesta. Este dispositivo ha permitido por tanto definir una solución para adaptar el dispositivo digital a plataformas Android, y probar dicha solución hasta consolidar la arquitectura propuesta.

Este artículo se estructura como sigue. El trabajo relacionado se analiza en la Sección II. En la Sección III se analizan los componentes principales de la arquitectura de seguridad de Android de los que se nutre la solución. El diseño del testigo digital para la plataforma Android se propone en la Sección IV, mientras que el prototipo específico desarrollado para evaluar la arquitectura propuesta se describe en la Sección V. Se incluye además un análisis del rendimiento para determinar los efectos de los mecanismos de cifrado en la base de datos de evidencia digital. Finalmente, en la Sección VI se sintetizan las conclusiones principales y las direcciones futuras.

II. Trabajo relacionado

Los trabajos de informática forense en dispositivos Android suelen centrarse en analizar el dispositivo como contenedor de evidencias digitales, como por ejemplo las metodologías para la adquisición, algunas de ellas propuestas en [5], [6], [7], [8]. Otros, aportan análisis específicos sobre los rastros que dejan las aplicaciones en los dispositivos, comparativas [9], [10], [11], [1], o bien introducen soluciones para alguna parcela de la informática forense, como lo es la adquisición de memoria volátil [11].

En nuestro caso la perspectiva es diferente; el dispositivo será de un usuario que quiere emplearlo para recabar evidencias de forma segura, apoyándose en la seguridad nativa que ofrecen las nuevas plataformas. Por lo tanto, consideramos que los trabajos que más se relacionan con la solución propuesta son aquellos que emplean elementos seguros anti-tampering en el contexto de evidencias digitales e informática forense, o bien relacionados con la gestión de evidencias digitales.

Por ejemplo, en [12] el autor describe como usar un TPM para adquirir evidencias digitales y enviarlas a los puntos finales para su posterior análisis. Aunque la solución aportada no considera las opciones de salto entre dispositivos, sí se destacan las ventajas en el uso de este tipo de hardware para garantizar un núcleo de confianza para el proceso completo de adquisición de evidencias digitales. En [13] se introduce la noción de cadenas de custiodia digitales seguras. El autor también propone el uso de chips TPM para proteger la generación de evidencias digitales en los puntos de adquisión. Esta solución se centra en los beneficios de usar el TPM en entornos distribuidos. Sin embargo, como en el anterior, no hay resultados tangibles de la solución propuesta. Se proponen ideas similares en [14], [15], ambos centrados en el TPM y las SVM (*Secure Virtual Machines*).

Finalmente, *DroidWatch* [16] es una solución muy interesante que incluye una aplicacion Android y un servidor web remoto para recibir la evidencia digital. La solución considera diferentes eventos que pueden capturarse en el teléfono inteligente y enviarse al servidor. Sin embargo, *DroidWatch* no considera fortalezer la captura de dicha evidencia digital utilizando los nuevos elementos de seguridad en las arquitecturas de Android, como es el caso del *secure element*, o la posibilidad de realizar un diseño orientado a extender la solución a múltiples tipos de dispositivos IoT.

La noción de *digital witness* se introduce en [4] como solución para el análisis forense en escenarios IoT (*IoT-Forensics*). Se basa en la inclusión de chips de seguridad integrados para implementar una *Cadena de Custodia Digital* [4]. Por lo tanto, dicha solución define cómo la evidencia digital debe ser adquirida y preservada por los dispositivos IoT (denotados como testigos digitales) y luego enviada a otras entidades autorizadas que pueden ser otros testigos digitales en la cadena. La solución no obstante debe ser adaptada para permitir su implementación en plataformas móviles que sirvan de puente entre el usuario y sus dispositivos IoT.

A diferencia de las contribuciones anteriores en este campo, este artículo proporciona el diseño de una solución para adquirir evidencias digitales y manejarlas utilizando el elemento seguro en entornos Android. Esta solución se basa en el concepto de testigo digital (*Digital Witness*) que se encuentra muy maduro, analizado en varios artículos, incluido el análisis de soluciones de privacidad [3]. Sin embargo, se necesita un análisis tomando como punto de partida una plataforma específica, en este caso Android, para estudiar la idoneidad de los requisitos del testigo y su flexibilidad para adaptarse a los diferentes contextos. Con el objetivo de ofrecer un diseño de la arquitectura de testigo digital para Android completo, se ha implementado un prototipo cuyo código se encuentra disponible en GitHub (https://github.com/silvia-cr/selvia) para servir a la comunidad. Esto ha permitido completar la arquitectura, mejorándola para su futura exportación a otros sistemas.

III. Seguridad en Android

Considerando la arquitectura dividida en 5 niveles que presenta Android [17], la solución propuesta aprovechará las capacidades de seguridad ofrecidas por el kernel y por la capa de abstracción de hardware (HAL). En concreto, la capa del kernel de Linux permite aprovechar funciones de seguridad claves de linux, además de utilizar otros métodos de seguridad propios de Android para proveer comunicaciones seguras entre los procesos. Como se puede comprobar en [18], estos métodos son principalmente el aislamiento en la ejecución de aplicaciones en un sandbox, una partición para el sistema y ejecución en modo seguro, sistema de ficheros basados en permisos, criptografía y arranque seguro, entre otros. En cuanto a la capa *HAL*, esta ofrece interfaces para utilizar las capacidades del hardware en el *framework* de la API de Java. Esta capa consiste en módulos de biblioteca, cada uno de los cuales implementa una interfaz para un componente hardware específico. Cuando el framework de una API, por ejemplo la API de Java, intenta acceder al hardware del dispositivo, el sistema carga el módulo de la biblioteca para ese componente hardware.

En concreto, parte del trabajo desarrollado durante este paper se centra en cómo aprovechar el *secure element* (SE) que integran los nuevos modelos [19]. Un SE permite almacenar información criptográfica, y representa una tercera parte confiable en el propio dispositivo. Podría considerarse una evolución de otras soluciones previas como el TPM, pero adaptado a dispositivos personales. En concreto nos centramos en la modalidad de *SE* integrado en placa, por restricciones del testigo digital. Actualmente los desarrolladores de terceros cuentan, principalmente, con dos mecanismos para permitir la interacción con el SE: *Android Keystore System* y *Open Mobile API*. Sus ventajas e inconvenientes se resumen en la Tabla I.

III-A. Android Keystore System

Android Keystore System fue introducido en Android de forma nativa en la versión API 18 (Android 4.3). Este sistema permite almacenar claves en un *almacén de claves*, restringiendo de esta manera su uso y acceso. El almacén de claves en este contexto se refiere a la ubicación dónde se almacenarán las claves criptográficas de forma segura, por tanto tenemos que el sistema *Keystore* es la interfaz proporcionada por Android para comunicarse con el almacén de claves del dispositivo, en este caso el *secure element*. Las características de seguridad principales de este sistema se describen en [20]: i) mecanismos para la prevención de extracción de claves, ii) el módulo de seguridad de hardware y iii) la autorización de uso de las claves. Para almacenar las claves mediante *Keystore*, se asocian alias a estas, de forma que sólo el alias sale del almacén de claves.

III-B. Open Mobile API

Otra de las tecnologías que Android aporta para utilizar el *secure element* es *Open Mobile API* (*OMAPI*). Esta característica se trata de un mecanismo que permite a las aplicaciones autorizadas a comunicarse con los *applets* que se encuentran dentro del *secure element* [21], en concreto

Tabla I
COMPARACIÓN DE OPCIONES DE ACCESO AL ELEMENTO SEGURO

	Android Keystore System	**OMAPI**
Ventajas	- Gestión de claves - Las claves no abandonan nunca el hardware seguro - Cada clave tiene un alias (el alias es el que se conoce) - Versiones de Android mayores o iguales a API 18 (mayor número de dispositivos compatibles)	- Indica explícitamente la utilización del *secure element*, con lo que se cumple la restricción del testigo digital de dependencia con hardware seguro embebido.
Inconvenientes	- Solo se almacenan claves, no cadenas personalizadas - Aplicaciones diferentes con mismo nombre de paquete comparten claves	- Versiones de Android mayores o iguales a API 28 (menor número de dispositivos compatibles) - Release de julio de 2018, muy poca información

se trata de la especificación de una API para permitir la utilización de estos por las aplicaciones móviles. Esta librería fue introducida de forma nativa en el sistema a partir de la API 28 (Android 9), siendo necesaria su inclusión a mano en versiones anteriores si querían utilizarse sus características. Utilizando OMAPI, las comunicaciones con el SE se realizan mediante mensajes *APDU* cumpliendo con el estándar *ISO/IEC 7816-4*. Estos mensajes *APDU* son la unidad de comunicación entre una *smart card* y su lector [22]. Estos mensajes estarían divididos en dos categorías (comando y respuesta), teniendo cada una de ellos su propia estructura.

La especificación oficial de esta librería [23] fue publicada en julio de 2018, al igual que la especificación concreta para la plataforma Android [24]. En el momento de la realización de este trabajo encontramos poca información o ejemplos concretos de su uso para este proyecto, más allá de las citadas especificaciones oficiales.

III-C. Resumen

Basado en las características mencionadas previamente, *OMAPI* ha sido escogido para acceder al *secure element* para almacenar hashes. Como se observa en la Tabla I, los inconvenientes de *Android Keystore System* son muy importantes para esta solución, debido a que en este caso es imprescindible que solo la aplicación desarrollada tengo acceso a las claves generadas. Asimismo, *Android Keystore System* ha sido escogido para almacenar claves cifradas, pues es la mejor opción para utilizar esta funcionalidad. El uso de estas características debe tenerse en cuenta para la sección V.

IV. ARQUITECTURA AUDITED

El testigo digital debe cumplir con una serie de requisitos básicos para ser considerado como tal, detallados en [4]: (i) disponer de un núcleo de confianza, (ii) ser gestionado o estar autorizado por un responsable humano, (iii) atender a un conjunto de políticas de configuración, entre las que están las opciones de seguridad y gestión de las evidencias digitales, (iv) contar con un registro de evidencias, y (v) ser capaz de transmitir las evidencias a otras entidades autorizadas. Estos requisitos son definidos basados en los estándares actuales de gestión de evidencias digitales y las necesidades específicas del análisis forense en IoT identificadas en [4].

Considerando dichos requisitos y las soluciones que aporta Android (sección III), la Figura 1 describe la arquitectura propuesta, que se basa en la arquitectura digital de un testigo digital [4] pero adaptada a las posibilidades que ofrece la plataforma Android, recibiendo el nombre de AUDITeD. La implementación de la arquitectura dependerá de las características de la plataforma final, en esta sección se describen los componentes básicos para Android y cómo se ha tenido en cuenta tanto el rol de los clientes o usuarios propietarios del dispositivo mediante una aplicación que funcione como testigo digital y los puntos oficiales de recogida de las evidencias (OCP, por sus siglas en inglés, *official collection point*) [3].

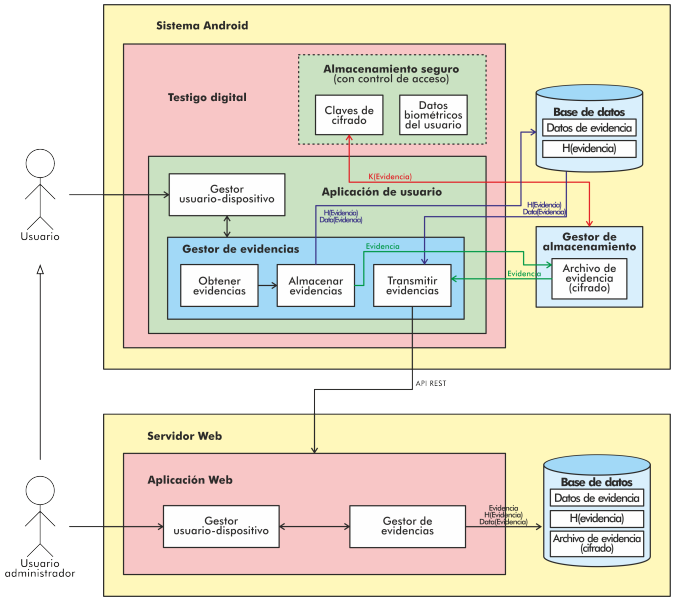

Figura 1. Arquitectura AUDITeD - Testigo digital modular para Android

Hay que tener en cuenta que la solución propuesta cuenta con dos componentes que son similares en el testigo digital y el OCP: el gestor usuario-dispositivo y el gestor de evidencias digitales. Aunque su propósito es bastante similar, su diseño (y desarrollo) es muy diferente en ambos casos. En particular, los componentes para el testigo digital en el lado del cliente requieren considerar muchos más requisitos y pasos para el manejo de evidencias digitales y también coordinar las acciones y decisiones finales que se tomarán con el usuario. Además, la identidad del usuario se verifica mediante biometría aprovechando la funcionalidad del teléfono móvil. El OCP deberá reunir la condición de laboratorio apto para la gestión de evidencias electrónicas, aplicando las recomendaciones para la gestión final de las evidencias indicadas en las normas y leyes vigentes, fuera del ámbito de este trabajo. Las evidencias en él recopiladas serán analizadas por un grupo de expertos al que se deberá garantizar acceso de forma segura.

A continuación se describen las decisiones de diseño que afectan a ambas partes de la comunicación.

IV-A. Aplicación del Testigo Digital (cliente)

Como se observa en la Figura 2, el gestor de operaciones entre el usuario y el dispositivo (Figura 1), se encuentra dividido en una serie de partes bien diferenciadas: por un lado estaría el *core* o núcleo, la parte que se mantendrá inmutable para cualquier dispositivo Android, y por otro los módulos que se añaden por flexibilidad para su adaptación a múltiples versiones de dispositivos. Por su función se ha denominado a dichos módulos conectores.

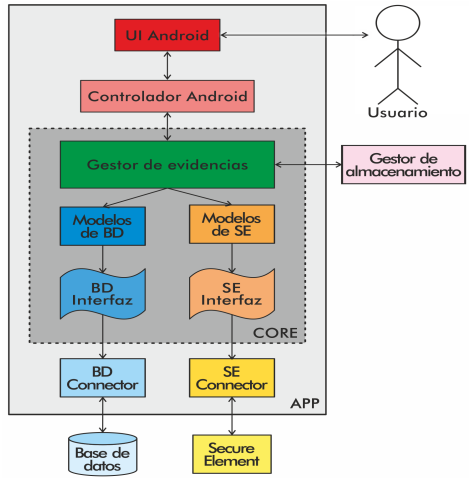

Figura 2. Componentes necesarios para el testigo digital móvil

IV-A1. Núcleo: El núcleo se encuentra dividido en tres módulos: (i) gestor de evidencias, (ii) modelos de almacenamiento e (iii) interfaces con los recursos.

El gestor de evidencias digitales es el módulo que se encuentra a más alto nivel, conteniendo la funcionalidad principal. El gestor se comunicará con el controlador de la plataforma Android y con el gestor de almacenamiento, utilizando para ello la funcionalidad aportada por los modelos de almacenamiento. Por tanto, es completamente independiente de la plataforma en la que se ejecute, y no tiene conocimiento del motor de base de datos o *secure element* utilizados. Es en este módulo dónde se construirá, con ayuda de los modelos, la adquisición y transmisión de evidencias al servidor.

Los modelos de almacenamiento permiten desacoplar al gestor del acceso a los recursos de la plataforma. Por una parte se considera el acceso a las bases de datos definidas para el almacenamiento de las evidencias digitales cifradas y por otra el acceso al elemento seguro para ofrecer la protección adicional requerida por los requisitos de testigo digital. Cabe destacar que, al igual que la capa del gestor de evidencias, esta capa no tendrá conocimiento del motor de base de datos utilizado, únicamente empleará la funcionalidad aportada por las interfaces para interactuar con dicho motor.

Finalmente, la capa de las interfaces es la encargada de mantener la relación de métodos y funciones que serán necesarios para la comunicación con la base de datos y el *secure element* del dispositivo. Estas interfaces se hacen imprescindibles para poder llevar a cabo la portabilidad del núcleo de la aplicación a otra plataforma, o incluso para mantener la plataforma, pero cambiar el motor de base de datos o el *secure element*. De esta forma, si se quiere implementar el testigo digital en Android, pero cambiar el motor de base de datos, únicamente sería necesario desarrollar el conector final con la base de datos implementando esta interfaz.

IV-A2. Punto oficial de recogida de evidencias digitales (servidor): El propósito del punto final de recogida de evidencias digitales es la recepción y el almacenamiento final de la evidencia antes de ser analizada por el experto o grupos de expertos autorizados por la autoridad competente o el cliente de ser una investigación privada. El OCP ha sido diseñado para permitir la recepción de diversos tipos de evidencias, y es independiente de la plataforma Android (Figura 3). Se plantea como un testigo digital fijo (debe ser capaz de obtener evidencias al igual que lo haría un testigo digital intermedio), que tiene por tanto mayor capacidad de almacenamiento.

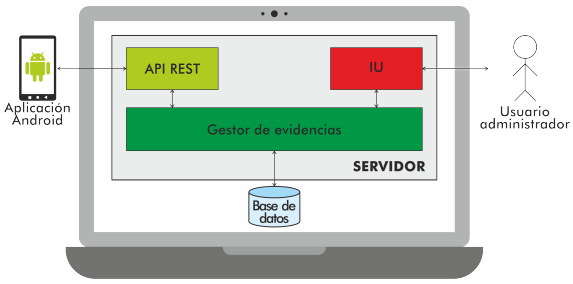

Figura 3. Componentes necesarios para el OCP (testigo digital fijo)

Disponemos de tres componentes principales en el OCP: (i) el Gestor de Evidencias Digitales para el Servidor (GEDS), (ii) la Interfaz con los Testigos Digitales (ITD) y (iii) la Interfaz con la Administración (IA).

El componente GEDS habilita la comunicación desde el testigo digital hacia el ITD. Este componente recibirá las evidencias digitales para ser almacenadas en la base de datos. Entonces, el GEDS podrá ser accedido por una entidad autorizada utilizando la IA. Por simplicidad en este escenario consideramos que el administrador y el investigador digital son la misma entidad. Sin embargo, en un entorno en el que se cuente con un equipo de trabajo para la investigación, ambos perfiles (administrador / investigador digital) deben estar separados, y el administrador únicamente debería poder configurar los accesos técnicos y nunca acceder a las evidencias almacenadas en la base de datos.

La ITD tiene el rol de negociar cómo el testigo digital (Aplicación Android en Figura 3) debe comunicarse con el OCP. Este componente envía la información del servidor al testigo digital para definir sus políticas. El testigo y el servidor se encuentran vinculados por medio de un registro inicial en la entidad OCP. El servidor no revela información crítica a las entidades externas que se identifican como testigo digital o testigos que no estén bajo su tutela. Esto se hace así, porque atendiendo a los requisitos de testigo digital definidos en [3], cualquier testigo digital debe ser aprobado por un OCP, por lo que se requiere un contacto inicial para la entrega de credenciales.

Finalmente, los responsables encargados de gestionar las evidencias electrónicas, así como los investigadores, accederían a través de interfaces dedicados conforme el rol específico del usuario autorizado. Una vez que las evidencias fueron alojadas en el OCP, el investigador digital tendrá las evidencias para analizar en pasos posteriores que van más allá del ámbito de este trabajo.

V. Prototipo y discusión de resultados

En esta sección se detalla el diseño del prototipo realizado conforme a la arquitectura AUDITeD propuesta en la sección IV. La herramienta desarrollada se ha denominado SELVIA (por sus siglas en inglés *Secure ELement-based digital eVIdence Acquisition System for Android*). Tanto el código de SELVIA, como información adicional se encuentran disponibles en GitHub (https://github.com/silvia-cr/selvia), incluyendo un vídeo demostrativo del uso de la herramienta.

De acuerdo con los resultados de las secciones previas, el dispositivo escogido para desarrollar el diseño en la plataforma Android debe de cumplir con las siguientes características:

1. Android Pie 9.0 o superior (version 28 o superior). Con el objetivo de poder hacer uso de OMAPI (ver sección III).
2. Secure Element (SE) integrado en la placa, por los requisitos del testigo digital.
3. Software para desarrollar utilidades seguras para el SE, en aras de implementar los requisitos deseados.
4. Autenticación usando la biometría para determinar la identidad del usuario.

Del listado anterior, las restricciones más fuertes han sido la primera y la tercera, limitando el número de dispositivos a terminales de última generación. En concreto el desarrollo del prototipo se ha realizado empleando un terminal Google Pixel 3, que incluye el chip Titán M (elemento seguro). El chip *Titán M* cuenta con el hardware seguro más avanzado hasta el momento de los incluidos en dispositivos Android. Este chip ha sido específicamente diseñado para el dispositivo *Google Pixel 3* para asegurar los datos y el sistema operativo. Se trata de tomar las mejores características de los chips Titán utilizados en *Google Cloud*, y adaptarlos a los dispositivos móviles [25]. El acceso al chip Titán M se llevará a cabo empleando los mecanismos analizados en la Sección III.

V-A. Pruebas de evaluación del prototipo

Las pruebas de evaluación comprueban la funcionalidad implementada: autenticación empleando huella, adquisición de evidencias gestionadas por medio del SE (almacenamiento cifrado y consulta, eliminar evidencias e histórico).

Respecto a la autenticación empleando la huella dactilar, se requiere cuando se inicia sesión en la aplicación y también antes del envío de un conjunto de evidencias al servidor. El testigo se plantea como una solución para que el usuario pueda decidir qué evidencias reportar. Todas las operaciones realizadas sobre las evidencias recopiladas dejarán rastro en un histórico por trazabilidad, que es eliminado sólo cuando el propio histórico ha sido enviado a un OCP y se ha recibido confirmación del destino. Todo el tráfico va cifrado y se usan certificados para verificar la identidad tanto del terminal del usuario (que se encuentra vinculado con el propietario, responsable en última instancia) y del receptor de las evidencias. La testificación anónima no se considera en este punto.

Los metadatos de la evidencia se almacenan en un campo a parte a modo de documentación adicional sobre la evidencia, entendiendo que no constituirá una prueba robusta, pero sí puede ser significativo cuando se cuenta con varias evidencias sobre el hecho analizado.

Encontramos las siguientes limitaciones:

- Autenticación de los usuarios pero sin control de acceso basado en roles.
- Escritura en el SE limitada y falta de documentación.
- Normalización de los datos.

Por una parte, Android ofrece la posibilidad de autenticar al usuario en la aplicación mediante biometría, pero no permite crear diferentes roles de acceso a la aplicación. Esto implica que un testigo digital no podría ser compartido por diferentes usuarios en esta plataforma. Tenemos que restringirlo a un único responsable humano.

AUDITeD emplea el SE para la gestión de las claves empleadas durante todo el proceso, pero también se plantea la escritura de hashes en el SE a modo de caja negra. Esto implica considerar las limitaciones de espacio en el SE para el uso de múltiples hashes sobre una misma evidencia, como sería lo deseable. Debido a las limitaciones de espacio, en el SE se escribiría sólo el hash MD5 de la evidencia, mientras que sería posible tener en cuenta tanto el MD5 como otros (SHA256) en la base de datos. En cuanto a la escritura en el SE se encuentra carente de documentación, aunque los comandos OMAPI son similares a la escritura en smart-cards.

Para la gestión de múltiples tipos de evidencias se requieren mecanismos que permitan normalizar los datos. En el caso de la prueba de concepto se ha puesto el foco en imágenes, pero otros datos (p.ej. logs de Android) podrían ser también exportados. Idealmente debería emplearse un formato común de evidencias que pueda ser empleado en dispositivos limitados en recursos, no sólo planteadas para plataformas como el OCP (como es el caso en las normas actuales). El formato DXML (*Digital Forensic XML*) podría explorarse en un futuro junto con otras opciones más cercanas a la solución propuesta, basada en API-request con el servidor, y por tanto empleando JSON. No hay aún estándares oficiales ni esquemas específicos sobre DXML.

Tabla II
Tiempo de ejecución y tamaño de datos no cifrados y cifrados

	BD no cifrada		BD cifrada	
#Inserciones	**Tiempo (s)**	**Tamaño (Kb)**	**Tiempo (s)**	**Tamaño (Kb)**
10	0,214	20	7,530	32
100	1,221	20	71,567	32
1000	7,503	80	722,516 (∼12min)	92
10000	78,674	600	7566,300 (∼2h10m)	624

V-B. Pruebas de rendimiento

Resulta previsible que los accesos a la base de datos (BD) cifrada afecten al rendimiento. El objetivo en este apartado es mostrar el efecto que tiene sobre la solución propuesta a fin de fomentar trabajos que permitan mejoras en este ámbito. Los datos mostrados en la Tabla II se corresponden a ejecuciones realizadas en el emulador, que corresponde a un dispositivo óptimo, dado que no tiene un uso real por parte de un usuario. Esto también permite tener datos centrados únicamente en el uso de la funcionalidad de testigo digital. Para medir los tiempos se han realizado las inserciones en 3 ejecuciones, y se ha calculado la media aritmética de estas. En cada inserción se ha realizado una conexión con la base de datos. En cuanto

al tamaño de archivo, éste no varía entre ejecuciones, siendo la base de datos cifrada ligeramente mayor.

Los resultados muestran un incremento significativo en tiempo. Las mejoras en este punto son difíciles de conseguir, o bien se prescinde del control de las evidencias por parte del usuario y se envían fueran del dispositivo sin almacenamiento en éste desde el momento de su adquisición - conllevaría en cualquier caso el cifrado para su transmisión, y un pre-análisis para determinar si se envía o no la evidencia en base a un conjunto de criterios - o bien se minimizan los datos a almacenar en la base de datos.

V-C. Discusión: vulnerabilidad CVE-2019-9465

Cabe destacar que la prueba de concepto se empleó un dispositivo Google Pixel 3 con chip Titán M, que salió al mercado en 2019. Este es el primer dispositivo que, por sus características, ha permitido un primer desarrollo de testigo digital que esperemos ayude a su portabilidad a otros entornos. La confianza en la seguridad del chip hizo que Google anunciase meses después de su lanzamiento recompensas en torno a los 500.000$ (y más tarde, en noviembre de 2019 1M$) por el descubrimiento de vulnerabilidades en su chip. A principios de 2020 se anunciaba la vulnerabilidad CVE-2019-9465 que afecta al chip Titán M [26].

Esto no afecta al prototipado en tanto a que se centraba en analizar la viabilidad del testigo digital con la tecnología disponible, y se ha comprobado que los componentes pueden implementarse a día de hoy en un dispositivo personal. Además, permite tomar feedback que ayude a exportar esta solución a otras plataformas, como está siendo el caso. Sin embargo, sí influye en la utilización de la versión actual de la plataforma Android escogida como solución de testificación digital. Esto abre una nueva línea de análisis, sobre cómo mejorar la protección de los propios elementos de seguridad embebida. Este es, sin duda, el talón de Aquiles de esta y otras soluciones dependientes de la seguridad hardware: si el núcleo de confianza resulta comprometido, afectará a todo.

VI. Conclusiones y trabajo futuro

Este documento propone una arquitectura (AUDITeD) para exportar el concepto de testigo digital a dispositivos Android. La solución emplea el *Secure Element* en el proceso de gestión de la evidencia electrónica, y permite a los propietarios del dispositivo decidir qué evidencias quieren reportar para su posterior análisis. Con el objetivo de probar la arquitectura propuesta, se ha desarrollado un prototipo publicado a través de GitHub. Este artículo sienta las bases para desarrollar sistemas más complejos en un futuro, incluso para exportar el trabajo a otros sistemas.

A fin de minimizar el efecto que las vulnerabilidades sobre los componentes hardware puedan tener en una solución, la cooperación entre varios tipos de plataformas actuando como testigos es necesaria. Por ello estamos trabajando en exportar esta solución a otras plataformas, como Raspberry Pi con TPM integrado. En este caso el esquema para la transmisión de las evidencias electrónicas sería cooperativo. La gran dependencia del hardware seguro hace necesaria la dedicación de expertos de seguridad en este área con el fin de minimizar el efecto en las soluciones desarrolladas que se apoyen en dichos mecanismos.

Agradecimientos

Este trabajo ha sido financiado por el proyecto EU EU H2020-SU-ICT-03-2018 No. 830929 CyberSec4Europe (cyber- sec4europe.eu). También por el Proyecto de Innovación Educativa PIE19-183 de la Universidad de Málaga.

Referencias

[1] A. Mylonas, V. Meletiadis, L. Mitrou, and D. Gritzalis, "Smartphone sensor data as digital evidence," *Computers & Security*, vol. 38, pp. 51–75, 2013.
[2] S. L. Garfinkel, "Digital forensics research: The next 10 years," *digital investigation*, vol. 7, pp. S64–S73, 2010.
[3] A. Nieto, R. Rios, and J. Lopez, "Iot-forensics meets privacy: towards cooperative digital investigations," *Sensors*, vol. 18, no. 2, p. 492, 2018.
[4] A. Nieto, R. Roman, and J. Lopez, "Digital witness: Safeguarding digital evidence by using secure architectures in personal devices," *IEEE Network*, vol. 30, no. 6, pp. 34–41, 2016.
[5] T. Vidas, C. Zhang, and N. Christin, "Toward a general collection methodology for android devices," *digital investigation*, vol. 8, pp. S14–S24, 2011.
[6] B. Martini, Q. Do, and K.-K. R. Choo, "Conceptual evidence collection and analysis methodology for android devices," *arXiv preprint arXiv:1506.05527*, 2015.
[7] A. M. d. L. Simao, F. C. Sicoli, L. P. de Melo, F. E. de Deus, and R. T. de Sousa Junior, "Acquisition of digital evidence in android smartphones," in *9th Australian Digital Forensics Conference*, 2011, p. 116.
[8] X. Lee, C. Yang, S. Chen, and J. Wu, "Design and implementation of forensic system in android smart phone," in *The 5th Joint Workshop on Information Security*, 2009.
[9] P. Andriotis, G. Oikonomou, and T. Tryfonas, "Forensic analysis of wireless networking evidence of android smartphones," in *WIFS 2012*. IEEE, 2012, pp. 109–114.
[10] D. Walnycky, I. Baggili, A. Marrington, J. Moore, and F. Breitinger, "Network and device forensic analysis of android social-messaging applications," *Digital Investigation*, vol. 14, pp. S77–S84, 2015.
[11] J. Sylve, A. Case, L. Marziale, and G. G. Richard, "Acquisition and analysis of volatile memory from android devices," *Digital Investigation*, vol. 8, no. 3-4, pp. 175–184, 2012.
[12] J. Richter, N. Kuntze, and C. Rudolph, "Security digital evidence," in *2010 Fifth IEEE International Workshop on Systematic Approaches to Digital Forensic Engineering*. IEEE, 2010, pp. 119–130.
[13] N. Kuntze and C. Rudolph, "Secure digital chains of evidence," in *2011 Sixth IEEE International Workshop on Systematic Approaches to Digital Forensic Engineering*. IEEE, 2011, pp. 1–8.
[14] B. Böck, D. Huemer, and A. M. Tjoa, "Towards more trustable log files for digital forensics by means of "trusted computing"," in *2010 24th IEEE International Conference on Advanced Information Networking and Applications*. IEEE, 2010, pp. 1020–1027.
[15] N. Borhan, R. Mahmod, and A. Dehghantanha, "A framework of tpm, svm and boot control for securing forensic logs," *International Journal of Computer Applications*, vol. 50, no. 13, 2012.
[16] J. Grover, "Android forensics: Automated data collection and reporting from a mobile device," *Digital Investigation*, vol. 10, pp. S12–S20, 2013.
[17] Arquitectura de la plataforma. https://developer.android.com/guide/platform/
[18] System and kernel security. https://source.android.com/security/overview/kernel-security.html
[19] Cts test for secure element. https://source.android.com/compatibility/cts/secure-element
[20] Android keystore system. https://developer.android.com/training/articles/keystore
[21] J. O'Donoghue. As easy as pie: Google's android implementation of omapi. https://globalplatform.org/as-easy-as-pie-googles-implementation-of-omapi-in-android-paves-the-way-for-simple-deployment-of-secure-apps-across-a-broad-range-of-mobile-devices/.
[22] Smart card application protocol data unit. [Online]. Available: https://en.wikipedia.org/wiki/Smart_card_application_protocol_data_unit
[23] *Open Mobile API Specification Version 3.3*, GlobalPlatform Technology.
[24] *Open Mobile API – Android Binding Version 0.0.0.5 for OMAPI v3.3*, GlobalPlatform Technology.
[25] X. Xin. Titan m makes pixel 3 our most secure phone yet. https://www.blog.google/products/pixel/titan-m-makes-pixel-3-our-most-secure-phone-yet/. Android Security Team.
[26] Cve-2019-9465. [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-9465