SEGURIDAD EN MENSAJES WAP CONTINUOUS PROVISIONING PARA REDES TETRA

J. Martínez Herrera[1], Universidad de las Ciencias Informáticas (UCI), La Habana, Cuba
Recibido Octubre 10, 2011 – Aceptado Febrero 12, 2012
http://dx.doi.org/10.18566/puente.v6n1.a06

***Resumen* — Este artículo presenta una implementación realizada para la obtención de un sistema seguro que pueda proveer a los micronavegadores de terminales TETRA de configuraciones tales como nueva página de inicio, nuevos bookmarks y valores del proxy. Esta implementación usa los conceptos principales brindados por el Laboratorio de Tecnologías de la Información del Instituto Nacional de Estándares de Estados Unidos acerca de algoritmos matemáticos y computacionales como el SHA (*Secure Hash Algorythm*). Además especifica algunas restricciones que deben ser tomadas en cuenta para el proceso de obtención de valores HMAC (*Hash Message Authentication Code*) para estos tipos de mensajes.**

***Palabras Claves* — HMAC, Microbrowsers, SHA-1, *WAP Continuous Provisioning*.**

***Abstract*— This paper presents the implementation made for a system that can provide microbrowsers located in mobile devices with information like the startpage, bookmarks and proxy configuration. This implementation uses the main concepts given by the Information Technology Laboratory of the National Institute of Standards and Technology of USA about mathematical and computational algorithms but also specify some restrictions that must be taken into consideration for the process of obtaining the HMAC (Hash Message Authentication Code) for this kind of messages.**

***Keywords*—HMAC, SHA-1, WAP, WAP Continuous Provisioning, microbrowsers.**

I. INTRODUCCIÓN

En la actualidad existe la tendencia en las redes móviles a evolucionar hacia la generalización del uso del protocolo TCP/IP. Esta cualidad viene dada por el desarrollo de las infraestructuras que brindan soporte a dichas redes. Redes como las de celulares han aumentado sus tazas de transferencia y anchos de banda a una velocidad increíble. Esto de la mano del incremento de los servicios que le son provistos al cliente final.

Debido a estas razones, el protocolo WAP (del inglés Wireles Application Protocol) ha ido perdiendo su espacio dentro de estas redes dentro de las cuales fue tan útil en el comienzo de la implantación de los servicios de datos. Sin embargo los sistemas TETRA (del inglés TErrestial Trunked Radio) han comenzado a adoptarlo como suyo debido a la mejoría de las prestaciones de las infraestructuras de redes que brindan servicios a usuarios de radio profesionales. Uno de los mayores atractivos que presenta este protocolo es la posibilidad de envío de diversos tipos de mensajes. Entre las tecnologías que permiten el envío de estos mensajes se encuentran los mensajes WAP Continuous Provisioning. Mediante el envío de estos es viable la configuración remota de algunas características de los micro navegadores que residen en los terminales y que utilizan el protocolo WAP para el envío de paquetes, razón por la cual permiten la navegación de una forma muy parecida a la realizada a través del protocolo HTTP (del inglés Hyper Text Tranfer Protocol). Esto permite que los administradores de la red puedan configurar remotamente los navegadores de los terminales sin que este dispositivo sea llevado a la fábrica o a una sucursal de la empresa, ahorrándoles tiempo y dinero a los clientes profesionales de este tipo de redes de radio digital. Sin embargo dichos mensajes pueden ser una amenaza sino se cuenta con un mecanismo que permita verificar la integridad de los datos generados por los administradores de los servicios de datos en la red. Se han reportado casos reales donde atacantes, mediante el uso de esta tecnología, han logrado enviar configuraciones erróneas y creado un caos entre los usuarios de los terminales.

Por lo mencionado anteriormente se hace imprescindible el uso de un mecanismo para la generación de un código de autentificación, que en el

[1] J. Martínez. Ingeniero en Ciencias Informáticas de la Universidad de las Ciencias Informáticas 2008. e-mail: jherrera@uci.cu.

caso que nos ocupa será un HMAC (del inglés Keyed-Hashed Message Authentification Code), que permita brindar la seguridad requerida para evitar la aceptación de información nociva para los terminales y con ello restringir los ataques que puedan aparecer dentro del sistema, ajustándose siempre a las características de la red en cuestión.

II. PROTOCOLO WAP.

WAP (del inglés Wireless Application Protocol) es un conjunto de nuevos estándares diseñados para extender los servicios de Internet al entorno de la telefonía móvil. Su desarrollo está coordinado actualmente por la OMA, aunque en un inicio fue llevado a cabo por el WAP Forum hasta que se consolidó con la OMA. Surge dada la combinación de dos tecnologías emergentes de amplio crecimiento y difusión en los últimos años: las comunicaciones inalámbricas e internet. El protocolo no solo posibilita que los dispositivos móviles puedan acceder a internet, sino que provee servicios avanzados adicionales tales como el desvío de llamadas inteligentes, entre otras. [1][2].

III. ARQUITECTURA WAP.

El protocolo se basa en la arquitectura definida para el WWW (del inglés Word Wide Web), pero adaptada a los nuevos requisitos del sistema. En la Fig. 1 se muestra el esquema de la misma. De esta forma, en el terminal móvil existiría un "micronavegador" encargado de la vinculación con la pasarela, a la cual se le realizan peticiones de información que son tratadas y redirigidas al servidor de información adecuado. Una vez procesados los datos en el servidor, se envían a la pasarela que nuevamente los procesa para enviarlos al terminal. [3].

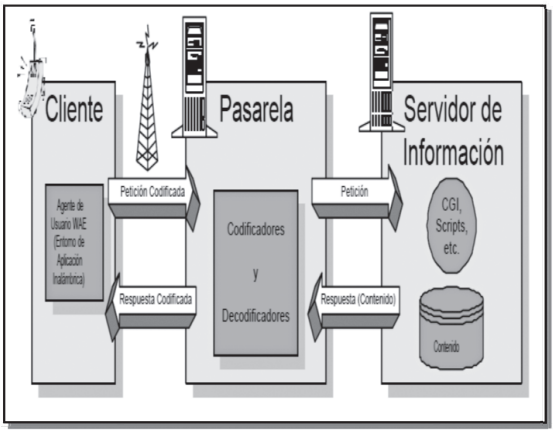

Fig. 1. Funcionamiento del Protocolo WAP. [3]

IV. ESQUEMA DE LA PILA WAP

La arquitectura WAP está pensada con el objetivo de proporcionar un "entorno escalable y extensible en el desarrollo de aplicaciones para dispositivos de comunicación móvil". Para ello, se define una estructura en capas, en la cual cada capa es accesible por la capa superior así como por otros servicios y aplicaciones a través de un conjunto de interfaces muy bien definidos y especificados. [3] Este esquema de capas es mostrado en la Fig. 2.

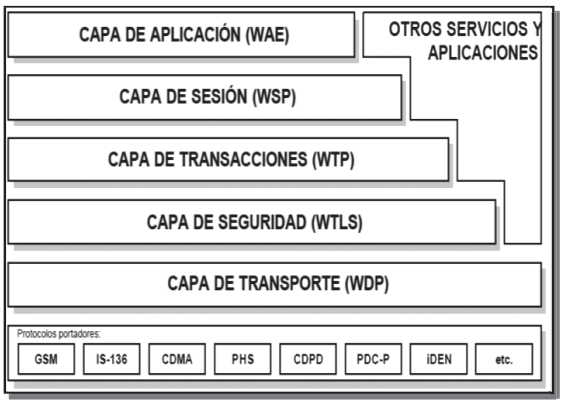

Fig. 2. Arquitectura WAP, capas presentes en la misma y posibles portadores. [3]

Dentro del sistema de capas definido en la arquitectura WAP existen cuatro de ellas muy importantes, de las cuales la capa de seguridad es opcional y las tres restantes intervienen directamente en el proceso de envío de los mensajes WAP Continuous Provisioning y que son necesarias en la conformación de la PDU (del inglés Protocol Data Unit) a enviar según las características indicadas en los documentos oficiales de la OMA. Por ello se describen a continuación dichas capas, especificando las funciones que llevan a cabo dentro de la pila WAP.

El WAE (del inglés Wireless Application Environment) está basado en las tecnologías de la WWW y optimizado para los sistemas de comunicaciones móviles. Este entorno incluye un micronavegador que posee, entre otras, las siguientes funcionalidades: [6].

- Un lenguaje denominado WML similar al HTML (del inglés Hypertext Markup Language), pero optimizado para su uso en terminales móviles.
- Ficheros de contenido WBXML (del inglés Wireless Binary eXtensive Markup Language),

los cuales son una versión hexadecimal codificada de un fichero XML basados en valores y reglas definidas por el protocolo WAP.

El WSP (del inglés Wireless Session Protocol) proporciona al WAE una interfaz con dos servicios de sesión: uno orientado a la conexión que funciona por encima del WTP (del inglés Wireless Transaction Protocol) y otro no orientado a la conexión que funciona por encima de la capa WDP (del inglés Wireless Datagram Protocol) que proporciona servicio de datagramas seguro o no seguro. Actualmente, esta capa consiste en servicios adaptados a aplicaciones basadas en la navegación Web, proporcionando las siguientes funcionalidades: [7].

- Semántica y funcionalidades del HTTP/1.1 en una codificación compacta.
- Negociación de las características del Protocolo.
- Suspensión de la Sesión y reanudación de la misma con cambio de sesión.
- En el mecanismo WAP Continuous Provisioning es la capa que establece la sesión para que se pueda enviar la información al terminal.

WDP (del inglés Wireless Datagram Protocol) proporciona un servicio fiable a los protocolos de las capas superiores de WAP y permite la comunicación de forma transparente sobre los protocolos portadores válidos. Debido a que este protocolo proporciona un interfaz común a los protocolos de las capas superiores, las capas de Seguridad, Sesión y Aplicación pueden trabajar independientemente de la red inalámbrica que dé soporte al sistema [8].

El tema de las capas resulta muy ilustrativo dentro de este artículo puesto que en epígrafes posteriores se procederá a explicar, de forma más detallada, los campos necesarios para proveer seguridad a dichos mensajes y en qué capa se incluyen. Por lo antes mencionado se decide brindar una breve información conceptual del esquema de la pila WAP para ilustrar a los lectores.

V. SERVICIO DE APROVISIONAMIENTO A DISPOSITIVOS MÓVILES.

A pesar de las prestaciones que brindan muchos de los terminales móviles en el mercado, existe un problema fundamental a la hora de actualizar software o agregar servicios y funcionalidades al usuario, haciéndose necesario que el usuario del terminal tenga que remitirse a una sucursal de la empresa que proporcionó el equipo. Este proceso se hace en ocasiones engorroso para el cliente ya que en todos los países donde se utilizan los equipos no hay sucursales de la empresa.

La optimización de este proceso se lleva a cabo mediante el uso de WAP y viene dada por el empleo de mensajes WAP Continuous Provisioning, los mismos consisten en un mecanismo que utiliza la pila de protocolos WAP, mediante el cual los proveedores ofrecen información, servicios y configuración a los dispositivos. Este mecanismo tiene como característica fundamental que ha sido diseñado para ser extensible, y permite modificar características y agregar funcionalidades a los terminales móviles.

Este procedimiento de modificar características en los terminales móviles es responsabilidad del proveedor de servicios y puede ser solicitado por el usuario o bajo interés particular del proveedor, aunque en la actualidad pocos usuarios se interesan por conocer cómo modificar parámetros en la configuración del dispositivo móvil. Cada proveedor tiene características propias, así como configuraciones específicas para que el terminal móvil pueda usar su red. En algunos casos esta se introduce al dispositivo mediante una tarjeta SIM con la información necesaria o también se puede configurar el terminal cuando se encienda, localice la red de su proveedor y este le brinde los parámetros de configuración de forma transparente al usuario. Las configuraciones pueden ser tales como proxies, puntos de acceso a la red, páginas de inicio del navegador, etc. [4].

Los servicios de aprovisionamiento donde se modifican parámetros en el terminal puede ser en dos momentos: cuando este enciende, llamándose “bootstrapping” y cuando se configura luego de estar encendido el terminal llamándose “WAP Continuous Provisioning”. La arquitectura del mecanismo se basa en la división entre la configuración mediante bootstrapping y WAP Continuous Provisioning. El bootstrapping tiene como objetivo crear la conexión inicial entre el terminal y la infraestructura del proveedor, mientras que el Continuous Provisioning modifica o agrega parámetros en el dispositivo luego que este establece una comunicación con la infraestructura del proveedor. En la Fig. 3 se muestra la arquitectura de los servicios de aprovisionamiento. [4].

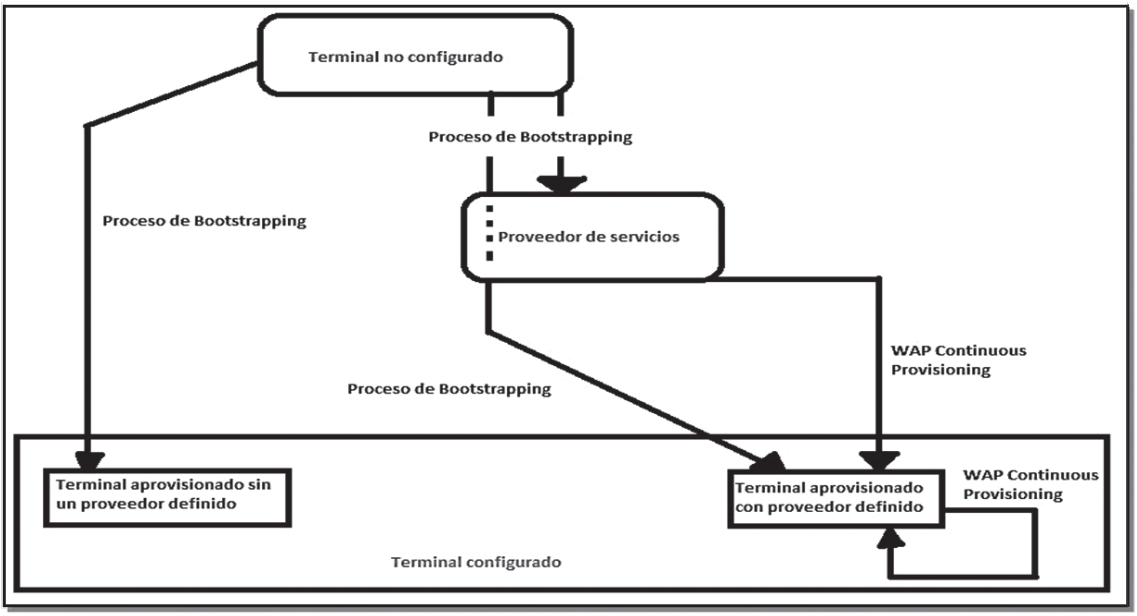

Fig. 3. Arquitectura de los Servicios de Aprovisionamiento. [4]

Dentro de las características del WAP Continuous Provisioning cabe destacar que el mismo se puede ejecutar sin importar el software del dispositivo. Además se debe señalar que este mecanismo puede ser habilitado por operaciones de cautela del proveedor, acontecimientos inteligentes de la red o petición mediante voz o datos del usuario. Un requisito para poder establecer el mecanismo es que el dispositivo debe estar encendido, de lo contrario el mismo no funcionará. La información que podría actualizarse puede ser una o varias combinaciones de las siguientes: [4].

- *Información de conectividad.*
- *Selección del proveedor de servicios.*
- *Configuración del proxy y el valor del DNS (del inglés Domain Name Server).*
- *Información para acceder a distintas aplicaciones.*
- *Configuraciones en el navegador.*

El mecanismo consiste en el envío de un WBXML, aunque también puede enviarse en Su forma de texto plano como XML (del inglés eXtensive Markup Language), el cual es interpretado en la capa de aplicación del dispositivo. En la Fig. 4 se muestra un mensaje en su formato XML con el cual se pretende modificar, en el micronavegador localizado en el terminal, la página de inicio (<parm name="STARTPAGE" value="http://192.168.10.1:8080/SerWap">), el valor del proxy lógico (<parm name="PROXY-ID" value="192.168.10.1"/>), así como la adición de una nueva bookmark mediante los campos presentes en la característica RESOURSE (<characteristic type ="RESOURCE">). Existen otros valores que son obligatorios para modificar el proxy como el APN (del inglés Network Access Point), etc. Sin embargo no es objetivo de este artículo la explicación detallada de cada uno de los campos que se pueden utilizar para el envío de configuraciones remotas a los terminales de una red. En la Fig. 5 se muestra el mismo mensaje de la Fig. 4 en su formato WBXML. Puede notarse como se reduce el volumen de la información a enviar luego de sometido el XML al proceso de conversión.

```
<?xml version=\"1.0\" encoding=\"UTF-8\"?>
<!DOCTYPE wap-provisioningdoc PUBLIC \"-
//WAPFORUM//DTD PROV 1.0//EN\"
\"http://www.wapforum.org/DTD/prov.dtd\">
<wap-provisioningdoc>
<characteristic type="BOOTSTRAP">
   <parm name="PROXY-ID" value="192.168.10.1"/>
   <parm name="NAME" value="xyz"/>
</characteristic>
<characteristic type="PXLOGICAL">
   <parm name="PROXY-ID" value="192.168.10.1"/>
   <parm name="NAME" value="Index"/>
   <parm name="STARTPAGE"
value="http://192.168.10.1:8080/index.html"/>
   <characteristic type="PXPHYSICAL">
      <parm name="PHYSICAL-PROXY-ID"
value="Gateway"/>
      <parm name="PXADDR" value="111.31.222.46"/>
      <parm name="PXADDRTYPE" value="IPV4"/>
      <parm name="TO-NAPID" value="NAP 1"/>
      <characteristic type="PORT">
         <parm name="PORTNBR" value="80"/>
         <parm name="SERVICE" value="OTA-HTTP-
PO"/>
      </characteristic>
   </characteristic>
</characteristic>
<characteristic type="NAPDEF">
   <parm name="NAPID" value="NAP 1"/>
   <parm name="BEARER" value="GSM-GPRS"/>
   <parm name="NAME" value="OUR GPRS"/>
   <parm name="NAP-ADDRESS" value="our.net"/>
   <parm name="NAP-ADDRTYPE" value="APN"/>
</characteristic>
<characteristic type="APPLICATION">
   <characteristic type ="RESOURCE">
      <parm name ="URI" value="http://www.uci.cu"/>
      <parm name ="NAME" value="UCI"/>
   </characteristic>
</characteristic>
</wap-provisioningdoc>
```

Fig. 4. Mensaje WAP Continuous Provisioning en formato XML.

```
030B6A0045C656018715060331393322E3136382E31302E31
00018707060378797A000101C651018715060331393322E313
6382E31302E3100018707060353657257617000018 71C0603
687474703A2F2F3139322E3136382E31302E313A3830383
02F5365725761700001C65201872F06034761746577617920
3100018720060331313 12E33312E3232322E3436000187210
68501872206034E415020310001C653018723060338300001
872406D201010101C65501871106034E41502031000187100
6B501870706034F555220475052530001870806036F75722E
6E6574000187090689010 1C65501C65901873A0603687474
703A2F2F7563692E6375000187070603554349000101010 1
```

Fig. 5. Mensaje WAP Continuous Provisioning en formato WBXML.

VI. ENCABEZADO WSP

Como fue explicado en el epígrafe IV, para aprovisionar con información a un terminal móvil es necesario añadirle al mensaje presentado en la Fig. 5 los valores correspondientes a los encabezados de las capas presentes en la arquitectura WAP. Debido a que la tecnología WAP Continuous Provisioning constituye un servicio no orientado a la conexión sólo se utilizan 3 capas de la mencionada arquitectura: WDP, WSP y WAE. Debido a que, de las aludidas, únicamente la capa WSP proporciona campos para la inclusión de seguridad en los mensajes (esto es de forma independiente a la seguridad que se pueda proveer con el uso de las bondades de la capa WTLS) sólo se abordarán los campos correspondientes a este encabezado.

WSP es un protocolo optimizado para llevar peticiones y respuestas HTTP (del inglés Hyper Text Transfer Protocol). Como sucede con el WBXML la principal función de WSP es la compresión de estas peticiones, respuestas y encabezados HTTP. Esta característica es importante para este desarrollo puesto que los mensajes WAP Continuous Provisioning incluyen los encabezados HTTP Content-Type y Content-Size. Adicionalmente pueden enviarse encabezados específicos de WAP, de los cuales se utilizan en este caso el destinado para WAP Push. Según la siguiente imagen el encabezado WSP para los mensajes de tipo Push toma la siguiente forma para la parte de los encabezados (headers) sin contar el identificativo de la transacción (Push ID): [7].

Para el envío de un mensaje con seguridad dentro de los encabezados mostrados en la Fig. 6 deben añadirse los siguientes campos:

- **0x91h** ------ Este valor representa que el mensaje presenta seguridad.
- **0x81h** ------ Puede tomar los valores 0x80h, 0x82h, 0x83h correspondientes a USERPIN (el usuario del terminal introduce el código manualmente), NETWPIN (el terminal buscará en una dirección determinada de la red el código de autenticación), USERNETWPIN y USERPINMAC en ese mismo orden. Es importante resaltar que el valor que tome el código o PIN (del inglés Personal Identifying Number), sin importar de qué lugar se obtenga, constituirá la llave mediante la cual las 2 partes de la comunicación generarán el valor del MAC (del inglés Message Authentication Codes).
- **0x92h** ------ Este valor representa el campo MAC.

Name	Type	Source
HeadersLen	uintvar	Length of the *ContentType* and *Headers* fields combined
ContentType	multiple octets	S-Push.req::Push Headers *or* S-ConfirmedPush.req::Push Headers *or* S-Unit-Push.req::Push Headers
Headers	(*HeadersLen* – length of *ContentType*) octets	S-Push.req::Push Headers *or* S-ConfirmedPush.req::Push Headers *or* S-Unit-Push.req::Push Headers
Data	multiple octets	S-Push.req::Push Body *or* S-ConfirmedPush.req::Push Body *or* S-Unit-Push.req::Push Body

Fig. 6. Campos para Push y Confirmed Push. [7]

VII. ESTÁNDAR SECURE HASH

Este estándar especifica cinco algoritmos seguros de truncado: SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512. Todos ellos son funciones iterativas de truncado de una sola vía que pueden procesar un mensaje para producir una representación condensada denominada mensaje digerido. Estos algoritmos propician la determinación de la integridad del mensaje: cualquier cambio realizado sobre el mismo resultará, con una muy alta probabilidad, en un mensaje digerido diferente del original. Esta propiedad es útil en la generación y verificación de firmas digitales y códigos de autenticación de mensajes, y en la generación de bits o números aleatorios. [11].

Cada algoritmo puede ser descrito en dos estados: pre-procesamiento y computación de truncado. El pre-procesamiento incluye el rellenado de un mensaje, el análisis del mensaje rellenado en bloques de m-bits y la configuración de los valores de inicialización que serán usados en la computarización de truncado. Esta última genera un mensaje programado con el mensaje rellenado y utiliza dicho mensaje programado, junto con las funciones, constantes y operaciones de palabras[1] para iterativamente generar una serie de valores truncados. El valor final generado por la computación de truncado es usado para determinar el mensaje digerido. [11].

[1]Una palabra es una cadena de w-bits que puede ser representada como una secuencia de dígitos hexadecimales. Para convertir una palabra a dígitos hexadecimales, cada cadena de 4-bits de longitud es convertida a sus dígitos hexadecimales equivalentes. Por ejemplo la cadena de 32-bit "1010 0001 0000 0011 1111 1110 0010 0011" puede ser expresada como "a103fe23".

Los cinco algoritmos divergen principalmente en la fortaleza de la seguridad que proveen para los datos que son truncados. Adicionalmente, los cinco algoritmos difieren en el tamaño de los bloques y palabras de datos que utilizan durante una operación de truncado. En la TABLA I se muestra las propiedades básicas de los algoritmos mencionados. [11].

TABLA I.

PROPIEDADES DE LOS ALGORITMOS SEGUROS DE TRUNCADO

Algoritmos	Tamaño del Mensaje (bits)	Tamaño del Bloque (bits)	Tamaño de la Palabra (bits)	Tamaño del Digest del Mensaje (bits)
SHA-1	$< 2^{64}$	512	32	160
SHA-224	$< 2^{64}$	512	32	224
SHA-254	$< 2^{64}$	512	32	256
SHA-384	$<2^{128}$	1024	64	384
SHA-512	$<2^{128}$	1024	64	512

Debido a que los mensajes de tipo WAP Provisionining serán enviados dentro de una red cuyo ancho de banda es de 28,8 Khz, se requiere que los mismos no presenten gran tamaño y se compriman lo más posible. Estas restricciones deben ser aplicadas de la misma forma a las claves generadas por lo que se seleccionó como función aprobada de truncamiento SHA-1, puesto que es la más adecuada para cumplir con los requisitos de tamaño impuestos por la red TETRA.

El sistema de envío de mensajes fue diseñado y programado en forma de API (del inglés Application Program Interface) utilizando el lenguaje Java[2] es

[2]Java es un lenguaje de programación orientado a objetos y cuenta con una gran robustez. También se puede decir que es un lenguaje de alto rendimiento ya que soporta la concurrencia a

por ello que la implementación de la seguridad debe ser hecha bajo esta misma plataforma; sin embargo la implementación del estándar SHA-1 (necesario para el cálculo del HMAC, que se explica con mayor detenimiento en el epígrafe VIII) presenta una restricción que constituye un problema para el programador. Dicha limitación consiste en que en Java no existe el tipo de dato **unsigned.** Esto resulta en una incorrecta manipulación de los bits cuando se efectúan operaciones de corrimiento (>> o <<) con ellos, puesto que ubica los valores en la parte negativa del objeto y con ello se habilita el bit de signo. La dificultad deviene en una situación complicada, por ello, el programador debe tener en cuenta en todo momento la necesaria eliminación del bit de signo y con ello su conversión al valor correcto guardado en el objeto de tipo entero. Sin embargo existe una clase contenida dentro del paquete destinado a la seguridad (java.security.*) llamada MessageDigest en la cual se realizan las operaciones correctamente de manera interna, por lo cual sólo se hace importante el conocimiento de los parámetros a utilizar para la generación de los valores truncados. Para el uso de estos beneficios se declara y utiliza de acuerdo a lo mostrado en la Fig. 7:

```
private static byte[] SHA1(byte[] sha1hash)
    throws NoSuchAlgorithmException,
UnsupportedEncodingException  {
            MessageDigest md;
              md = MessageDigest.getInstance("SHA-1");
              md.update(sha1hash, 0, sha1hash.length);
              sha1hash = md.digest();
              return sha1hash; }
```

Fig. 7. Obtención de SHA-1 mediante el uso de Message Digest en Java.

Mediante la operación Message Digest get Instance ("SHA-1"); se define la función de truncado aprobada a ser usada y se obtiene en sha1hash el resultado de dicha función.

VIII. HMAC

Proveer una vía de verificación de la integridad de la información trasmitida a través de un medio no confiable una necesidad primordial en el mundo de las comunicaciones y la computación abierta. Mecanismos que brindan dichos chequeos de integridad son llamados usualmente códigos de autenticación de mensajes o MACs. Típicamente los códigos de autenticación de mensajes son usados entre dos partes que comparten una clave secreta, de manera que puedan autenticar la información transmitida entre ambas. En el estándar define que para la obtención de un MAC tiene que utilizarse una función criptográfica de truncado (hash) en conjunto con una llave secreta. Por esta razón el mencionado mecanismo es llamado HMAC. [9] Para calcular un HMAC es necesario el uso de una función criptográfica de truncamiento aprobada. [11] HMAC además utiliza una clave secreta para el cálculo y verificación del MAC. [10].

HMAC utiliza los siguientes parámetros: [10]

- **B** ------ Tamaño del bloque (en bytes) de la entrada de la función aprobada de truncado.
- **H** ------ Una función de truncado aprobada.
- **ipad** ------ Relleno interno; específicamente el byte 0x36h repetido B veces.
- **K** ------ Clave secreta compartida entre el originador y el presutuo receptor(es).
- $\mathbf{K_0}$ ------ La clave K después de cualquier pre-procesamiento necesario para formar una clave de B bytes.
- **L** ------ Tamaño de los bloques (en bytes) de la salida de una función de truncado aprobada.
- **opad** ------ Relleno externo; específicamente el byte 0x5Ch repetido B veces.
- **texto** ------ Los datos a través de los cuales el valor del HMAC será calculado; el texto no incluye una clave rellenada. El largo del texto es de n bits, donde $0 \leq n < 2B - 8B$.
- **X** ------ 'N' Notación hexadecimal, donde cada símbolo en la cadena 'N' representa 4 bits binarios.
- || ------ Concatenación.
- $\oplus$ ------ Operación OR Exclusivo (XOR).

Para calcular un MAC a través del dato 'texto' y mediante el uso de la función HMAC se lleva a cabo la siguiente operacción [10]:

$$\mathbf{MAC\ (text) = HMAC\ (K, text) = H((K0 \oplus opad) \| H((K0 \oplus ipad) \| text))}$$

Fig. 8. Función HMAC. [10]

En las TABLA II. y Fig. 9 se muestran detalladamente cómo se lleva a cabo el proceso de construcción y obtención de cada una de las partes necesarias para la obtención del HMAC.

través de hilos (threads). Es altamente portable pues los programas desarrollados con él pueden ejecutarse en cualquier tipo de hardware o sistema operativo con la única restricción de que tengan instalados una instancia de la máquina virtual de Java. Contiene un gran número de librerías que constituyen extras y de esta forma mantiene las facilidades básicas del lenguaje en un mínimo y le agrega gran cantidad de funcionalidades necesarias para el desarrollo de software, también evita la posibilidad de manipular la memoria lo cual le confiere una mayor seguridad al no permitir el acceso a los recursos del sistema. [12]

TABLA II.
DESCRIPCIÓN DETALLADA DE LAS OPERACIONES DE LA FUNCIÓN

Pasos	Descripción de cada paso
1	Si la longitud de K = B entonces K_0 = K. Ir al paso 4.
2	Si la longitud de K > B: truncar K para obtener una cadena de L bytes, luego añadir (B-L) ceros para crear una cadena de B bytes K_0 (ej., K_0 = H(K) || 00...00). Ir al paso 4.
3	Si la longitud de K < B: añadir ceros al final de K para crear una cadena de B bytes K_0 (ej., si K tiene 20 bytes de longitud y B = 64, entonces K tendrá 44 bytes de valor cero 0x00h).
4	Aplicar OR Exclusivo a K_0 con ipad para producir una cadena de B bytes: **$K_0 \oplus$ ipad.**
5	Añadir el flujo de datos de 'texto'a la cadena resultante del paso 4: **($K_0 \oplus$ ipad) || text.**
6	Aplicar H al flujo generado en el paso 5: **H(($K_0 \oplus$ ipad) || text).**
7	Aplicar OR Exclusivo a K_0 con opad: **$K_0 \oplus$ opad.**
8	Añadir el resultado del paso 6 al del paso 7: **($K_0 \oplus$ opad) || H(($K_0 \oplus$ ipad) || text).**
9	Aplicar H al resultado del paso 8: **H((K0 $\oplus$ opad)|| H((K0 $\oplus$ ipad) || text)).**

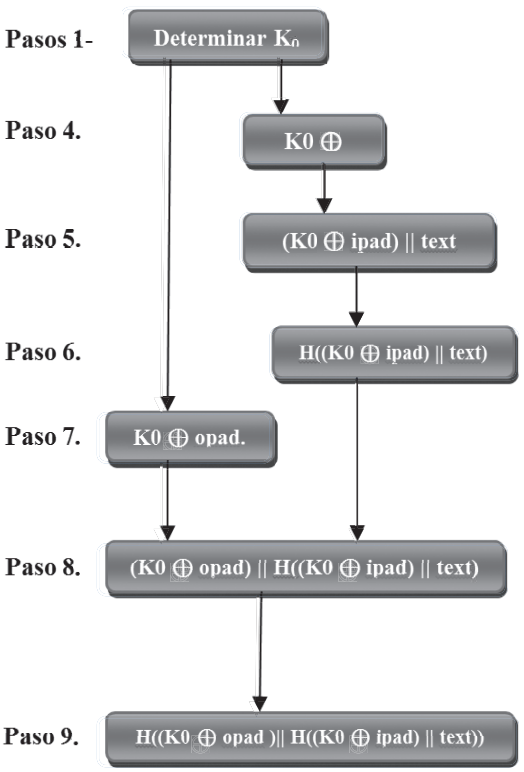

Figura 9. Construcción del HMAC. [10]

Cada uno de los pasos anteriores fueron recogidos en una clase que se vincula con el sistema de envío de mensajes para ser utilizado. Dicha clase se muestra en la Fig. 10.

```
HMAC

-ExclusiveOr(byte[] x, byte[] y): byte []
-Concat(byte[] x, byte[] y): byte []
-SHA1(byte[] sha1hash): byte []
+ComputeHash(byte[] key, byte[] wbxml): byte []
```

Fig. 10. Clase que permite calcular un valor HMAC.

IX. CASO DE ESTUDIO.

En esta Sección se muestra un caso de estudio, donde un mensaje de tipo WAP Continuous Provisioning es codificado a su formato WXBML y posteriormente, mediante el uso de una clave PIN se genera un HMAC para dicho mensaje.

Se desea enviar un mensaje Provisioning que permita cambiar, en el navegador del terminal, los valores de la página de inicio, el proxy y un bookmark. Para este propósito se utilizó el XML correspondiente a la Fig. 4. El HMAC generado mediante el uso de las clases correspondientes puede verse en la Fig. 11.

```
34373534413944333445433932433933323633443939393146
38394645324435423232343534443O
```

Fig. 11. HMAC generado para el ejemplo mostrado en la Fig. 13.

Como puede apreciarse en el ejemplo el HMAC consta de una longitud de 40 caracteres, esto si se interpreta como bytes se estaría en presencia de una cadena de 20 dígitos hexadecimales, que multiplicados por 8 (cada bytes está compuesto por 8 bits) se tendrían 160 bits, correspondiendo así con la las características dadas en la TABLA 1 para el algoritmo SHA-1.

Si se observa la Fig. 12 se podrá notar dicho HMAC contenido dentro del encabezado WSP mostrado en color rojo.

870B8423F00113**0106311F2DB691819234373534413944333444543393243393332363344393939314638394645324435423232343534434430008 1EA**030B6A0045C656018715060331
39322E3136382E31302E3100018707060378797A000101C6
5101871506033139322E3136382E31302E31000187070603

Fig. 12. Fragmento 1 de mensaje WAP obtenido una vez codificado el ejemplo mostrado en la Fig. 4.

870B8423F00123496E6465780001871C0603687474
703A2F2F3139322E3136382E31302E313A38303830
2F696E6465782E68746D6C0001C65201872F060347
617465776179000187200603313131 2E33312E32323
22E34360001872106850187220603 4E415020310001
C65301872306033830000187240 6

Fig.13. Fragmento 2 de mensaje WAP obtenido una vez codificado el ejemplo mostrado en la Fig. 4.

870B8423F00133D201010101C65501871106034E41502031
0001871006B501870806036F75722E6E65740001870906890
101C655010101

Fig.14. Fragmento 3 de mensaje WAP obtenido una vez codificado el ejemplo mostrado en la Fig. 4.

El encabezado WDP también está recogido en los tres ejemplos mostrados en las Fig. 12, 13 y 14, y puede diferenciarse del resto en que los valores están en color azul. Los bytes restantes y claramente identificables en color negro corresponden al WBXML del mensaje con los datos a configurar en el micronavegador. Los valores representados anteriormente han sido generados por las clases y APIs referenciadas en el presente trabajo.

X. CONCLUSIONES.

En este trabajo se ha presentado un estudio sobre la inclusión de seguridad en el envío de mensajes WAP Continuos Provisioning dentro de las redes TETRA.

Las clases y APIs mencionadas en el cuerpo del artículo muestran que los resultados son satisfactorios una vez incluida la seguridad en los campos correspondientes de los encabezados. Además, dichas clases fueron desarrolladas de manera en que puedan ser acopladas a cualquier desarrollo que utilice estos tipos de mensajes, con el único requerimiento de que los sistemas que pretendan utilizarla hayan sido desarrollados en Java. La ventaja principal que representa este desarrollo radica en que, a pesar de la seguridad presente dentro de las redes TETRA, se le añade seguridad a los mensajes por lo que reducen los riesgos de que se introduzcan configuraciones nocivas para el sistema de datos.

REFERENCIAS

[1] Open Mobile Alliance, "WAP 2.0 Technical White Paper.", disponible en: http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html

[2] Open Mobile Alliance, "WAP Architecture.", disponible en: http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html

[3] Open Mobile Alliance, "Binary XML Content Format Specification", disponible en: http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html

[4] Open Mobile Alliance, "Provisioning Architecture Overview.", disponible en: http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html

[5] Open Mobile Alliance, "Provisioning Content.", disponible en: http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html

[6] Open Mobile Alliance, "Wireless Application Environment Specification.", disponible en: http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html

[7] Open Mobile Alliance, "Wireless Session Protocol Specification.", disponible en : http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html

[8] Open Mobile Alliance, "Wireless Datagram Protocol.", disponible en : http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html

[9] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", Internet Engineering Task Force, Request for Comments (RFC) 2104, February 1997.

[10] Federal Information Processing Standards Publication, "The Keyed-Hash Message Authentication Code (HMAC)", Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, MD 20899-8900, 2008.

[11] Federal Information Processing Standards Publication, "Secure Hash Standard (SHS)", Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, MD 20899-8900, 2008.

[12] Centro de Tecnología Informática Universidad de Navarra, "Java", Centro de Tecnología Informática Universidad de Navarra, disponible en: http://www.unav.es/cti/manuales/Java/indice.html#1.1

BIOGRAFÍA

Joan Martínez Herrera recibió el grado de Ingeniero en Ciencias Informáticas de la Universidad de las Ciencias Informáticas (UCI), La Habana, Cuba y es el programador principal, asesor del sistema de mensajería de WAP y analista principal en el proyecto SERWAP (Servidor de Aplicaciones WAP) en TLM (Centro Telemática) dentro de la UCI. Sus intereses de investigación actuales son los servicios WAP, aplicaciones para redes TETRA y dispositivos móviles.