METIS

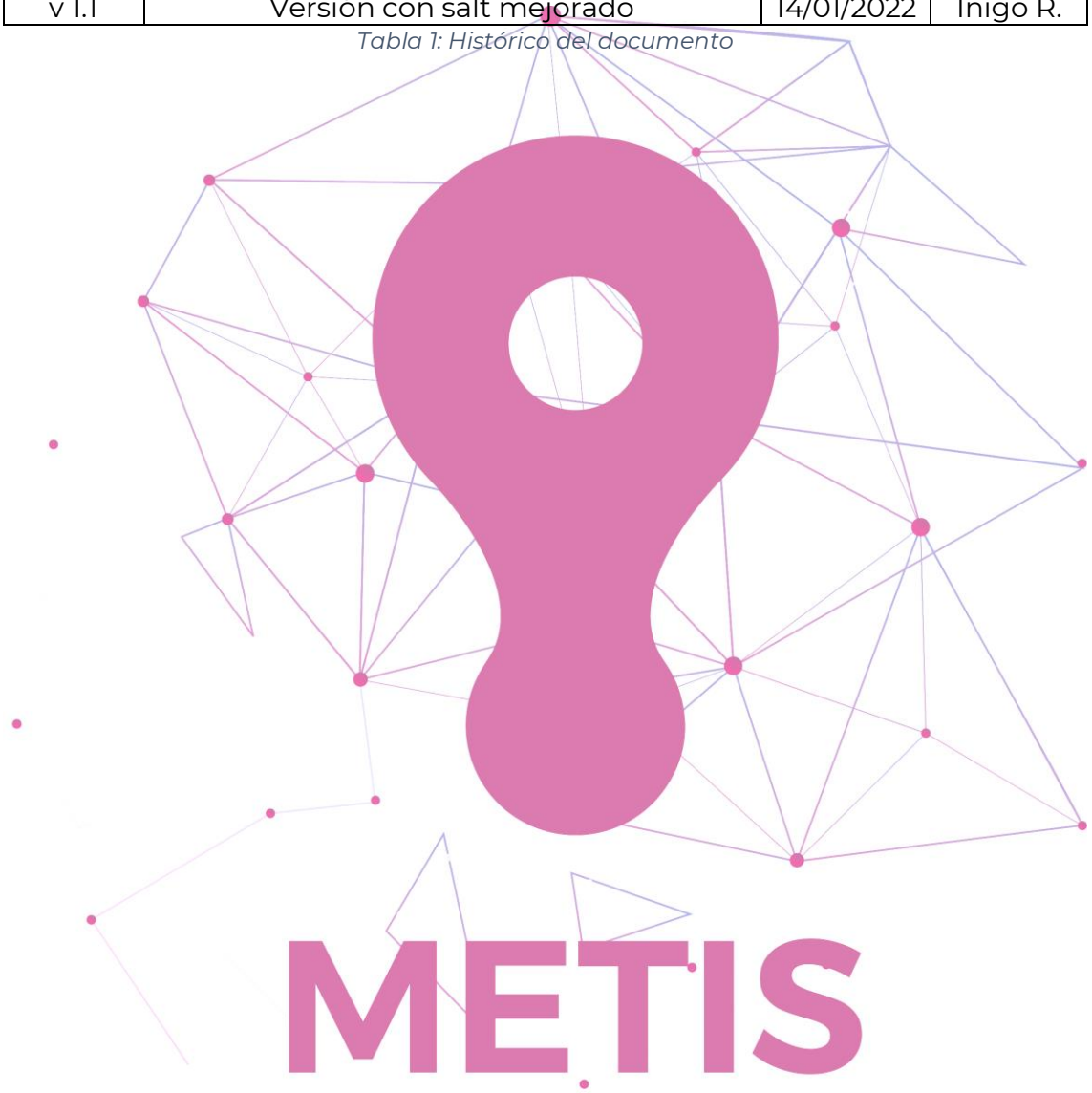
Biblioteca de anonimización de direcciones MAC

v 1.1

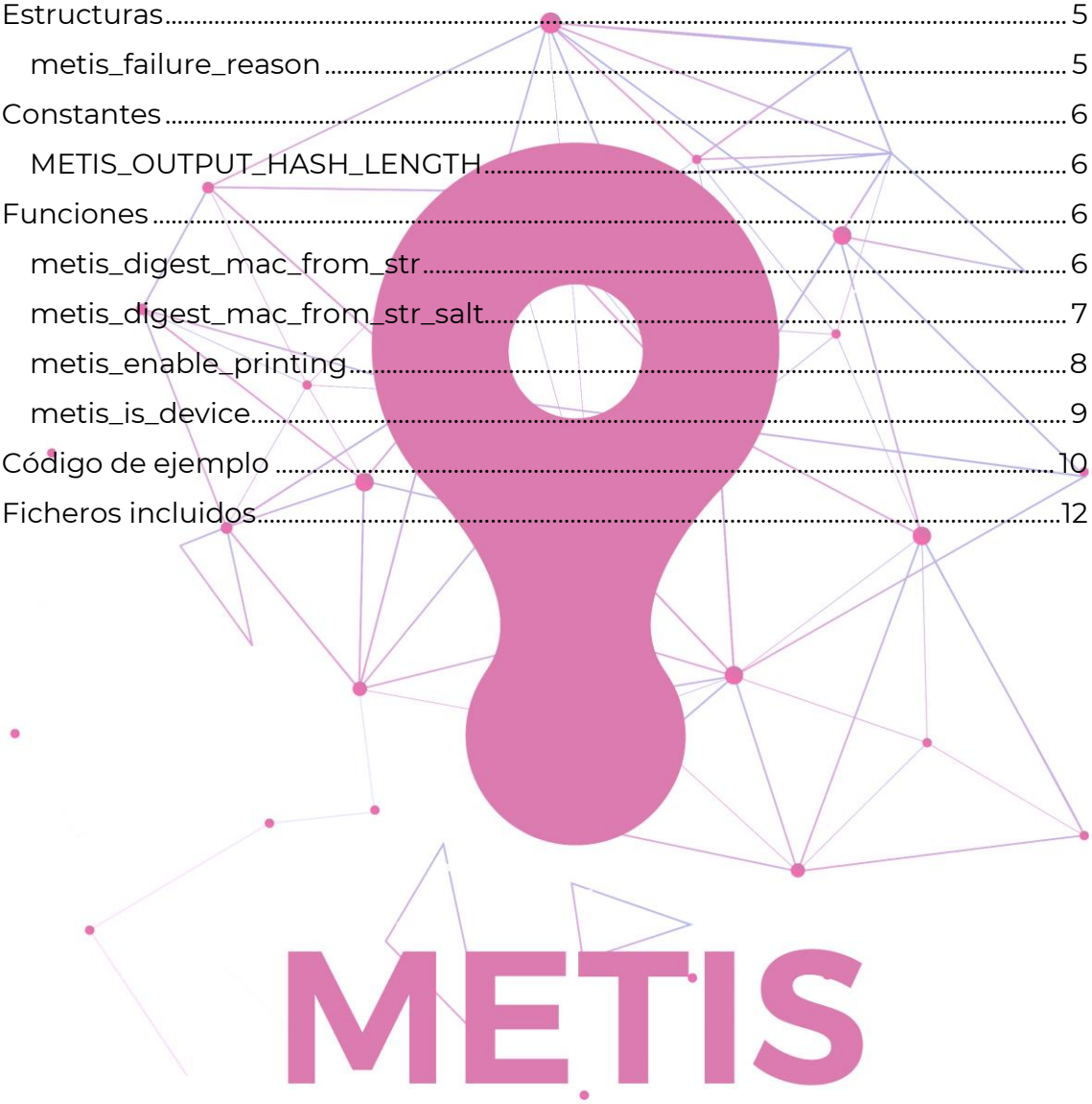
Historial

Versión	Descripción	Fecha	Autor
v 1.0	Versión Inicial.	15/7/2019	Iñigo R.
v 1.1	Versión con salt mejorado	14/01/2022	Iñigo R.

Tabla 1: Histórico del documento



Índice



Introducción

Todos los dispositivos WiFi disponen una dirección, en principio única, que los identifica y permite el control y reparto de los recursos dentro del canal de comunicación en el que estén operando.

Esta dirección, denominada dirección MAC por sus siglas en Inglés, Medium Access Control, se transmite en claro, es decir, sin cifrar en todos los paquetes WiFi que el dispositivo transmita. Esta dirección suele fijarse de fábrica en el dispositivo, y es fija.

La utilidad de esta dirección se suele asemejar con la matrícula de un coche, donde el identificador, claramente visible permite el control del tráfico circulatorio y de por sí, no desvela más información personal sobre la persona propietaria del coche.

Sin embargo, y a la luz de los nuevos usos que se les da a los smartphones, varias instituciones y gobiernos están incluyendo este tipo de direcciones como parte del conjunto de datos que se denominan Datos susceptibles de ser personalmente identificados (PII por sus siglas en inglés). Este hecho está recogido al menos, en los documentos del Reglamento General de Protección de Datos, de la Unión Europea y la Comisión Federal de Comercio, de los Estados Unidos.

Es por esto, que el producto Metis, de Purple Blob S.L. incorpora una serie de mecanismos de suma, muestreo y recombinado de los datos, además de utilizar una función irreversible de hash de 160 bits para proveer de un identificador anonimizado que pueda servir para la realización de estadísticas y conclusiones, a la vez que respeta la privacidad de las personas.

Esta biblioteca, de la que es objeto el presente documento, ofrece una interfaz de programación (API), sobre la cual, otras empresas pueden desarrollar productos que se beneficien de la investigación y desarrollo de Purple Blob en la generación de estos identificadores anonimizados. A su vez, estos productos de terceros quedarán habilitados para la interoperabilidad con los productos METIS de Purple Blob, como pueden ser sensores desplegados en otras calles o ciudades, la plataforma de análisis y reportes Sara, u otros productos que Purple Blob desarrolle basándose en el mismo formato de indicadores.

API C

Para dotar de la mayor flexibilidad posible de cara a posibles implementaciones de terceros que hagan uso de la biblioteca objeto de este documento. La API se ha desarrollado 100% compatible con el lenguaje de programación C.

Esto significa que los desarrollos de terceros que se encuentren desarrollados en este popular lenguaje orientado hoy en día a dispositivos IoT, embebidos y/o de bajo consumo, pueden beneficiarse directamente las funcionalidades ofrecidas.

Los proyectos que utilicen otros lenguajes de programación podrán encontrar sin duda, adaptaciones o métodos para llamar a las funciones expuestas sin problema alguno, dado que C es uno de los lenguajes más universales.

Para facilitar la tarea de adaptar dicha API a otros lenguajes fuera del conjunto C/C++, se ha provisto de una cabecera de SWIG (Fichero metis_algolib.i), que permite la generación, mayormente automatizada de envolturas para otros lenguajes de programación como Java, Python, Perl, Javascript y PHP entre otros muchos.

Es importante resaltar que las funciones que toman como entrada un const char* esperan una cadena de caracteres terminada por un carácter nulo.

Además, al final del presente documento, se ofrece código de ejemplo en C para operar con la biblioteca.

Estructuras

metis_failure_reason

Sintaxis:

```
typedef enum metis_failure_reason {
    metis_failure_reason_none,
    metis_failure_reason_non_utf8_string_input,
    metis_failure_reason_input_format_error,
    metis_failure_reason_hex_range_error,
} metis_failure_reason;
```

Descripción:

Este tipo de dato, es el tipo de dato de retorno en casi todas las funciones de la biblioteca, indicando si la operación se ha realizado sin error, o en caso contrario, el tipo de error ocurrido.

- **metis_failure_reason_none**: Describe el resultado de una función que se ha llevado a cabo sin problemas y los resultados se pueden comprobar de la manera esperada.

- **metis_failure_reason_non_utf8_string_input**: Se ha detectado un error en un parámetro de entrada de tipo cadena de texto, donde se han encontrado un carácter que no corresponde al conjunto de caracteres UTF-8.
- **metis_failure_reason_input_format_error**: Se ha detectado un error en un parámetro de entrada de tipo cadena de texto, donde se ha detectado que no se ha seguido el formato indicado de entrada. Esto ocurre por ejemplo cuando no se respeta el formato hexadecimal con puntos con el que se espera recibir la dirección MAC en las funciones que lo necesitan.
- **metis_failure_reason_hex_range_error**: Se ha detectado un error en un parámetro de entrada de tipo cadena de texto, donde se han encontrado un carácter que no corresponde al conjunto de caracteres hexadecimal. Esto ocurre, por ejemplo, cuando alguno de los caracteres de entrada a una función que toma una MAC como cadena de caracteres contiene un carácter que no se encuentra en el rango 0-F, definido en el espacio de caracteres hexadecimal.

Constantes

METIS_OUTPUT_HASH_LENGTH

Sintaxis:

```
const uint8_t METIS_OUTPUT_HASH_LENGTH;
```

Descripción:
Esta constante define la longitud de salida de las funciones que necesitan que se reserve un buffer de salida, cuya mínima capacidad sea de esta longitud. Esta cifra ya tiene en cuenta el último carácter nulo obligatorio en algunos lenguajes.

Funciones

metis_digest_mac_from_str

Sintaxis:

```
metis_failure_reason
metis_digest_mac_from_str(const char *mac_str, char *out_buf);
```

Descripción:

Anonimizar una dirección MAC y convertirla en un hash de la longitud indicada por METIS_OUTPUT_HASH_LENGTH.

Parámetros:

Parámetro	Tipo	Entrada / Salida	Descripción
mac_str	**const char** *	Entrada	Puntero a cadena de texto que contenga una dirección MAC en modo texto en formato “AA:BB:CC:DD:EE:FF”
out_buf	**char** *	Salida	Puntero a cadena de texto de longitud mínima METIS_OUTPUT_HASH_LENGTH en la que se almacenará el hash resultante.

Valores de retorno:

Enum de tipo metis_failure_reason, de acuerdo a los valores descritos en la sección 3.1.

metis_digest_mac_from_str_salt

Sintaxis:

```
metis_failure_reason
metis_digest_mac_from_str_salt(
        const char *mac_str,
        const char *salt,
        char *out_buf
);
```

Descripción:

Anonimizar una dirección MAC y convertirla en un hash de la longitud indicada por METIS_OUTPUT_HASH_LENGTH. A esta función se le indica, además un número, que se combinará con la MAC en el proceso de anonimización para generar un identificador nuevo, permitiendo rotar los identificadores anonimizados cuando sea requerido.

Parámetros:

Parámetro	Tipo	Entrada / Salida	Descripción
mac_str	**const char** *	Entrada	Puntero a cadena de texto que contenga una dirección MAC en modo texto en formato “AA:BB:CC:DD:EE:FF”

salt	**const char** *	Entrada	Puntero a cadena de texto que contenga 8 caracteres hexadecimales que se sumarán al proceso de anonimización.
out_buf	**char** *	Salida	Puntero a cadena de texto de longitud mínima METIS_OUTPUT_HASH_LENGTH en la que se almacenará el hash resultante.

Valores de retorno:

Enum de tipo metis_failure_reason, de acuerdo a los valores descritos en la sección 3.1.

metis_enable_printing

Sintaxis:

```
void metis_enable_printing(bool enabled);
```

Descripción:

Activar o desactivar la salida de texto directamente desde la biblioteca hacia los canales de salida estándar (STDOUT y STDERR). Esto está desactivado por defecto y permite, en caso de habilitarse, recabar más información sobre el funcionamiento interno de la biblioteca y comprobar que se están recibiendo bien los parámetros de entrada y salida.

La función tiene efecto global, lo cual significa que afecta a todo el espacio de aplicación, independientemente del hilo de ejecución desde el que se invoque.

Es, por tanto, una función sincronizada, que sólo se puede ejecutar una vez en paralelo.

NOTA: Esta función no tendrá efecto alguno en la librería compilada para procesadores Xtensa LX6.

Parámetros:

Parámetro	Tipo	Entrada / Salida	Descripción
enabled	**bool**	Entrada	Un valor verdadero habilita las funciones de salida

			de texto. Un valor falso, lo deshabilita.

Valores de retorno:

No tiene.

metis_is_device

Sintaxis:

```
metis_failure_reason
metis_is_device(const char *mac_str, bool *out);
```

Descripción:

Determinar si una dirección MAC corresponde de manera probable a un dispositivo móvil.

Esta función realiza una consulta a un listado interno de la biblioteca para determinar si, por medio del fabricante indicado en la dirección MAC, el dispositivo por el que se ha consultado es susceptible de ser un Smartphone, Tablet, u otro dispositivo personal con funcionalidad WiFi.

Parámetros:

Parámetro	Tipo	Entrada / Salida	Descripción
mac_str	**const char** *	Entrada	Puntero a cadena de texto que contenga una dirección MAC en modo texto en formato “AA:BB:CC:DD:EE:FF”
out	**bool** *	Salida	Puntero a valor booleano en el que se almacena el resultado de la consulta.

			Un valor verdadero indica que el dispositivo consultado es probablemente un dispositivo móvil personal. Un valor falso indica lo contrario.

Valores de retorno:

Enum de tipo metis_failure_reason, de acuerdo a los valores descritos en la sección 3.1.

Código de ejemplo

A continuación, se ha incluido un programa de ejemplo en C, en el que se demuestra la forma de interactuar con las diferentes funcionalidades descritas en las secciones anteriores.

```
#include <stdio.h>
#include "metis_algolib.h"

int main( int argc, const char* argv[] )
{
    const char* exampleMac = "76:4a:7f:9b:da:57";
    const char* exampleMac2 = "84:89:ad:29:f8:5f";
    const char* exampleMac3 = "8Z:89:ad:29:f8:5Z";
    const char* exampleMac4 = "8Z:89:ad:2ŋ:f8:5";
    char out_mac[METIS_OUTPUT_HASH_LENGTH];
    bool isDevice = false;

    metis_enable_printing(true);

    printf("TESTING MAC: %s\n", exampleMac);
    if (metis_digest_mac_from_str(exampleMac, (char*)&out_mac) ==
metis_failure_reason_none ) {
        printf("OK\n");
        printf("Resulted MAC: %s\n", out_mac);
    } else {
        printf("FAILED\n");
```

```
    }

    printf("TESTING MAC: %s\n", exampleMac2);
    if (metis_is_device(exampleMac2, &isDevice) == metis_failure_reason_none ) {
        printf("OK\n");
        printf("Is Device?: %s\n", isDevice ? "YES" : "NO");
    } else {
        printf("FAILED\n");
    }

    printf("TESTING MAC: %s\n", exampleMac3);
    if (metis_is_device(exampleMac3, &isDevice) == metis_failure_reason_none ) {
        printf("OK\n");
        printf("Is Device?: %s\n", isDevice ? "YES" : "NO");
    } else {
        printf("FAILED\n");
    }

    printf("TESTING MAC: %s\n", exampleMac4);
    if (metis_digest_mac_from_str(exampleMac4, (char*)&out_mac) ==
metis_failure_reason_none ) {
        printf("OK\n");
        printf("Resulted MAC: %s\n", out_mac);
    } else {
        printf("FAILED\n");
    }
}
```

La salida del programa es la siguiente.

```
TESTING MAC: 76:4a:7f:9b:da:57
[METIS] Input MAC: [76:4a:7f:9b:da:57]
[METIS] Final Hash: 5e373
OK
Resulted MAC: 5e373
TESTING MAC: 84:89:ad:29:f8:5f
[METIS] MAC OUI: 8685997
[METIS] MAC 84:89:ad:29:f8:5f is a mobile device
OK
Is Device?: YES
TESTING MAC: 8Z:89:ad:29:f8:5Z
[METIS] Error parsing HEX string input. Make sure all chars are valid!
FAILED
TESTING MAC: 8Z:89:ad:2ŋ:f8:5
```

```
[METIS] Input MAC: [8Z:89:ad:2ŋ:f8:5]
[METIS] Mac format check failed: Expecting 17 char length string
FAILED
```

Ficheros incluidos

A continuación, se describen los diferentes ficheros que forman parte del proyecto entregado a los clientes de la biblioteca y sus diferentes propósitos.

- **libmetis_algolib.h**: Fichero de cabecera de C, en el que se declaran las funciones ofrecidas por la biblioteca y que es necesario para la utilización de esta de manera directa o mediante la envoltura para algún otro lenguaje de programación.
- **libmetis_algolib_lx6.h**: Fichero de cabecera de C, para el caso de las implementaciones basadas en procesadores Xtensa LX6.
- **libmetis_algolib.i**: Fichero de definiciones de SWIG que facilita el desarrollo de envolturas para la utilización de esta biblioteca en otros muchos lenguajes.
- **libmetis_algolib_armv7hf.a**: Compilación estática de la librería, sin dependencias, para arquitecturas de CPU basadas en ARMv7, con procesador de números flotantes por hardware.
- **libmetis_algolib_i686.a**: Compilación estática de la librería, sin dependencias, para arquitecturas de CPU basadas en Intel de 32 bits.
- **libmetis_algolib_x64-64.a**: Compilación estática de la librería, sin dependencias, para arquitecturas de CPU basadas en Intel de 64 bits.
- **libmetis_algolib_mips.a**: Compilación estática de la librería, sin dependencias, para arquitecturas de CPU basadas en MIPS.
- **libmetis_algolib_lx6.a**: Compilación dinámica de la librería, con dependencias con MbedTLS.
- libmetis_doc.pdf: Este fichero de documentación.

Todas las librerías han sido compiladas de forma estática, para favorecer su integración y no dependen de ninguna otra librería o software, dado que han sido compiladas con la librería estándar integrada MUSL. Todas las librerías son compatibles con el núcleo de Linux.