```
Título:     ==OpenTimeStamps==
Fecha de creación: 12-06-2022 (mm/dd/aa)
Hora de creación: 18h20/utc-04
tags: ['#bitcoin','#timestamp']
aliases: [Proof of Existence]
```

opentimestamps

En el proyecto Bitcoin/Nodo Bitcoin hemos usado este protocolo para verificar la inmutabilidad de archivos y binarios. La preferencia sobre las otras soluciones para un Timestamp Trustly es porque tiene estas dos particularidades:

1. Es gratis. No necesita pagar el fee de una transacción Bitcoin Esto pq usa Merkle Trees para agregar muchos hashes a una sola transacción.
2. Ha sido investigado en un contexto de análisis forense digital https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8732798 con una conclusión destacable: el servicio es extremadamente confiable evitando falsos positivos y falsos negativos siempre y cuando la firma de tiempo no requiera alta sensibilidad temporal (un timestamp sin segundos delay)
3. Es instantáneo.

Caracteristicas

1. Medida de Integridad.
   Se puede usar como medida de verificación que un documento no ha sido manipulado ni modificado.
2. Firma PGP.
   Las firmas PGP tienen una fecha de expiración. Si esta

es válida antes de generar un timestamp y existe una marca de tiempo válida es un caso de uso muy útil.

3. Evidencia de Autenticidad.
   Si aún solo hay un archivo para firmar en una transacción Bitcoin este dispone de 80 bytes para guardar un mensaje. Por esta limitante del tamaño se guarda solamente un Hash. Mucho mas si se usan Merkle Tree. Por esta razón este timestamp muestra que un mensaje es auténtico pero no muestra el mensaje en sí.

Tutorial de Uso

Para hacer una verificación se parte de un fichero `.ots`. Con este fichero se puede verificar la integridad de un archivo/binario sin delegar la consulta a un tercero.

Para instalar se sigue el comando

```
sudo pip3 install opentimestamps-client
```

Creando un sello de tiempo

Primero creamos un archivo prueba con un texto plano.

```
touch hello.txt && echo 'Hola mundo' > hello.txt
cat hello.txt
> hola mundo
```

Para crear una marca de tiempo

```
ots stamp hello.txt
>Submitting to remote calendar >https://a.pool.opentimestamps.org
>Submitting to remote calendar >https://b.pool.opentimestamps.org
>Submitting to remote calendar >https://a.pool.eternitywall.com
>Submitting to remote calendar >https://ots.btc.catallaxy.com
```

Generando el archivo `hello.txt.ots`

La operación *stamp* calcula el hash SHA256 "Secure Hash Algorithm" del fichero original, le concatena un *nonce* aleatorio de 128 bits para mantener la privacidad, y vuelve a calcular el hash SHA256, enviando este único valor a los servidores de calendario. Cada uno de los servidores de calendario añadirá el hash recibido a su árbol de Merkle y devolverá la respuesta necesaria para generar el fichero OTS inicial.

Este fichero OTS está aún incompleto porque no contiene el registro en la cadena de bloques.

Se puede verificar esto último con el comando

```
ots upgrade hello.txt.ots
>Calendar https://alice.btc.calendar.opentimestamps.org: >Pending confirmat
>Calendar https://bob.btc.calendar.opentimestamps.org: >Pending confirmatio
>Calendar https://finney.calendar.eternitywall.com: >Pending confirmation i
>Failed! Timestamp not complete
```

Pasado un tiempo se genera la firma en Bitcoin y estará completado.

```
ots upgrade hello.txt.ots
>Got 3 attestation(s) from cache
>Success! Timestamp complete
```

Verificando un fichero

Para verificar se necesitan dos archivos

1. El fichero.
2. El fichero ots.
   Y para no depender de un tercero de confianza tener un

Bitcoin/Nodo Bitcoin para hacerle consultas sobre el blockchain directamente. Aunque finalmente puede realizarse en algún servicio online.

Con el siguiente comando se verifica una prueba OTS

```
ots verify hello.txt.otx -f hello.txt
> Success! Bitcoin block 766199 attests existence as of 2022-12-06 -04
```

Información Archivo

Se puede consultar la información como el hash y transacción donde se realiza el timestamp

```
ots info hello.txt.ots

File sha256 hash: 41d85e0b52944ee2917adfd73a2b7ce3d3c8368533a75e54db881fac6
Timestamp:
append 5b29fbe0b085e22e26d19e239b03c419
sha256
 -> append 0e01db9eeb4f1d7975c5e53f5a0d3974
    sha256
...
```