XAILOUR
Community as a service

Verificatore notarizzazione

Notarizzazione e verifica documenti

EggNet Lab srls unipersonale
Via Villa 49/G, 40017 San Giovanni in Persiceto (BO)
P.Iva/C.f. 04112971207

info@xailour.net
www.xailour.net

Notarizzazione documenti

La notarizzazione dei documenti è un processo che permette di salvare su blockchain dei metadati che identificano univocamente un qualsiasi documento digitale.
Tali metadati sono identificanti e forniscono la prova di esistenza del documento ad una data certa, corrispondente al timestamp del blocco ove sono notarizzati i metadati in questione.
Per ogni documento notarizzato, vengono gestiti i seguenti metadati identificanti:

- hash del documento calcolato con un algoritmo noto a priori
- dimensione del documento
- mime type del documento

Modulo XAIBusiness

Dall'applicazione XAIBusiness è possibile gestire i propri documenti e notarizzarli tramite l'apposita voce di menù "Documenti".
Da quella pagina è possibile caricare i propri documento in formato ".pdf", ricevendo in risposta l'id della ricevuta per ogni documento notarizzato.

Modulo XAINotary

Il modulo XAINotary fornisce la funzionalità di notarizzazione documenti tramite le proprie API REST.
Sono disponibili tre endpoint:

- notarizzazione di file: restituisce l'id della ricevuta
- recupero della ricevuta: recupera la ricevuta
- verifica di autenticità: verifica la coerenza della ricevuta col documento notarizzato

Ricevuta

La ricevuta è un documento, rappresentato in formato JSON, che contiene tutte le informazioni relative alla notarizzazione del documento fornito in input al processo.

La ricevuta è composta da tre sezioni:

- DOCUMENT: metadati del documento notarizzato
- TRANS_PRV: informazioni sulla transazione effettuata nella blockchain privata di XAINet
- TRANS_PBL: informazioni sulla transazione effettuata nella blockchain pubblica di Ethereum

La ricevuta va conservata da parte dell'utente (o del sistema integrato) come prova di esistenza del documento e va fornita, insieme al documento stesso, agli strumenti di validazione per attestarne la veridicità.

DOCUMENT

Questa sezione contiene l'hash dei metadati identificanti del documento.
Questo è un esempio di metadati:


```
{
  "size": 4736951,
  "mtype": "application/pdf",
  "hash":
"4fe674416f2c50724a79bbad486f22daa8d3948d4f640a6fe6afcaa78d405f122f1561ab396a8e25ba4
aa766b6be3b4c1f4acb3c5719131b40e18ff558e4b910",
  "alg": "sha512"
}
```


dove:

- size: rappresenta la dimensione in byte del documento
- mtype. rappresenta la tipologia del documento
- hash: contiene l'hash del documento
- alg: algoritmo utilizzato per calcolare l'hash

TRANS_PRIV

Questa sezione contiene i dati relativi alla notarizzazione effettuata sulla blockchain di XAINet.
Ogni documento viene notarizzato con una transazione dedicata, creando quindi un rapporto 1 a 1 tra documento e transazione.


```
{
  "hash": "0x61670fdd2b8a05246ff071df295f8635dfaea1b5bda00385ebaf23be71891f4f",
  "payload":
"0x7b2273697a65223a2034373363935312c20226d74797065223a20226170706c69636174696f6e2
f706466222c202268617368223a2022346665363734343136663263353037323461373962626164343
836663232646161386433393438643466363430613666653661666361613738643430356631323266
631353631616233393661386532356261346161373636623662653362346331663461636233633353
```

```
731393133316234306531386666353538653462393130222c2022616c67223a202273686135313222
7d",
  "timestamp": 1611914939
}
```


Il campo hash contiene l'id della transazione.

Il campo payload contiene i metadati relativi al documento.

Il campo timestamp indica il momento in cui la transazione è stata accettata in un blocco della catena.

TRANS_PBL

Questa sezione contiene i dati relativi alla notarizzazione effettuata sulla rete pubblica di Ethereum.

La notarizzazione sulla rete pubblica avviene una volta al giorno, raggruppando tutte le transazioni della giornata generate da Xailour in un merkle tree.

Nel payload della transazione viene salvato il valore del merkle root.


```
{
  "hash": "0xf8c4d712fc233c3cc333da06f8c8072d769815899a214dd6d611c4ee21c6b066",
  "merklePath": [
    {
      "left":
"081c54cf745f411a16c497e8ad41c1d27c76b864f37a8416a3d4220e8ab7a962d733601240deeb921bfa
92fe7f9a024c56a060a67f33ca0b7ebe3fc998a40241"
    },
    {
      "left": "7000374f78bd49c95cfa587b26c6d12d087831e8b5154e774a1b5d8c66f0d02a"
    },
    {
      "left": "b64902f86e5bedad8f463fbd283068a801f80ceddf0360a9fd5b7ef3ff8ffa04"
    },
    {
      "left": "abca9a9a79415090954159eb5fe20ce202643d4accf92b9a383489981de356eb"
    },
    {
      "left": "783c1787c2447a41d8cc71e4985a604a40ff7e6d33beccb2fb7a7a70570fe2ad"
    }
  ],
  "timestamp": 1611915131
```

```
}
```

Il campo hash contiene l'id della transazione.
Il campo merklePath contiene i riferimenti necessari per la validazione del documento all'interno del merkle tree.
Il campo timestamp indica il momento in cui la transazione è stata accettata in un blocco della catena.

Validazione

La validazione è un processo composto da tre passaggi distinti:

1. verifica integrità della ricevuta rispetto al file
2. verifica della notarizzazione sulla blockchain privata di XAINet
3. verifica della notarizzazione sulla blockchain pubblica di Ethereum

Controlli

I controlli vengono eseguiti nell'ordine indicato e si fermano al primo errore; solo nel caso vengano superati tutti si può considerare il documento valido.

Integrità ricevuta

Il controllo consiste nel recuperare dal documento le informazioni relative a dimensione e tipologia. Successivamente ne va calcolato l'hash applicando l'algoritmo indicato sulla ricevuta e vanno confrontati i risultati ottenuti con quelli riportati sulla ricevuta.
La verifica è positiva solo se hash, dimensione e tipo calcolati corrispondono con quelli indicati sulla ricevuta.

Notarizzazione XAINet

Per verificare l'integrità della notarizzazione su XAINet vengono effettuati i seguenti controlli:

1. deve esistere una transazione con id corrispondente all'hash indicato nella ricevuta
2. la transazione deve avere lo stesso payload e timestamp indicati sulla ricevuta
3. il payload deve corrispondere all'hash dei metadati del documento

Notarizzazione Ethereum

Per verificare l'integrità della notarizzazione su Ethereum vengono effettuati i seguenti controlli:

1. deve esistere una transazione con id corrispondente all'hash indicato nella ricevuta
2. la transazione deve essere stata effettuata da uno dei wallet di Xailour verso l'altro wallet di Xailour
3. la transazione deve avere lo stesso payload e timestamp indicati sulla ricevuta
    a. il timestamp non viene indicato quando la transazione non è stata ancora minata: in questo caso viene mostrato un warning
4. la foglia generata dall'unione dell'id della transazione privata con il suo payload deve essere contenuto nel merkle tree:
    a. la radice viene recuperata dal payload della transazione pubblica
    b. i path da utilizzare nella verifica sono quelli specificati nella ricevuta

Validatore di terze parti

E' possibile realizzare un validatore esterno a Xailour per verificare che la ricevuta del documento sia stata notarizzata - attraverso XAINotary - sulla rete pubblica di Ethereum senza utilizzare le API di XAINotary stesso.
L'unico aspetto non verificabile, in questo contesto, è l'esistenza della transazione su XAINet.
Questo è l'algoritmo che occorre implementare per poter eseguire la validazione del documento nel caso in cui non si vogliano utilizzare le API native messe a disposizione da XAINotary:

1. verifica integrità ricevuta:
    a. estrapolare dal file dimensione e tipologia
    b. calcolare l'hash utilizzando l'algoritmo indicato nella ricevuta
    c. calcolare l'hash dei metadati estratti e confrontarlo con il valore della sezione document
2. verifica transazione privata
    a. recuperare il campo payload dalla sezione TRANS_PRV e verificare che la sua deserializzazione corrisponda con il contenuto della sezione DOCUMENT

```
// esempio di deserializzazione in javascript
const {StringDecoder} = require('string_decoder');
const payload =
'0x7b2273697a65223a20343733363935312c20226d74797065223a20226170706c69636174696f6
e2f706466222c202268617368223a2022346665363734343136663263353037323461373962626261
643438366632326461613864333934386434663634306136666536616663616137386434303566
313232663135363161623339366138653235626134616137363662366265336234633166346163
623363353731393133316234306531386666353538653462393130222c2022616c67223a2022736
```

```
861353132227d';
const decoder = new StringDecoder('utf8');
const input_ = decoder.write(Buffer.from(payload.substring(2),
'hex'));
const data = JSON.parse(input_)
console.log(data);
/*{
size: 4736951,
mtype: 'application/pdf',
hash:
'4fe674416f2c50724a79bbad486f22daa8d3948d4f640a6fe6afcaa78d405f122f1561ab396a8e25ba4
aa766b6be3b4c1f4acb3c5719131b40e18ff558e4b910',
alg: 'sha512'
}*/
```

```
# esempio di deserializzazione in python
import hexbytes
import json
payload='0x7b2273697a65223a20343733363935312c20226d74797065223a20226170706c696361
74696f6e2f706466222c202268617368223a2022346665363734343136663263353037323461373
962626164343836663232646161386433393438643466363430613666653661666361613738643
430356631323266313536316162333936613865323562613461613736366236626533623463311
663461636233633537313931333162343065313866663535386534623931302 22c2022616c67223
a2022736861353132227d'
input_ = hexbytes.HexBytes(payload).decode()
data = json.loads(input_)
print(data)
# {'size': 4736951, 'mtype': 'application/pdf', 'hash':
'4fe674416f2c50724a79bbad486f22daa8d3948d4f640a6fe6afcaa78d405f122f1561ab396a8e25ba4
aa766b6be3b4c1f4acb3c5719131b40e18ff558e4b910', 'alg': 'sha512'}
```

3. verifica transazione pubblica
    a. recuperare dalla rete mainnet di Ethereum la transazione con id corrispondente al campo hash della sezione TRANS_PBL
    b. confrontare il timestamp della transazione con quello indicato nella ricevuta
    c. verificare che i due attori coinvolti nella transazione siano uno dei due seguenti indirizzi (entrambi possono essere l'indirizzo from o quello to):
        i. 0x652481b9748bb34f83fdeeb76f09d08809e13530

ii. 0xd7f8ed72134719b76a3ecbf281239802d70b1ddb

d. creare la foglia del merkle tree:
   i. concatenare i campi hash e payload della sezione TRANS_PRV separandoli con un "." (in entrambi escludere il prefisso "0x")
   ii. creare un hash usando l'algoritmo sha512
e. verificare che il merkle tree sia corretto usando:
   i. la foglia appena generata
   ii. il merkle root presente nel payload della transazione pubblica
   iii. i path indicati nella ricevuta