Condividere informazioni in modo sicuro combinando Git e Blockchain

Paolo Speziali

Università degli Studi di Perugia - Dipartimento di Ingegneria

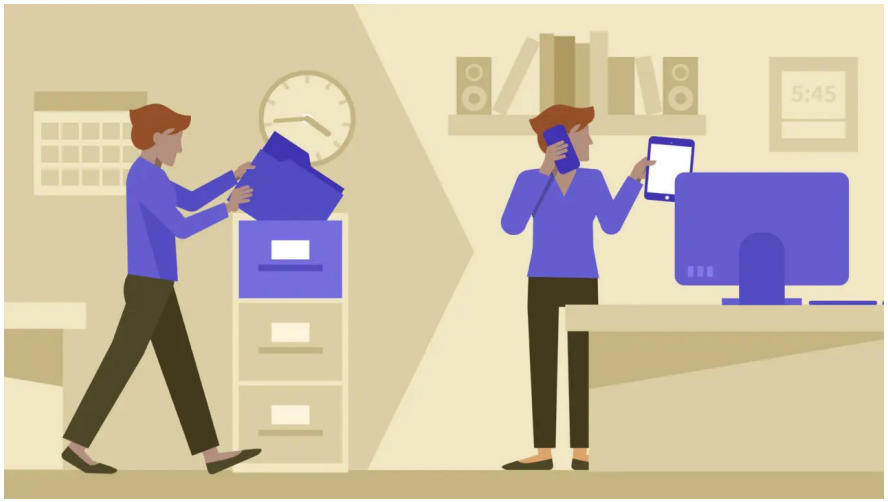
A.D. 1308
unipg
DIPARTIMENTO
DI INGEGNERIA

A.A. 2020/2021

Indice

La digitalizzazione

È in atto, negli ultimi anni, un piano di **digitalizzazione** delle PA. Esso mira all'evoluzione tecnologica di tutte le sue mansioni e alla creazione di portali web per il cittadino. L'esigenza di questa trasformazione si è fatta sentire anche da parte dell'Unione Europea, che con il Recovery Fund ci sta fornendo i fondi per attuarla.

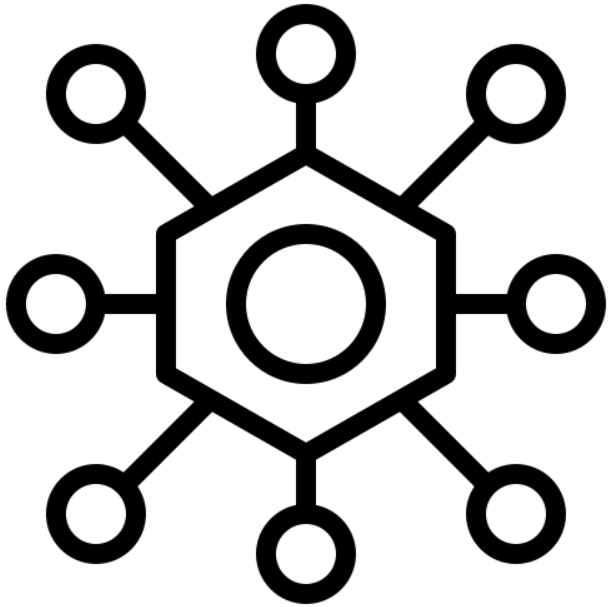


Il problema della burocrazia

Tuttavia, anche avendo i fondi necessari, sono molti i problemi che non permettono una digitalizzazione totale delle PA, tra cui la lenta e farraginosa macchina della burocrazia. Sembra necessario un processo di **sburocratizzazione** grazie a degli strumenti digitali che permettano di salvare, validare e condividere documenti in maniera sicura.

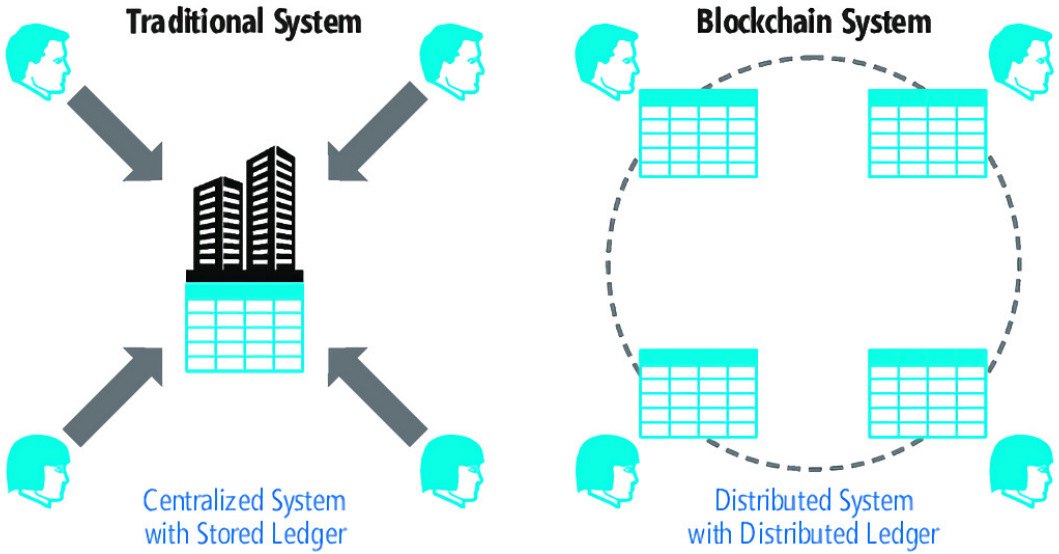
Gli strumenti attuali

Gli strumenti attualmente in utilizzo hanno un'architettura centralizzata: un'entità centrale si occupa dell'immagazzinamento e della verifica dei dati degli utenti. Ciò è potenzialmente rischioso, sia perché potrebbero verificarsi attacchi alle unità centrali, sia perché mettiamo in mano di un'azienda esterna i nostri dati.



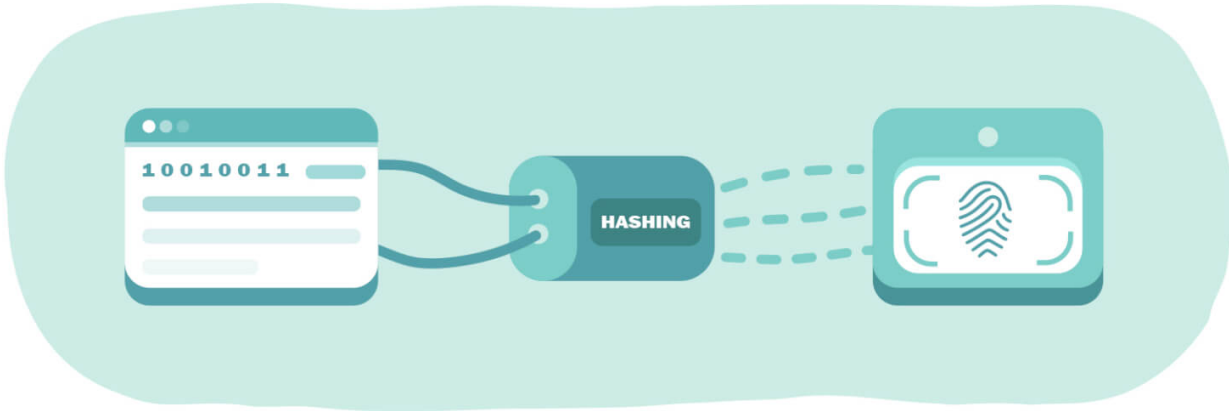
Strumenti distribuiti

Usando invece strumenti distribuiti, sia per la gestione dei file, per cui utilizzeremo **Git**, sia per la verifica delle informazioni, per cui useremo la **blockchain**, saremo in grado costruire uno strumento che può affidarsi alla parola di una moltitudine di entità, rendendo molto più complicati e rilevabili attacchi e manomissioni.



Funzioni crittografiche di hashing

Una funzione crittografica di hashing è una funzione di hashing, ovvero una funzione che permette di associare, a una qualsiasi sequenza m di lunghezza arbitraria in input, una sequenza in output $h(m)$ di lunghezza costante, con alcune proprietà aggiunte che deve seguire per poter essere considerata *crittograficamente sicura*. Esse impediscono di risalire all'input originale e facilitano i **controlli di integrità sui file**.



L'Obiettivo

Il sistema progettato ha lo scopo di riuscire a fornire a chi ne usufruisce un livello di sicurezza aggiuntivo sopra il software Git tramite un'opportuna comunicazione con la blockchain.
Il software, grazie a un'interfaccia user-friendly, deve permettere non solo di gestire le directory come normali repository, ma fornire anche degli utili strumenti di salvataggio di hash su blockchain, esportazione di sottoinsiemi di repository e verifica sia di singoli file che di moltitudini. Tutto ciò implementato con operazioni più o meno severe (e quindi onerose) e con un occhio di riguardo anche alla quantità di dati da memorizzare durante l'implementazione.

Perché blockchain?

L'utilizzo della blockchain nel progetto è giustificato da:

- Natura condivisa → Transazioni facilmente tracciabili
- Decentralizzazione → Resistenza allo **SPOF**[1]
- Immutabilità → Garantisce integrità dei dati
- Validazione *peer-to-peer* → Potere distribuito
- Disintermediazione → Eliminazione di *middle-men* e dei loro costi

[1] Single Point Of Failure

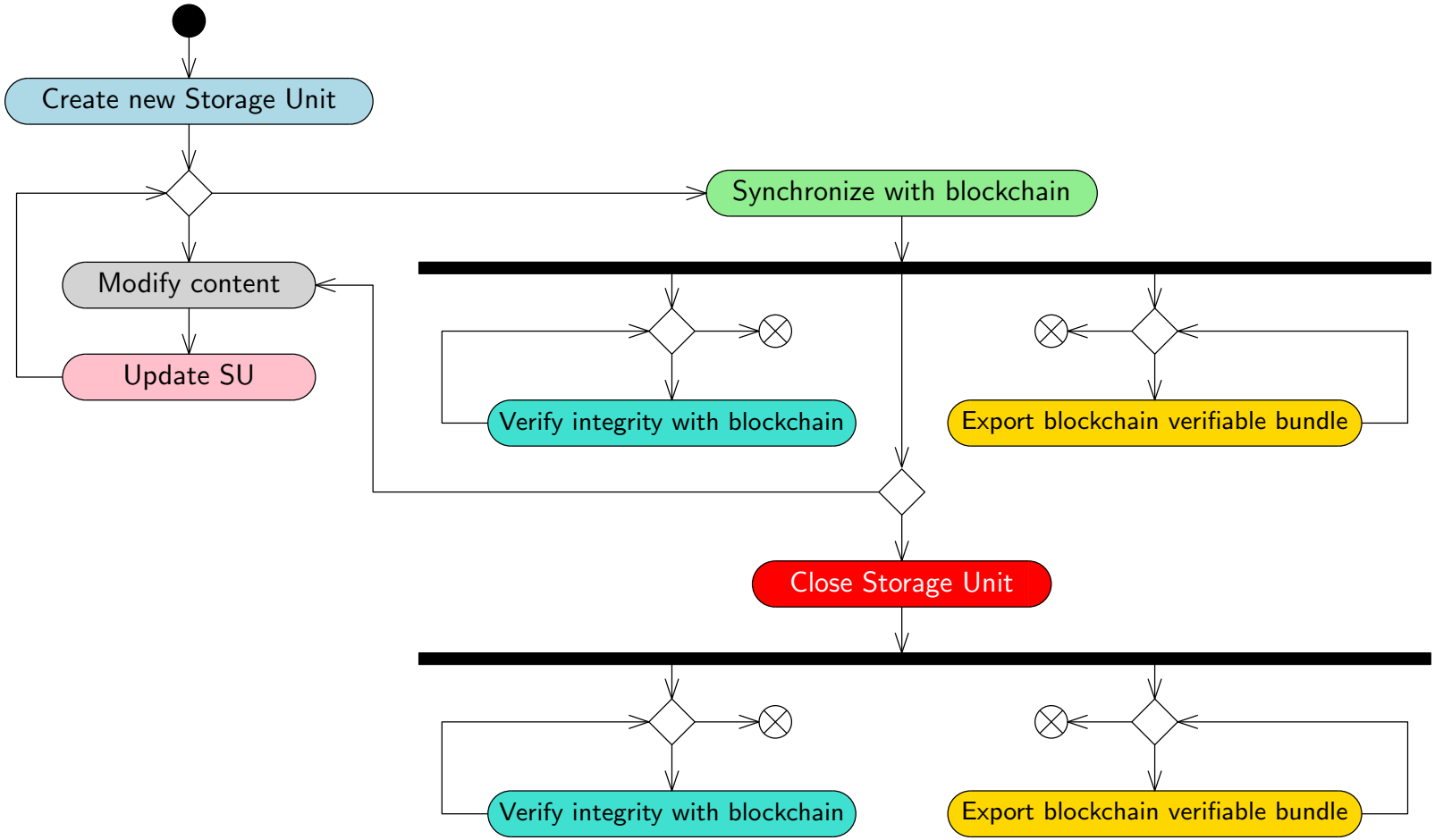
Il costo della blockchain

Il salvataggio delle informazioni su blockchain ha però un costo proporzionale al quantitativo di dati che vorremo memorizzarci.
Per superare questo problema dovrà essere implementata una soluzione che sfrutti degli accumulatori crittografici per memorizzare l'identità di molti collettivi di documenti con un unico hash.
Ovviamente la loro struttura dovrà essere tale da permetterci di andare a reperire informazioni passate e già calcolate in un tempo che sia relativamente ragionevole.

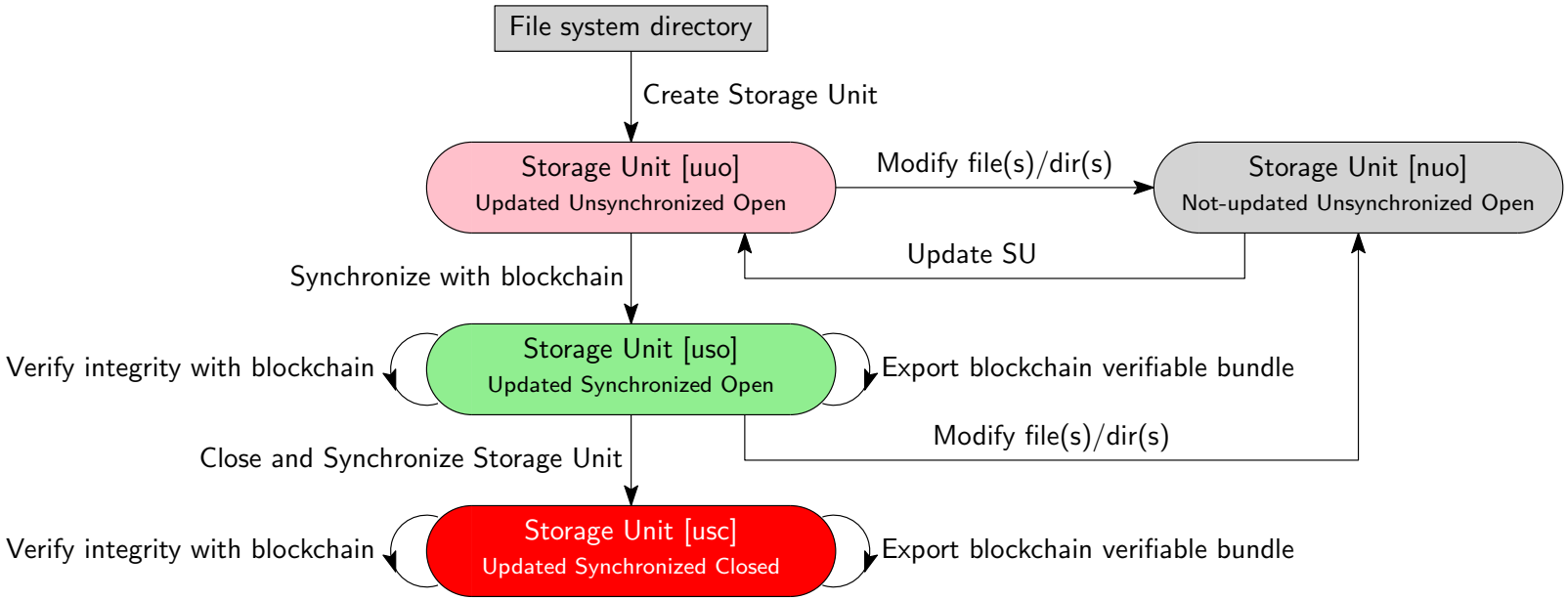
Il Software PineSU

Un'implementazione del sistema ideato è l'applicativo **PineSU**.
PineSU è un software *lightweight* in Javascript e che sfrutta il runtime Node.js.
L'applicazione va a considerare gli insiemi di file come delle entità chiamate **Storage Unit** (SU) con cui va ad inglobare logicamente una repository Git, costruendo, tramite metadati, una struttura introno ad essa. Queste SU sono le unità su cui si andranno ad effettuare le singole operazioni, eccetto la registrazione su blockchain che si svolgerà collettivamente con l'ausilio di accumulatori crittografici.

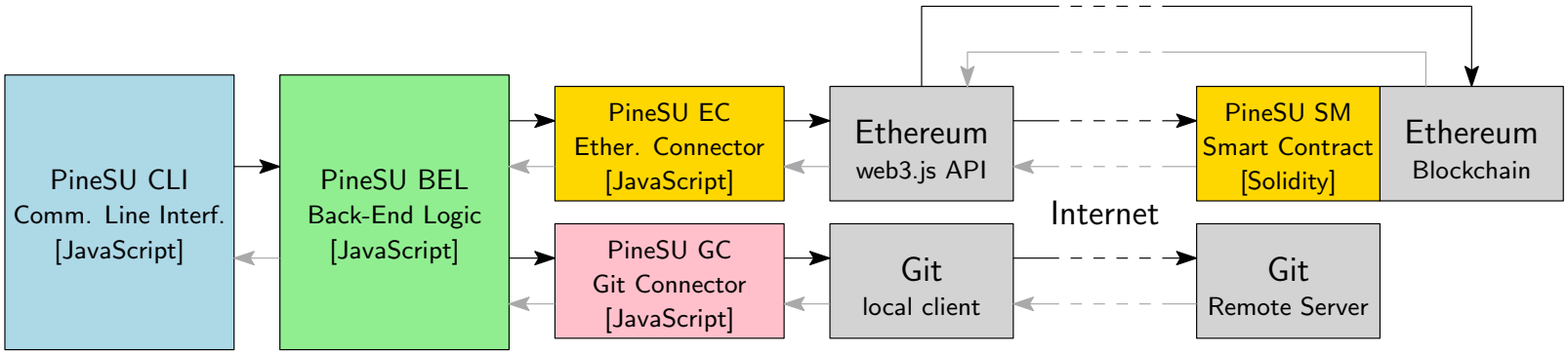
Workflow



Ciclo vitale di una Storage Unit



Architettura



Architettura (Cont.)

- *PineSU* **CLI** *(Command Line Interface)*: Modulo che crea l'interfaccia utente con cui è possibile interagire e richiama le funzioni degli altri moduli di conseguenza.
- *PineSU* **BEL** *(Back End Logic)*: Questo componente è il nucleo di PineSU. Gestisce tutte le SU e controlla la comunicazione con la blockchain e il client Git locale.
- *PineSU* **EC** *(Ethereum Connector)*: Si interfaccia con le API della blockchain.
- *PineSU* **GC** *(Git Connector)*: Si interfaccia con il client Git.
- *PineSU* **SM** *(Smart Contract)*: Permette registrazioni permanenti di singole SU nella blockchain.

Gli accumulatori di PineSU

- *SU Merkle Tree*: Le sue foglie sono gli hash dei file e directory della SU, la sua root è l'hash della SU stessa.
- *Storage Group (***SG***)*: Le sue foglie sono le SU da registrare su blockchain nella prossima transazione.
- *Merkle Calendar (***MC***)*.

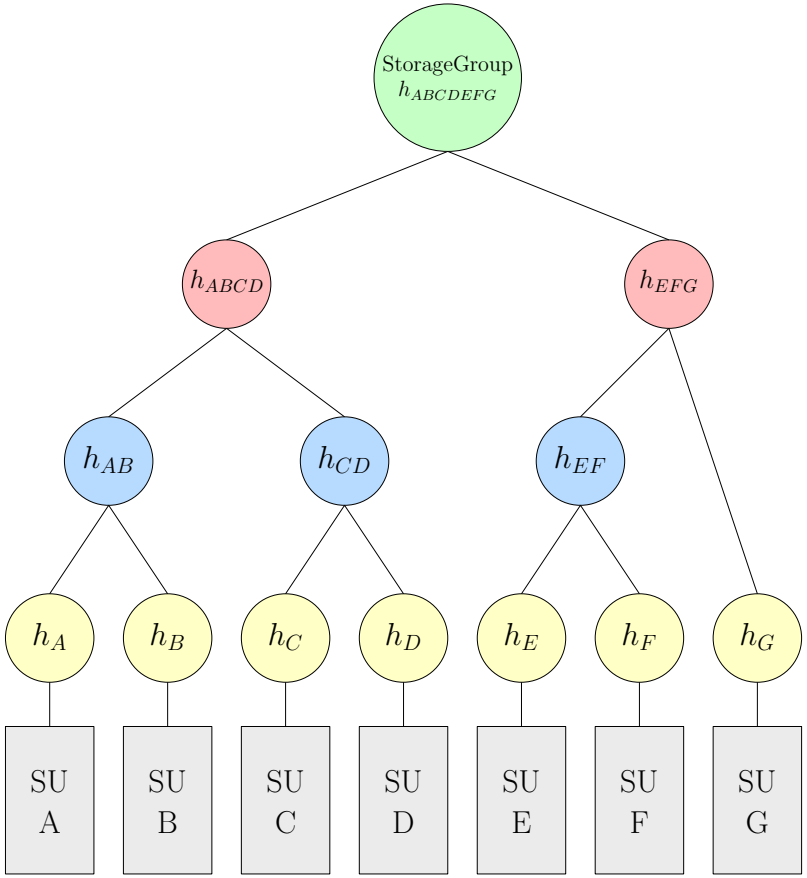


Figura: Uno Storage Group

Gli accumulatori di PineSU (Cont.)

Un **Merkle Calendar** è un albero in cui le foglie sono i Blockchain Synchronization Point (**BSP**), istanze di Storage Group, a loro volta raggruppate in nodi rappresentanti mesi e anni, ciò rende i reperimenti di registrazioni passate più agevoli e veloci.

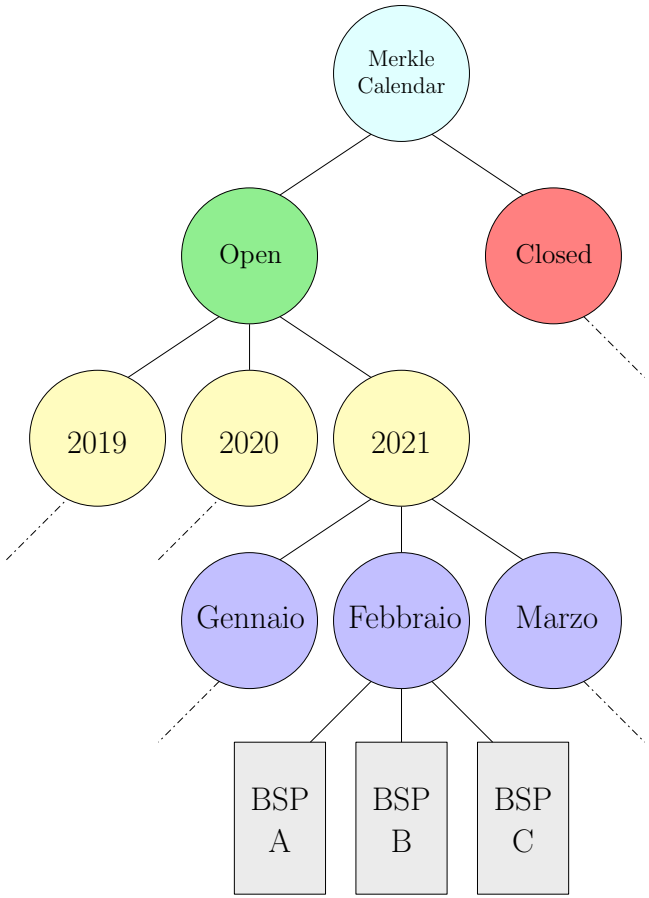


Figura: Un Merkle Calendar

Le operazioni disponibili

1. Creazione di una Storage Unit o Ricalcolo di una Storage Unit pre-esistente
2. Staging di una Storage Unit nello Storage Group
3. Registrazione dello Storage Group nella Blockchain
4. Chiusura di una Storage Unit
5. Esportazione di sottoinsiemi di file da una SU
6. Controllo di integrità di singoli file esportati da altre SU
7. Controllo di integrità su una SU