Condividere informazioni in modo sicuro combinando Git e Blockchain

Studente: Paolo Speziali
Relatore: Luca Grilli

Università degli Studi di Perugia - Dipartimento di Ingegneria
Corso di laurea triennale in Ingegneria Informatica ed Elettronica

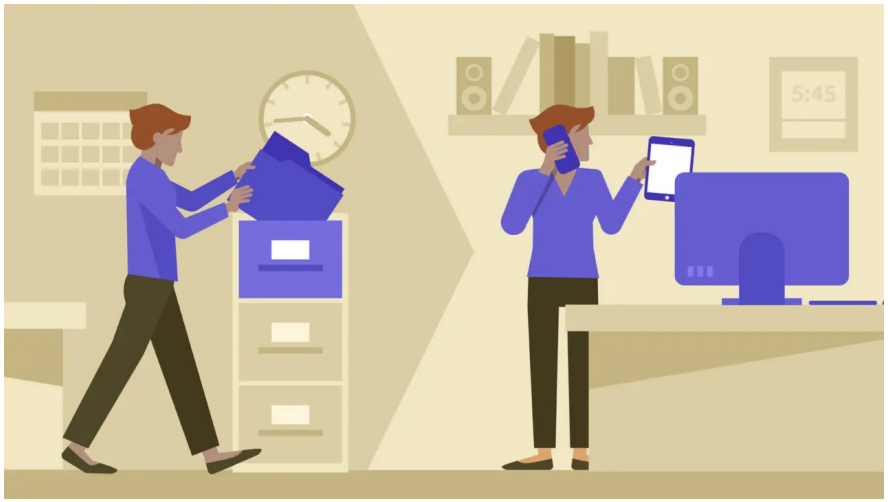
A.D. 1308
unipg
DIPARTIMENTO
DI INGEGNERIA

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, ben **11,75 milioni di euro**.



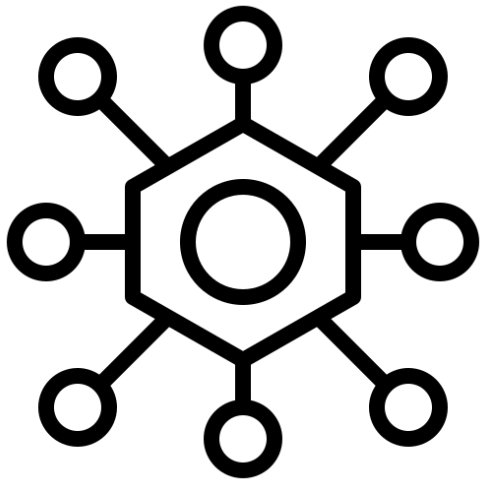
Il problema della burocrazia

Il più grande avversario della digitalizzazione è la **burocrazia** italiana: i suoi processi sono **lenti** e **complessi** anche a causa dell'**importanza** dei documenti da gestire. È necessaria una **sburocratizzazione** grazie a degli strumenti digitali che permettano di **salvare**, **validare** e **condividere** documenti senza abbassare il **livello di sicurezza**.

Gli strumenti attuali

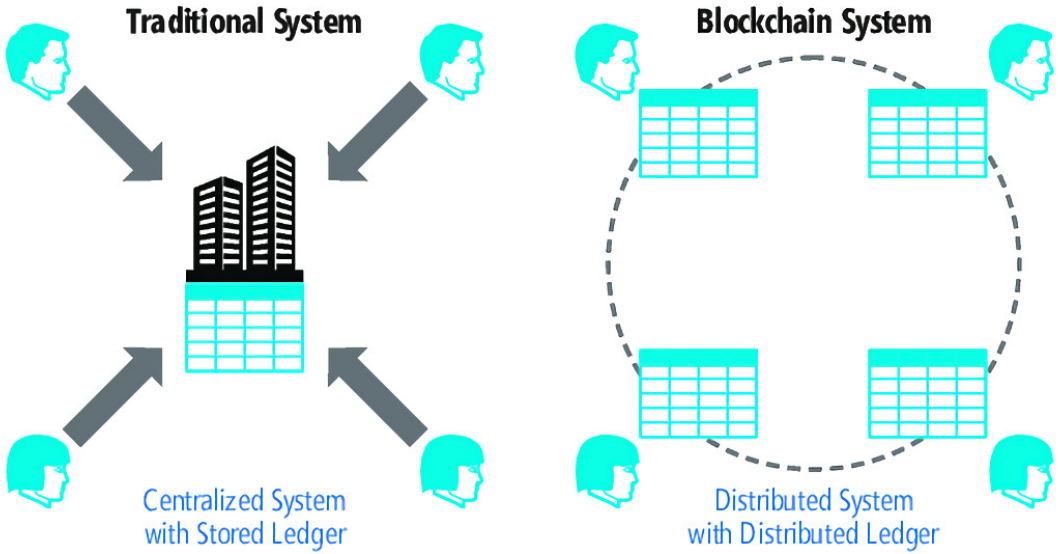
Uno strumento digitale solitamente segue uno di questi due paradigmi: **centralizzato** e **distribuito**.
Nel primo un'entità centrale si occupa dell'**immagazzinamento** e della **verifica** dei dati degli utenti.
Ciò ha diversi **svantaggi**:

- Potenziali attacchi all'entità
- Possibile uso malevolo dei nostri dati
- Alti costi d'intermediazione



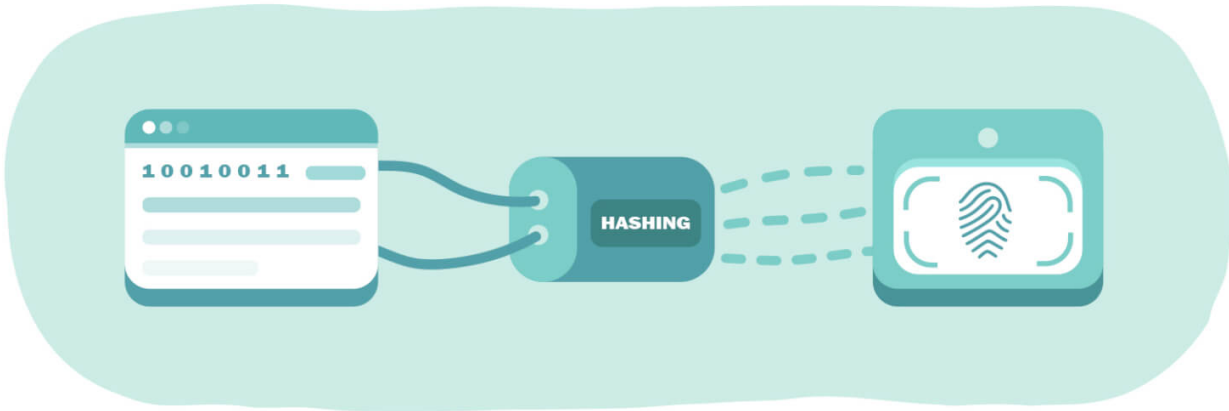
Strumenti distribuiti

Usando invece **un'architettura distribuita**, sia per la gestione dei file, sia per la verifica delle informazioni, 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**.

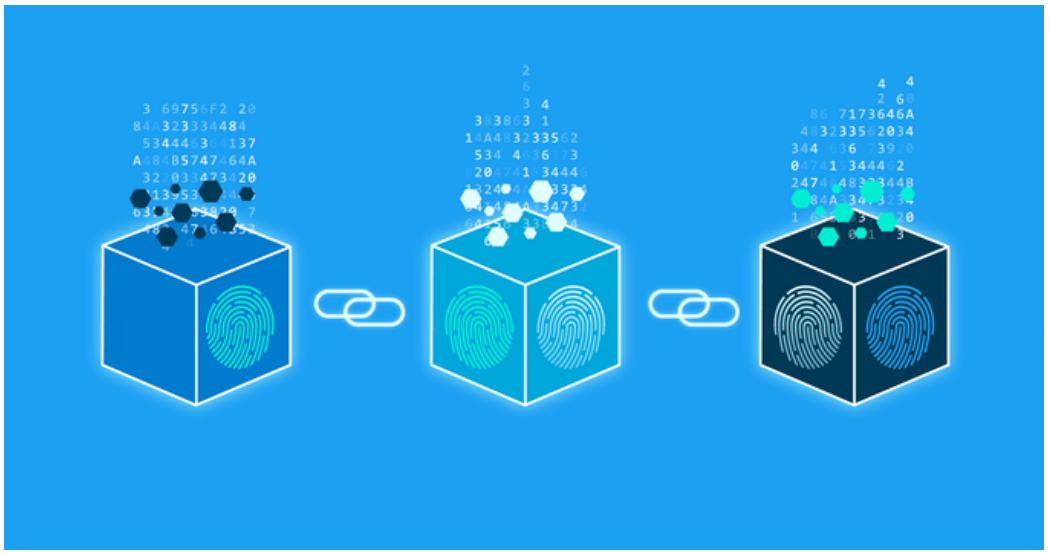


Git

Git è è il sistema di controllo di versione (**VCS**) distribuito più diffuso al mondo.
Un VCS suddivide gli insiemi di file e directory in repository. Git modella ogni repository come una *sequenza di snapshot* di un piccolo file system. Ogni volta che un utente salva lo stato del suo progetto (tramite l'operazione di *commit*), Git crea uno snapshot di tutti i file e le directory sotto controllo di versione in quel momento e lo archivia nel suo database locale. Inoltre, quasi ogni operazione di Git va ad aggiungere informazioni al suo database, anche se si tratta di un'operazione di rimozione, ciò assicura che ogni cambiamento sia reversibile.

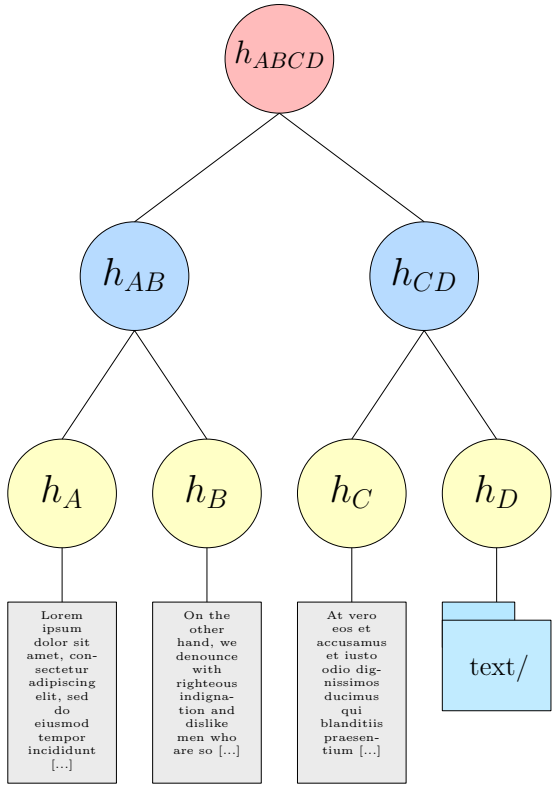
Blockchain

La **blockchain**. . .



Accumulatori crittografici

Gli **accumulatori crittografici** sono strumenti che permettono di comprimere molti elementi informativi in una costante di dimensione fissa. Un esempio ne sono i **Merkle Tree**, alberi binari in cui ogni foglia corrisponde all'hash di un elemento. Risalendo verso la radice ogni nodo interno calcolerà il proprio hash concatenando gli hash dei nodi figli, infine si otterrà una radice (**Merkle Root** o MR), univoca a quella lista di elementi che l'albero ha come foglie, in quella sequenza.



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 $\rightarrow$ Transazioni facilmente tracciabili

[1] Single Point Of Failure

Perché blockchain?

L'utilizzo della blockchain nel progetto è giustificato da:

- Natura condivisa → Transazioni facilmente tracciabili
- Decentralizzazione → Resistenza allo **SPOF**[1]

[1] Single Point Of Failure

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

[1] Single Point Of Failure

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

[1] Single Point Of Failure

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

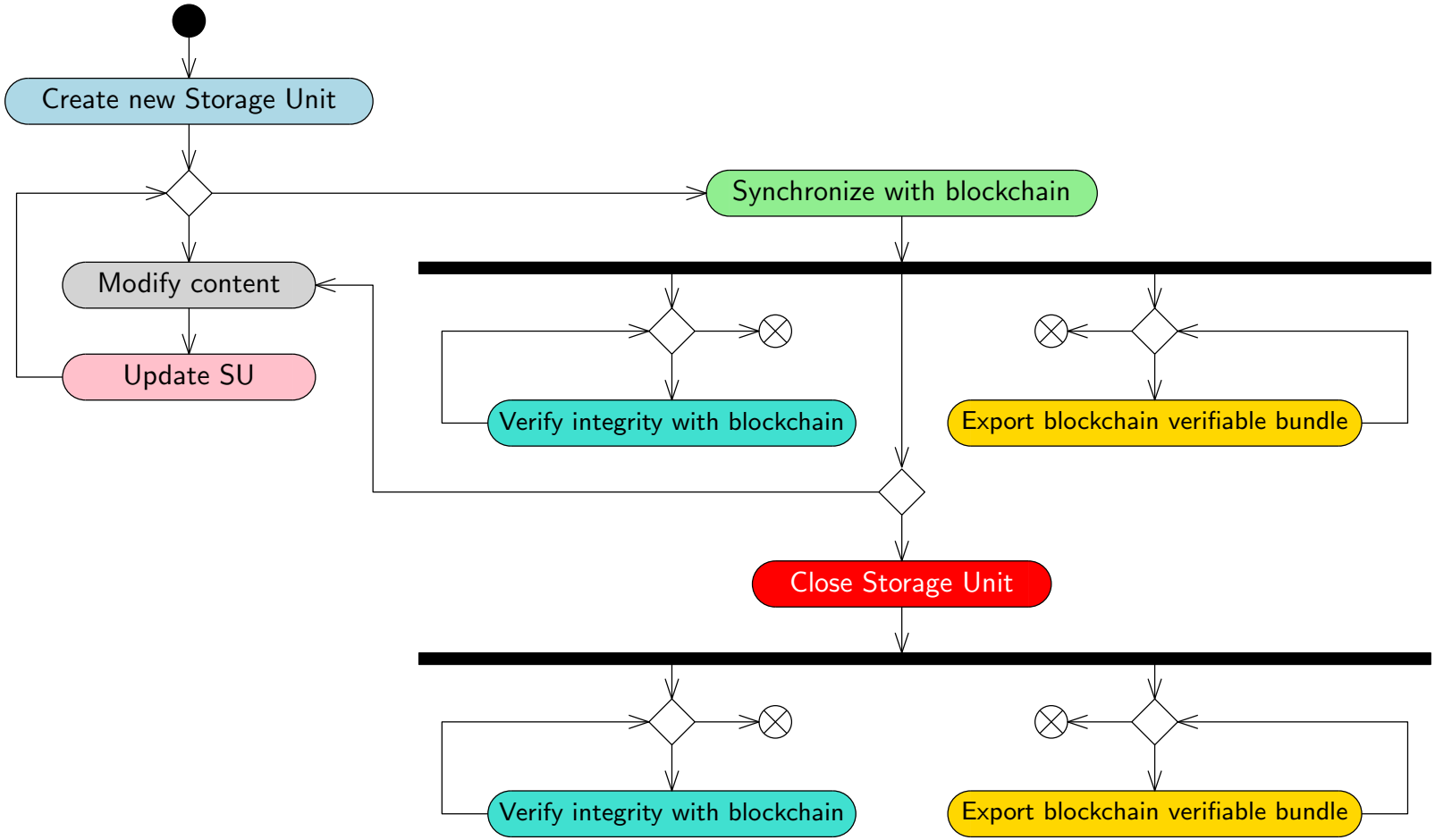
Il costo della blockchain

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.

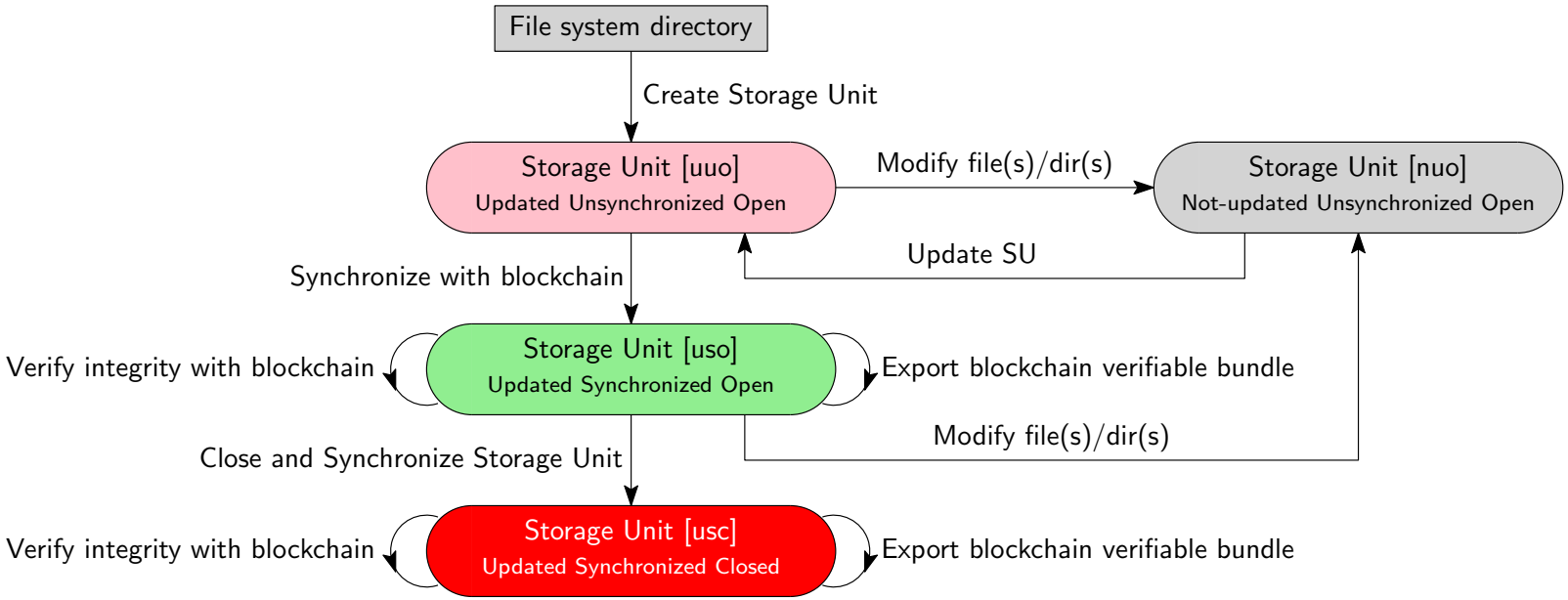
Il Software PineSU

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.

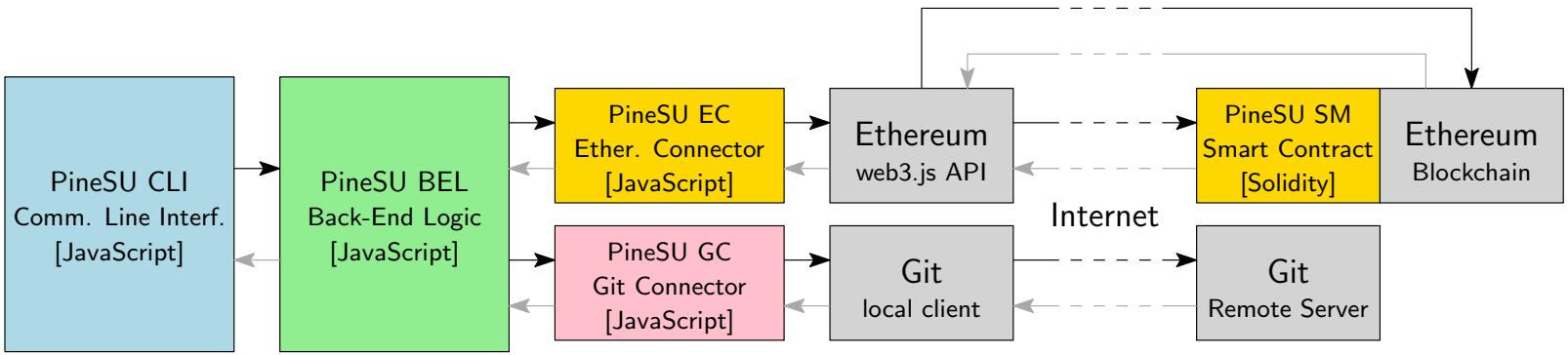
Workflow



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.

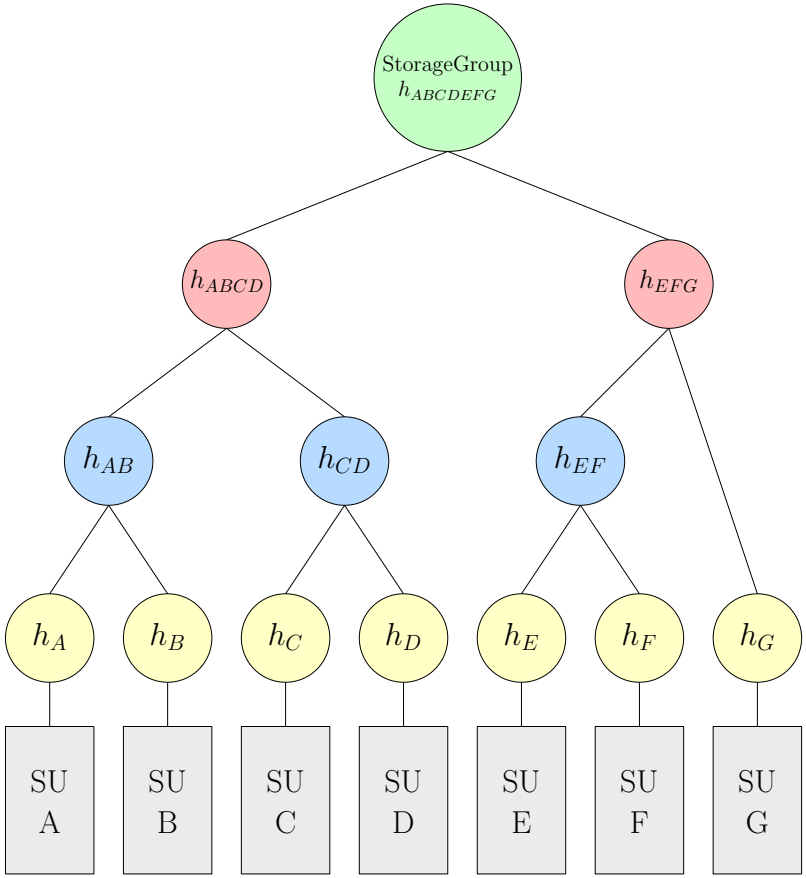


Figura: Uno Storage Group

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.

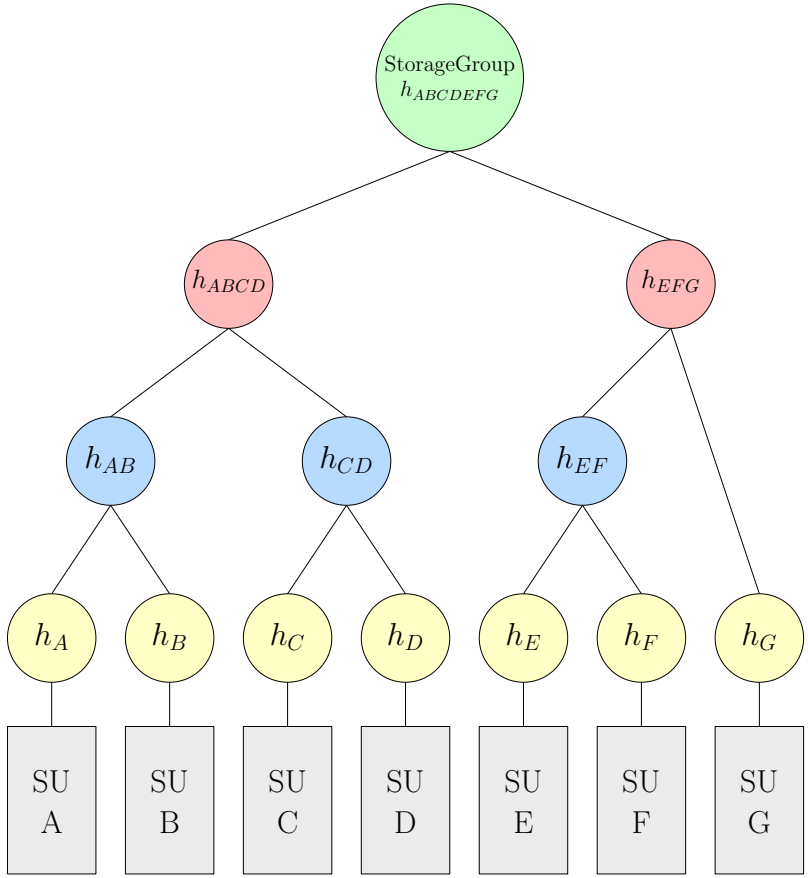


Figura: Uno Storage Group

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***)*.

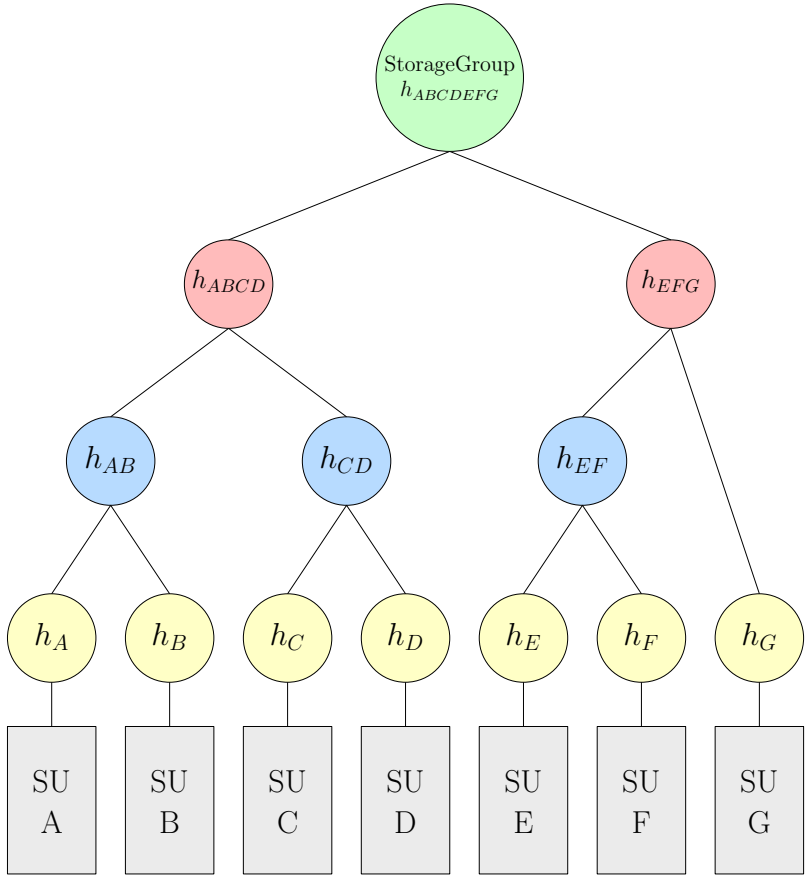


Figura: Uno Storage Group

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.

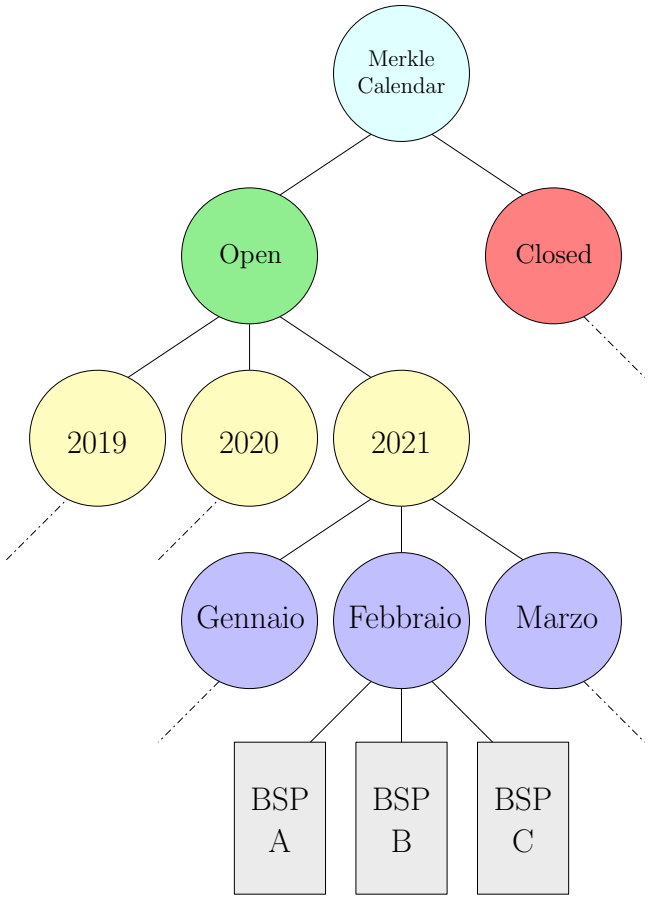


Figura: Un Merkle Calendar

Le operazioni disponibili

1. Creazione di una Storage Unit o Ricalcolo di una Storage Unit pre-esistente

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

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

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

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

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

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