ZKP Based Voting System

Gaspar Zuker y Agustín Venegas

Roadmap

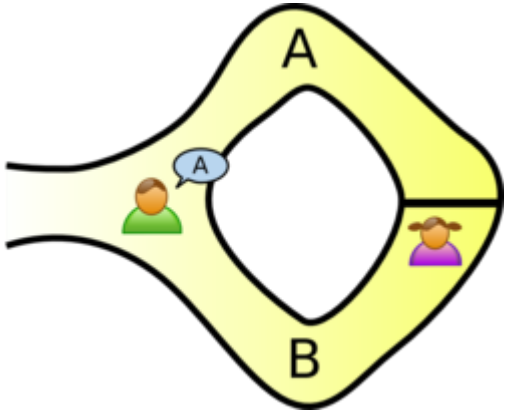
- Introducción/Motivación
  - ¿Es posible llevar a cabo una elección privada en una blockchain pública?
  - Zero Knowledge Proofs
- Detalles Técnicos
  - ZoKrates
  - Encriptación homomórfica
- Conclusiones

Introducción y motivación

¿Es posible llevar a cabo una elección privada en una blockchain pública?

- Nos preguntamos esto teniendo en cuenta que todo lo que se ve en la blockchain es público.
- Quisimos mantener la mayoría de las propiedades buscadas en un sistema de votación:
    - No se linkea el voto con el votante (Voto anónimo).
    - No se revelan los valores de votos individuales (Voto secreto).
    - Coersion Resistance ([HV-08]).
    - No se puede votar múltiples veces.
    - Solo pueden votar usuarios habilitado.

Zero Knowledge Proofs



¿Que necesitamos demostrar sin revelar?

- Demostrar habilitación para votar sin revelar la clave de votación.
- Demostrar que el voto es válido sin revelar su valor.
- Demostrar que todavía no votamos.

- Demostrar que conocemos el resultado de la votación sin mostrar la clave privada que desencripta resultados.
- Demostrar que el resultado de la votación que publicamos es válido.

Roadmap

- Introducción/Motivación
  - ¿Es posible llevar a cabo una elección privada en una blockchain pública?
  - Zero Knowledge Proofs
- Detalles Técnicos
  - ZoKrates
  - Encriptación homomórfica
- Conclusiones

¿Como logramos estas cosas?

- Utilizamos “ZoKrates” para crear las ZKP y probarlas.

- Tiene casi la misma sintaxis que Noir y genera circuitos compatibles con CirCom.

Noir ==> circom CIRCUIT COMPILER

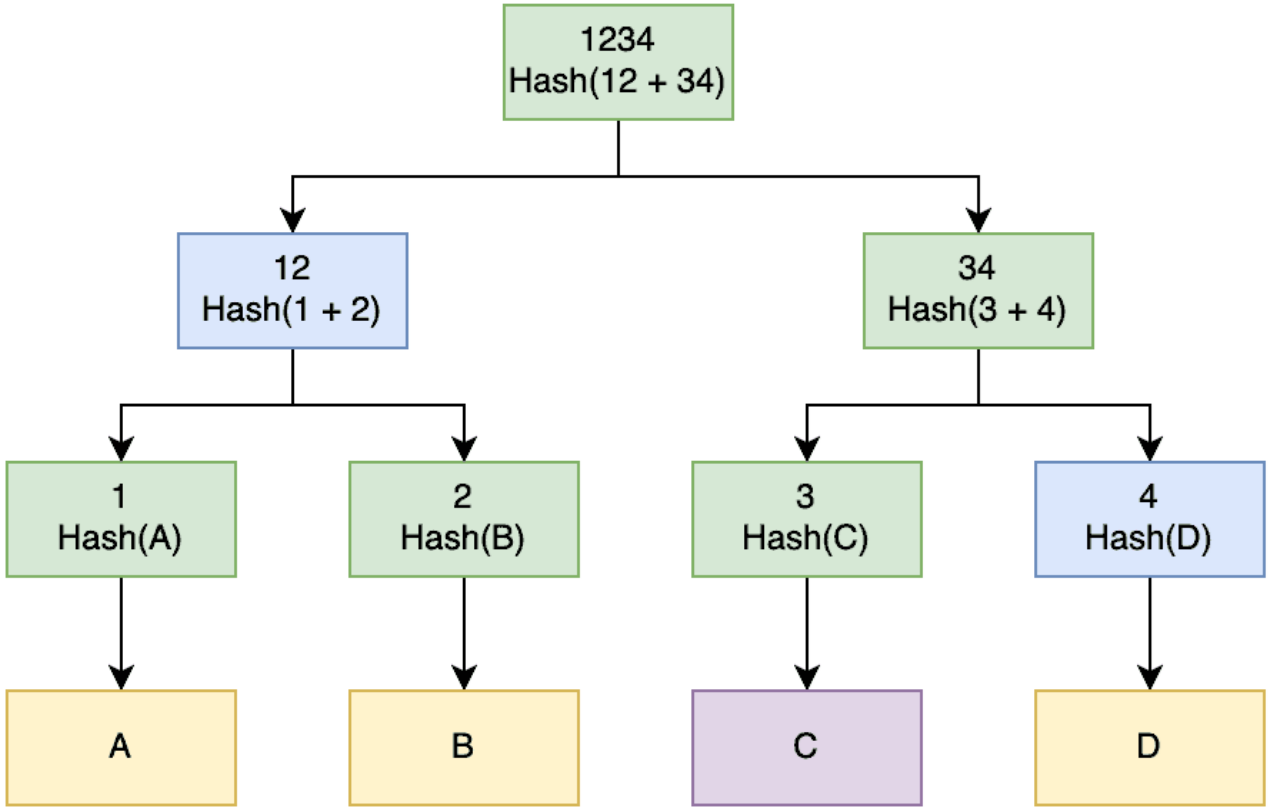
- Es una versión simple con integración en remixide, que nos permitió aprender más fácilmente los conceptos.

Habilitación para votar ⇒ Demostrar conocimiento de la preimagen de una hoja de un Merkle Tree

```
def checkMerkleInclusion(u32[8] root, private u32[8] token, private
bool[DEPTH] directionSelector, private u32[DEPTH][8] path) -> bool {
    // Start from the leaf
    u32[8] leaf = leafHash(token);
    u32[8] mut digest = leaf;

    // Loop up the tree
    for u32 i in 0..DEPTH {
        (u32[8], u32[8]) s = select(directionSelector[i], digest, path[i]);
        digest = hash(s.0, s.1);
    }

    return digest == root;
}
```



El punto clave está en hacer leafHash(token), ya que
Solo vamos a poder generar una prueba válida si
Conocemos la preimagen

Encriptación Homomórfica

RECALL

- Es una encriptación con propiedades homomórficas.

- Nos permite desencriptar el resultado de sumar todos los votos directamente.

$$\mathcal{C}_{r1}(t_1) \odot \mathcal{C}_{r2}(t_2) = \mathcal{C}_r(t_1 \oplus t_2)$$

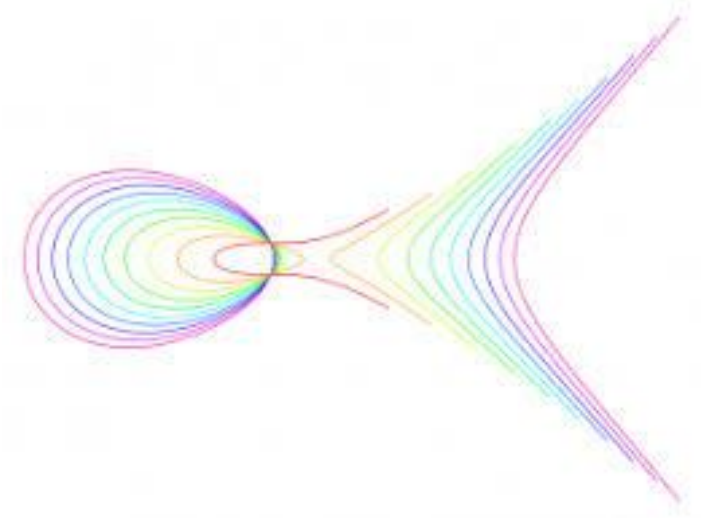
- En nuestro TP limitamos a una votación binaria para simplificar el esquema, dado que nos centramos en las ZKPs.

Voto válido – Mostrar que nuestro valor está bien calculado

- La prueba calcula “a mano” la encriptación del voto y la compara con la encriptacion que se va a hacer pública
- El smart contract la verifica sin usar variables privadas

La encriptación de números utilizando esquemas pesados puede ser demasiado costosa en el contexto de una ZKP. ¿Cómo resolvemos esto?

Curvas Elípticas



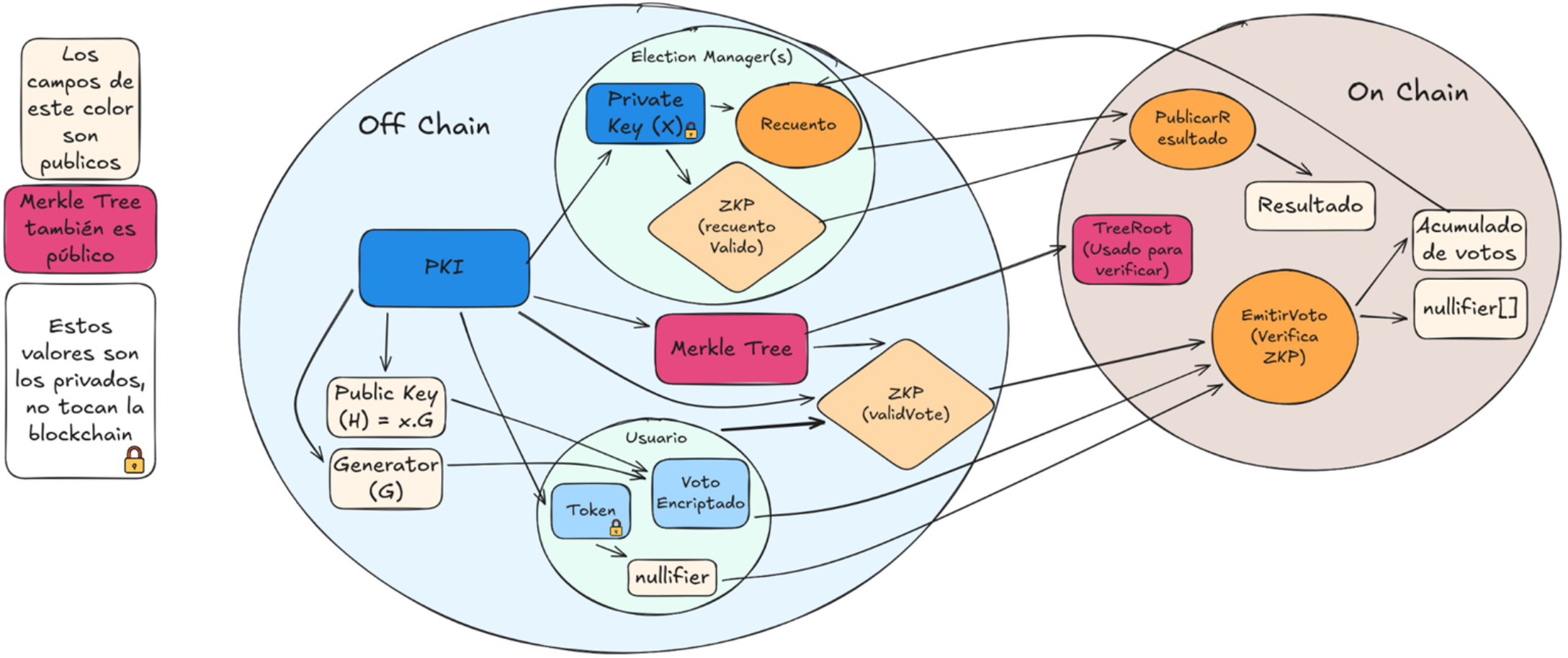
ElGamal – Encriptación Homomorfica con curvas elípticas

- Utilizamos las librerías de ZoKrates para curvas elípticas, con BabyJubJub como la curva elegida, y realizamos una implementación de EcElGamal.
- Podríamos haber utilizado Pallier ~~(como el otro grupo)~~ pero quisimos utilizar ElGamal porque (según investigamos) es más ligero para ZKPs y para mostrar una alternativa diferente pero igualmente segura [Knirsch et. al.]

Para implementarlas utilizamos las siguientes librerías, asegurándonos de que tenían los mismos parámetros:

- ZoKrates (librería estándar): https://github.com/Zokrates/ZoKrates/tree/latest/zokrates_stdlib/stdlib/ecc
- Solidity: https://github.com/yondonfu/sol-baby-jubjub/blob/master/contracts/CurveBabyJubJub.sol
- Python: zokrates_pycrypto.babyjubjub (Específicamente compatible con la librería estándar de ZoKrates)

Diagrama de la aplicación:



Roadmap

- Introducción/Motivación
  - ¿Es posible llevar a cabo una elección privada en una blockchain pública?
  - Zero Knowledge Proofs
- Detalles Técnicos
  - ZoKrates
  - Encriptación homomórfica
- Conclusiones

¿Qué conclusiones nos llevamos de este sistema de votación?

- Es posible utilizar pruebas de conocimiento nulas para conservar valores privados en una blockchain pública
- Es muy difícil asegurar la no coerción en una votación virtual, debido a la naturaleza de la votación a distancia
- Este esquema funciona independientemente de como esté implementada la PKI utilizada para distribuir las claves, sin embargo, agregar un sistema de creación de claves distribuida y compartir las claves con un treshold de usuarios colaboradores necesarios para desencriptar mejoraría la confianza en el sistema.
- Otra mejora que podría hacerse al TP es utilizar una mixnet para reducir la rastreabilidad de los votos.

¿Qué aprendimos en este tp?

- Los pasos y los elementos necesarios para realizar una zk-SNARK un verificador asociado.
  - La sintaxis para utilizar el lenguaje de Zokrates/Noir, basado en rust
  - Sus limitaciones, como la ausencia de while loops o arrays dinámicos
  - Como elegir que funciones utilizar para que sean ZK-Friendly (por ejemplo, funciones de hash o de encriptación)
- Como utilizar la interfaz del verificador y conectarla a un contrato en Solidity.
- Como utilizar encriptación homomórfica y desencriptarla.
- Como configurar parámetros de curvas elípticas y utilizar librarías para operar con las mismas, incluso entre varios lenguajes.
- Como manejar la endianness para transferir datos entre Python (local), solidity y ZoKrates, con los tipos de datos poco versátiles de las ZKP

References

- “Can a Public Blockchain Keep a Secret?” Fabrice Benhamouda et.al. https://eprint.iacr.org/2020/464.pdf
- “Helios: Web-based Open-Audit Voting” Ben Adida https://www.usenix.org/legacy/events/sec08/tech/full_papers/adida/adida.pdf
- “Comparison of the Paillier and ElGamal Cryptosystems for Smart Grid Aggregation Protocols “ https://www.en-trust.at/papers/Knirsch20a.pdf
- “Simple Verifiable Elections”, Josh Benaloh https://www.usenix.org/legacy/event/evt06/tech/full_papers/benaloh/benaloh.pdf
- “Cryptographers Held an Election. They Can’t Decrypt the Results”, NY Times. https://www.nytimes.com/2025/11/21/world/cryptography-group-lost-election-results.html