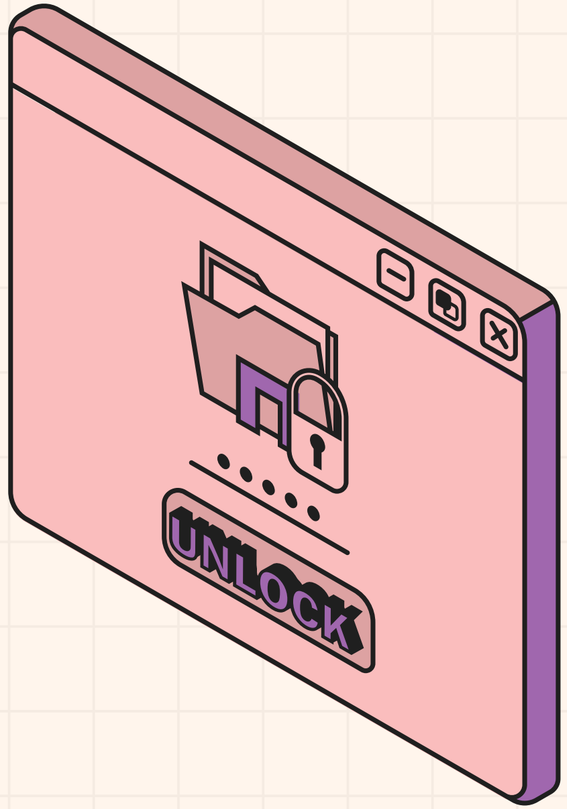
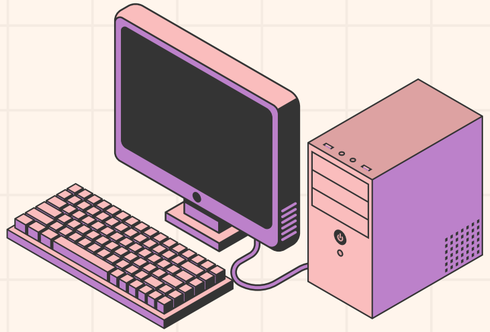
ZERO KNOWLEDGE PROOF

L'ANONYMAT DANS LA BLOCKCHAIN

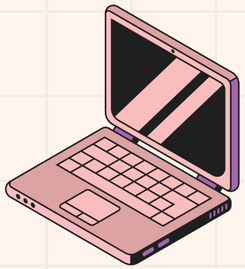


Tristan Looten, Maël Lator, Athur Vialix, Clovis Lortat-Jacob, Tom Delande

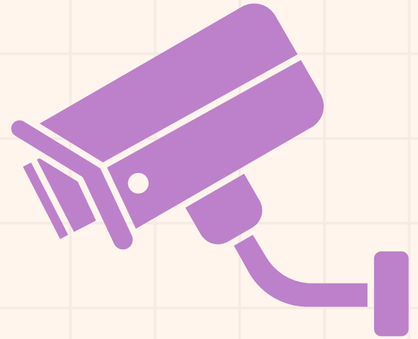
SOMMAIRE



- Anonymat dans la blockchain
- Zero knowledge proof
- Utilisation dans la blockchain
- Preuve de concept
- D’autres applications ?



ANONYMAT DANS LA BLOCKCHAIN



Les blockchain publique (Bitcoin, Ethereum) sont **pseudonymes** : transactions, adresses et montants visibles par tous sur le registre distribué

Gros problème d'anonymat si les adresses sont liées à une identité. Les **fuites de données** massives sont déjà arrivées !

Cela soulève un enjeu majeur et très actuel dans les blockchain :

Comment prouver quelque chose sur une transaction sans révéler la transaction elle-même ?

ZERO KNOWLEDGE PROOF (1)

“Protocole cryptographique permettant de prouver qu’une affirmation est vraie sans révéler aucune informations supplémentaire à son sujet”

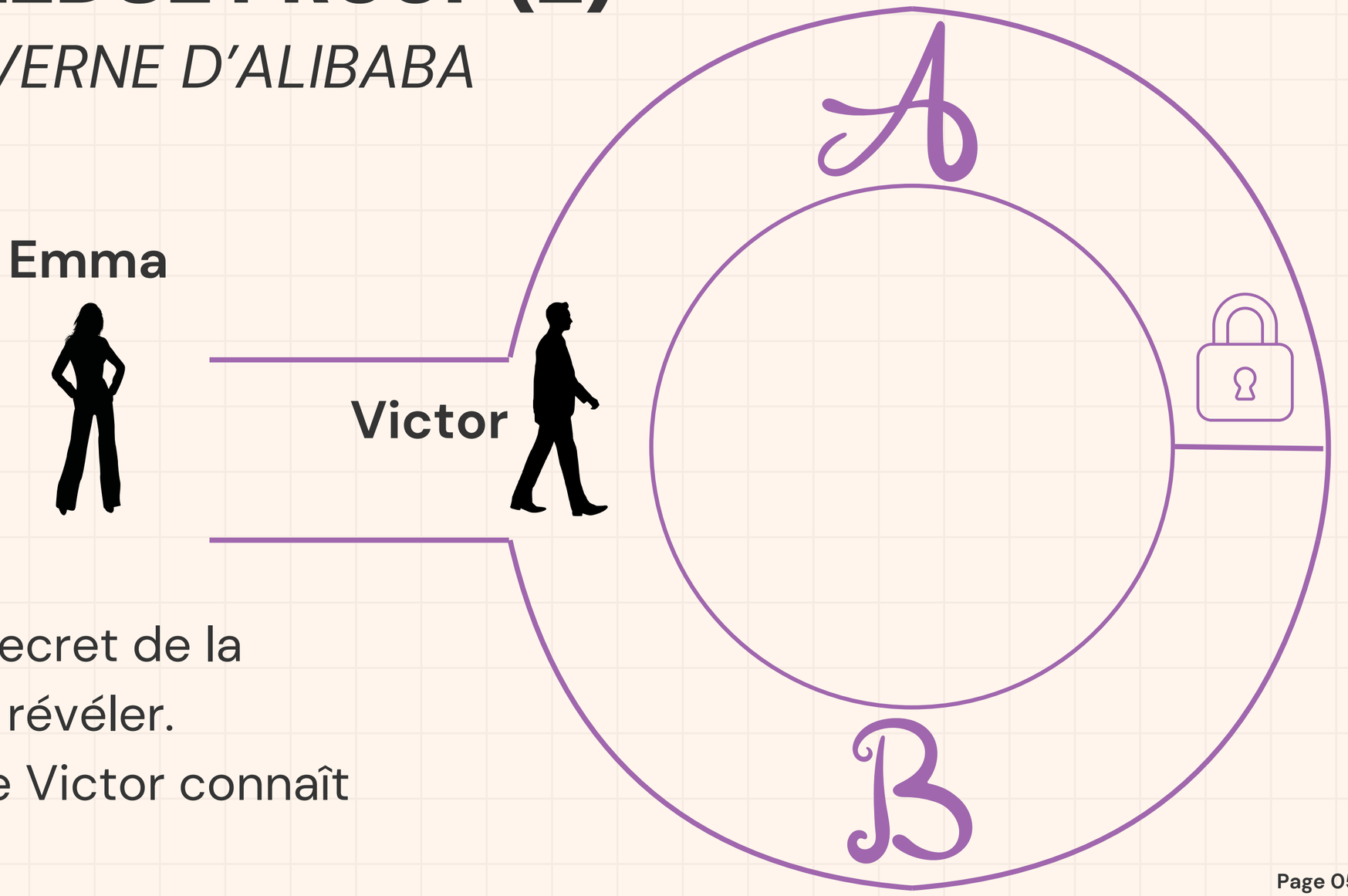
Un prouveur connaît un secret et veux le prouver au vérificateur.

Trois propriétés fondamentales :

1. Complétude : un prouveur honnête convainc un vérificateur honnête
2. Solidité : il est impossible (négligeable) de tromper le vérificateur
3. Zero-knowledge : la preuve ne révèle aucune information sur le secret

ZERO KNOWLEDGE PROOF (2)

EXEMPLE : LA CAVERNE D'ALIBABA



Victor connaît le mot secret de la porte et ne veux pas le révéler.
Emma veux vérifier que Victor connaît ce mot secret

ZERO KNOWLEDGE PROOF (3)

EXEMPLE 2 : LE LOGARITHME DISCRET

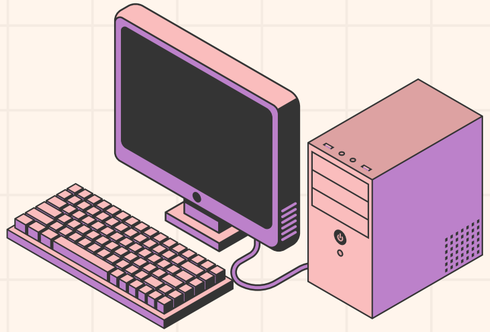
$$x = g^a \bmod p$$

→ x facile à calculer mais ‘a’ difficile à retrouver avec g et x

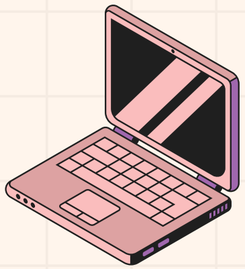
→ Moyen de se partager une clé en cryptographie ($K=g^{ab}=g^{ba}$)

$$t=g^r \rightarrow c \rightarrow z=r+a*c \rightarrow g^z = t*x^c \ ?$$

→ Procédure de preuve : engagement, défis et réponses



UTILISATION DANS LA BLOCKCHAIN

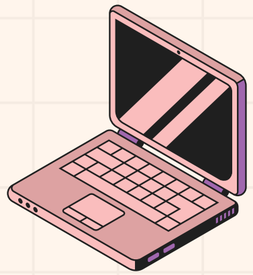


ZK-SNARKS

Problème du ZKP : communication entre le vérificateur et le prouver

Objectifs :

- Donner en une seule fois une preuve que l'on détient une information sans la révéler
- N'importe quel acteur peut vérifier instantanément que le prouveur détient bel et bien l'information qu'il prétend avoir



Lambda est critique : toute personne qui connaît lambda **peut fabriquer de fausses preuves.**
Pour éviter qu'un participant puisse tricher une tierce partie de confiance peut générer pk et vk et **garantir que lambda a été détruit.**

ALGORITHME

1 : Génération des clés – G

A partir d'un secret initial appelé **lambda** et d'un **programme C**, on génère deux clés publiques

Ces clés sont créées une seule fois

pk : clé de preuve
vk : clé de vérification
x : entrée publique

2 : Preuve – P

Le prouveur utilise **pk**, une entrée **publique x**, et un **témoin privé w** pour produire une preuve **prf** qu'il connaît w **vérifiant C(x, w) = vrai.**

3 : Vérification – V

Le vérificateur calcule **V(vk, x, prf)** et accepte si la preuve est correcte, sans apprendre le témoin.

Exemple d'usage

C(x, w) = (sha256(w) == x)
Bob génère (pk, vk).
Alice prouve qu'elle connaît s tel que sha256(s) = H sans révéler s.
Bob vérifie la preuve avec vk et accepte.

ZK SNARK - DÉTAIL ALGO

Rappel : on veut montrer que Commitment(w) = x sans révéler w

Transformation du commitment sous forme de circuit arithmétique

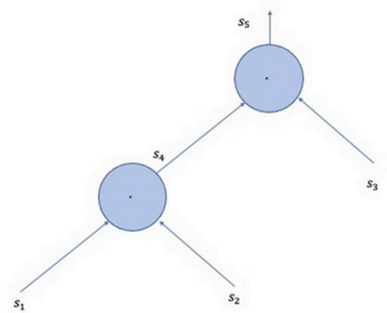


Figure 1: Arithmetic circuit for $f(s_1, s_2, s_3) = (s_1 \cdot s_2) \cdot s_3$

On veut maintenant montrer qu'on connait s :

$$s = (s_1, s_2, ..., s_n)$$

Mise en équation des contraintes du circuit :

constants $\{u_{i,q}, v_{i,q}, w_{i,q}\}_{1 \le i \le n, 1 \le q \le m}$,

$$\sum_{i=1}^{n} s_i u_{i,q} \cdot \sum_{i=1}^{n} s_i v_{i,q} = \sum_{i=1}^{n} s_i w_{i,q} \quad \forall 1 \le q \le m$$

Ces équations reviennent à montrer la divisibilité d'un polynome :

Definition 2. *(Quadratic Arithmetic Program) Pick target points $r_1, r_2, ...r_m \in \mathbb{F}_p$. Define $t(x) = \prod_{q=1}^{m}(x - r_q)$. Further, let $u_i(x), v_i(x), w_i(x)$ be degree $m-1$ polynomials such that for $1 \le i \le n, 1 \le q \le m$*

$$u_i(r_q) = u_{i,q} \quad (2)$$
$$v_i(r_q) = v_{i,q} \quad (3)$$
$$w_i(r_q) = w_{i,q} \quad (4)$$

$$\Big(\sum_{i=1}^{n} s_i u_i(x)\Big) \cdot \Big(\sum_{i=1}^{n} s_i v_i(x)\Big) - \Big(\sum_{i=1}^{n} s_i w_i(x)\Big) = h(x)t(x)$$

$$u(z) = \textstyle\sum_i s_i u_i(z), \text{ then } v(z) = \sum_i s_i v_i(z) \text{ and } w(z) = \sum_i s_i w_i(z)$$

Encodage qui respecte des propriétés mathématiques

Definition 3. *(Homomorphic Encoding) An injective homomorphism $E : \mathbb{F}_p \to G$ such that it is hard to find x given $E(x)$.*

For a cyclic group G of prime order p and multiplication as the group operation, we will use the encoding $E(a) = g^a$, for a generator g of G. This is homomorphic and injective. Moreover, by the hardness of the discrete logarithm problem, it is also hard to find x given $E(x)$.

Definition 4. *(Pairing Function) Suppose G_1, G_2, and G_T are groups of prime order p. A pairing function, or bilinear map, is a function*

$$e : G_1 \times G_2 \to G_T \quad (6)$$

such that if g, h, are generators of G_1 and G_2 respectively, then $e(g, h) \neq 1$ is a generator of G_T and $e(g^a, h^b) = e(g, h)^{ab}$.

Below, we list some basic properties about bilinear maps.

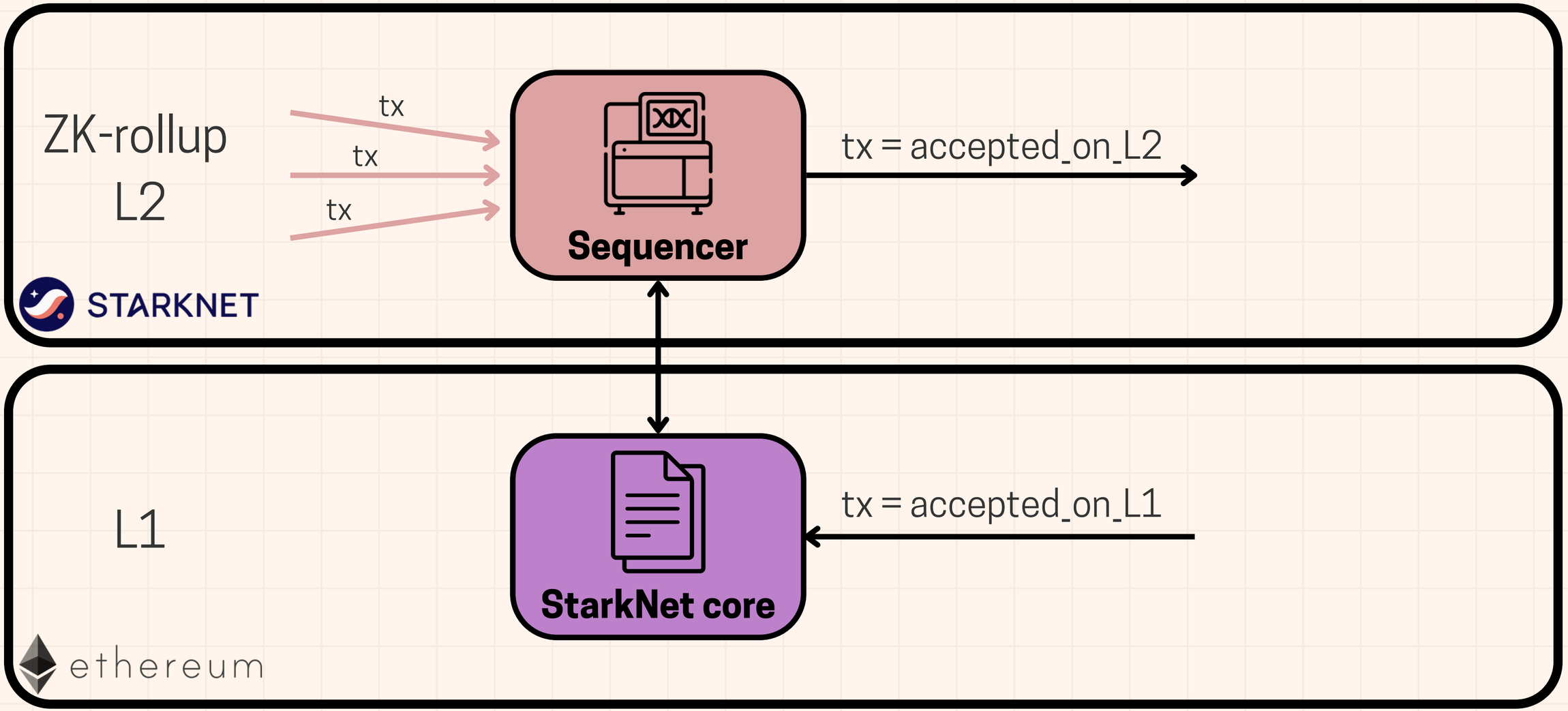
$$e(u, vw) = e(g^a, h^b h^c) = e(g, h)^{a(b+c)} = e(g^a, h^b)e(g^a, h^b) = e(u, v)e(u, w) \quad (7)$$
$$e(vw, u) = e(v, u)e(w, u) \quad (8)$$
$$e(u^x, v) = e(g, h)^{xab} = e(u, v^x) \quad (9)$$

Ce qui est finalement vérifié :

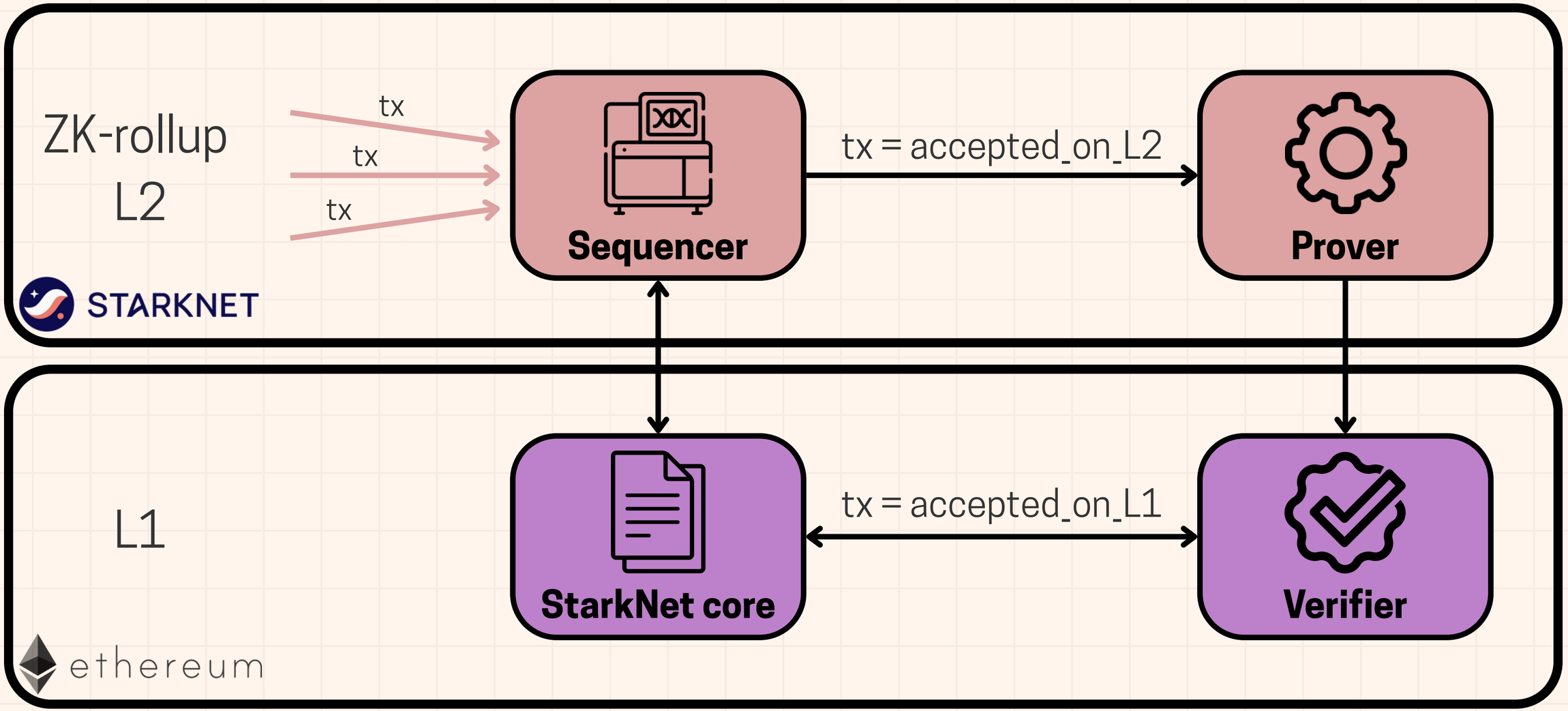
$$e(E(u(z)), E(v(z))) = e(E(w(z)), E(1)) \cdot e(E(t(z)), E(h(z)))$$

Autre vérification : u(z), v(z), w(z) sont des combinaisons linéaires valides, et les mêmes coefficients de s sont utilisés pour chaque polynome.

SCALABILITÉ DANS LA BLOCKCHAIN



SCALABILITÉ DANS LA BLOCKCHAIN



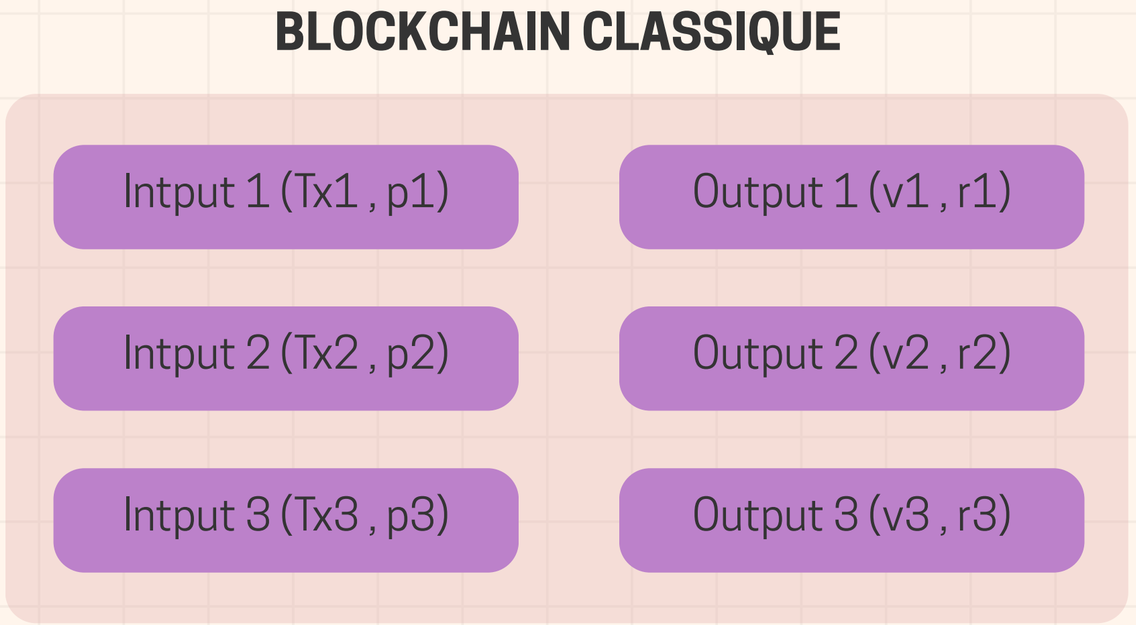
ANONYMAT DES TRANSACTIONS ZCASH (1)

1. PAS DE COINS INVENTÉS
Les nœuds vérifient que chaque Tx référencé existe.

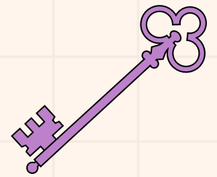
2. PAS DE DOUBLE DÉPENSE
Un TX ne peut être dépensé qu'une seule fois.

3. PAS DE DÉPENSE NON AUTORISÉE
Signature numérique obligatoire avec la clé privée du propriétaire.

4. PAS DE CRÉATION NI DESTRUCTION DE VALEUR
Les nœuds vérifient : somme(inputs) = somme(outputs).



ANONYMAT DES TRANSACTIONS ZCASH (2)



Clé de Transmission: TK

Clé de visualisation: VK **Clé de dépense : SK**

NOTE

- Destinataire
- Valeur
- Aléatoire

TK

CIPHERTEXT

skjdngojjbdsgob
sdogbosdbg

- Intput 1 → Output 1 (C1, cm1, pi1)
- Intput 2 → Output 2 (C2, cm2, pi2)
- Intput 3 → Output 3 (C3, cm3, pi3)

Chaque note est utilisée une unique fois

C_m = valeur·G + r·H (Pedersen).

Σ C_m(spends) – Σ C_m(outputs) = Cm(0)

homomorphisme additif

ZK-SNARK - Π

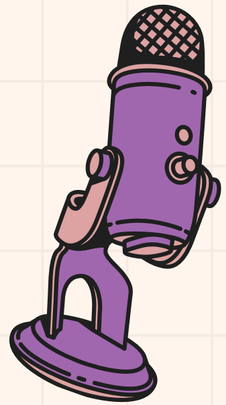
Il a correctement chiffré le note pour le destinataire
Le commitment respecte la structure
Le commitment est bien formé

Provenance ? Double dépense ? Dépense des autres ?

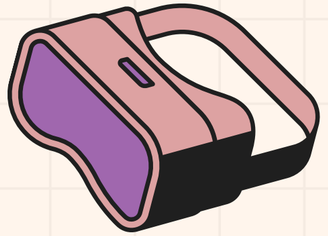
Sapling/Orchard, Merkle path, nullifiers, signatures aléatoires...

CODE

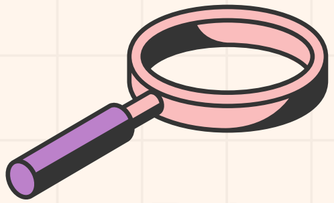
DEMO



D’AUTRES APPLICATIONS ?

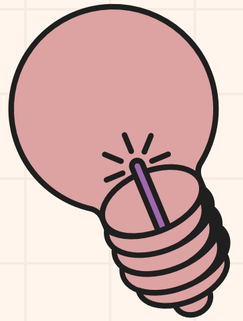


Systèmes d’authentification



Prouver son identité à quelqu’un sans lui la révéler (site internet, sécurité physique…)

Désarmement nucléaire



Prouver, lors de négociations de désarmement nucléaire ou de contrôle, la nature d’un objet sans lui révéler d’informations sensible (structures…) (Université de Princeton 2016)

Encodage homomorphique pour le ML

Permet de chiffrer des données tout en conservant certaines propriétés mathématiques, ce qui permet de :

- faire des calculs ou entraînements de modèles ML directement sur les données chiffrées.
- déchiffrer uniquement le résultat final, souvent agrégé ou résumé, sans jamais révéler les données individuelles.

BIBLIOGRAPHIE (1)

zk-SNARKs: A Gentle Introduction Anca Nitulescu - ENS ULM	https://www.di.ens.fr/~nitulesc/files/Survey-SNARKs.pdf
Introduction to zk-SNARKs - Consensys.io	https://consensys.io/blog/introduction-to-zk-snarks
A gentle introduction to shielded transactions	https://redshiftzero.com/post/utxo_privacy
Groth16 Explained	https://rareskills.io/post/groth16
Zcash - Wikipédia	https://en.wikipedia.org/wiki/Zcash
Non interactive zéro knowledge proof - Wikipédia	https://en.wikipedia.org/wiki/Non-interactive_zero-knowledge_proof
Preuve à divulgation nulle de connaissance - Wikipédia	https://fr.wikipedia.org/wiki/Preuve_%C3%A0_divulgation_nulle_de_connaissance

BIBLIOGRAPHIE (2)

What are Zero-Knowledge Rollups (ZK-rollups)? https://www.alchemy.com/blog/zero-knowledge-rollups?

Rollups Zero-knowledge (ZK) https://ethereum.org/fr/developers/docs/scaling/zk-rollups

A Review of zk-SNARKs, Thomas Chen, Hui Lu, Teeramet Kunpittaya, and Alan Luo§ https://arxiv.org/pdf/2202.06877

Cryptographie Homomorphe https://www.unilim.fr/pages_perso/deneuville/files/rapport_M2_09_2012.pdf

CNIL, On a testé le chiffrement homomorphe ! https://linc.cnil.fr/on-a-teste-le-chiffrement-homomorphe