# ÉTHERUM : UN SÉCURISE DÉCENTRALISÉ GÉNÉRALISÉ TRANSACTION REGISTRE

**BERLIN VERSION 630ed56 – 2022-02-13**

DR. GAVIN BOIS FONDATEUR, ÉTHERUM & PARITÉ

[GAVIN@PARITY.IO](mailto:GAVIN@PARITY.IO)

Abstrait. Le paradigme de la blockchain, lorsqu'il est associé à des transactions sécurisées par cryptographie, a démontré son utilité à travers un certain nombre de projets, Bitcoin étant l'un des plus remarquables. Chacun de ces projets peut être considéré comme une application simple sur une ressource de calcul décentralisée, mais unique. Nous pouvons appeler ce paradigme un modèle transactionnel singleton machine avec état partagé.

Ethereum implémente ce paradigme de manière généralisée. En outre, il fournit une pluralité de telles ressources, chaque avec un distinct Etat et en fonctionnement code mais capable pour interagir par un passage de message cadre avec autres. Nous discuter son motif, la mise en oeuvre questions, la Opportunités ce fournit et la futur obstacles nous envisager.

1. Introduction

Avec omniprésent l'Internet Connexions dans plus des endroits du monde, la transmission mondiale de l'information est devenue incroyablement bon marché. Des mouvements ancrés dans la technologie comme Bit- coin ont démontré par la puissance du défaut, les mécanismes de consensus et le respect volontaire du social Contrat, que ce est possible pour utiliser la l'Internet pour Fabriquer un système de transfert de valeur décentralisé et partageable à travers le monde et pratiquement gratuit à utiliser. Ce système peut être considéré comme une version très spécialisée d'un cryptographe- machine d'état basée sur les transactions et sécurisée. Suivi des systèmes tels que Namecoin ont adapté cette « monnaie l'application" de la technologie dans d'autres applications bien que plutôt simpliste ceux.

Ethereum est un projet qui tente de construire la génération technologie généralisée; technologie sur laquelle toutes les transactions des concepts de machine à états basés peuvent être construits. De plus il vise à fournir au développeur final une solution étroitement intégrée système de bout en bout pour créer des logiciels sur un paradigme de calcul exploré dans le courant dominant : une approche de confiance objet Messagerie calculer cadre.

* 1. **Conduite Facteurs.** Là sont beaucoup objectifs de cette projet; l'un des principaux objectifs est de faciliter les transactions entre des personnes consentantes qui n'auraient autrement aucun moyen se faire confiance. Cela peut être dû à la géographie séparation, interfaçage difficulté, ou alors peut-être la incompatible- capacité, incompétence, réticence, dépense, incertitude, les inconvénients ou la corruption des systèmes juridiques existants. Par en précisant un changement d'état système par un riche et unam- langage bigu, et en outre l'architecture d'un système de telle sorte que nous pouvons raisonnablement nous attendre à ce qu'un accord être Donc forcée de manière autonome, nous pouvez fournir un moyens pour cette finir.

Les transactions dans ce système proposé auraient plusieurs les attributs ne pas souvent trouvé dans la réel monde. Le incorruptible- capacité de jugement, souvent difficile à trouver, vient naturellement d'un interprète algorithmique désintéressé. Transparence, ou être capable de voir exactement comment un état ou un jugement est venu à propos du journal des transactions et des règles ou des instructions codes, ne se produit jamais parfaitement dans les systèmes humains puisque Naturel Langue est nécessairement vague, information

fait souvent défaut, et les vieux préjugés sont difficiles à secouer.

Globalement, nous souhaitons fournir un système tel que les utilisateurs peut être garanti que, peu importe avec quel autre individu des particuliers, systèmes ou alors organisations elles ou ils interagir, elles ou ils pouvez le faire avec une confiance absolue dans les résultats possibles et comment ceux résultats pourrait viens à propos.

* 1. **Précédent travail.** [Buterin [2013a] a d'](#_bookmark73) abord proposé la noyau de ce travail fin novembre 2013. Bien que maintenant évolué à bien des égards, la fonctionnalité clé d'un bloc- chaîne avec un langage Turing-complet et une la capacité de stockage inter-transaction illimitée reste modifié.

[Dwork et Naor [1992]](#_bookmark91) ont fourni les premiers travaux sur la utilisation d'une preuve cryptographique des dépenses de calcul ture (“preuve de travail”) comme moyen de transmettre une valeur signaler sur Internet. Le signal de valeur a été utilisé ici comme un Spam dissuasion mécanisme plutôt que quelconque type de devise, mais critique démontré la potentiel pour un canal de données de base pour véhiculer un *signal économique fort* , permettant à un récepteur de faire une assertion physique sans avoir à compter sur la *confiance* . [Arrière](#_bookmark69) [[2002]](#_bookmark69) a produit plus tard un système dans un similaire veine.

Le premier exemple d'utilisation de la preuve de travail comme signal économique fort pour sécuriser une monnaie a été par [Vish-](#_bookmark107) [numurthy et al.](#_bookmark107) [[2003].](#_bookmark107) Dans ce cas, le jeton était utilisé pour contrôler le commerce de fichiers peer-to-peer, fournissant « consommateurs » avec la possibilité d'effectuer des micro-paiements pour « fournisseurs » pour leurs services. Le modèle de sécurité offert par la preuve de travail a été complétée par des signatures numériques et un registre afin de s'assurer que le dossier historique ne pouvait pas être corrompu et que les acteurs malveillants ne pouvaient pas falsifier le paiement ou se plaindre injustement de la prestation de services éry. Cinq ans plus tard, [N akamoto [2008] a](#_bookmark87) introduit un autre un tel jeton de valeur sécurisé par preuve de travail, un peu plus large dans portée. Le fruit de ce projet, Bitcoin, est devenu le premier largement adopté global décentralisé transaction registre.

D'autres projets construits sur le succès de Bitcoin ; les altcoins introduit de nombreuses autres monnaies par altération au protocole. Certains des plus connus sont Litecoin et Primecoin, discuté par [Sprankel [2013].](#_bookmark101) Autres projets recherché pour prendre la coeur valeur teneur mécanisme de la pro- tocol et réutiliser ce; [Arron [2012]](#_bookmark70) discute, pour Exemple,

1

ÉTHERUM : UN SÉCURISE DÉCENTRALISÉ GÉNÉRALISÉ TRANSACTION COMPTABLE **BERLIN VERSION** 2

le projet Namecoin qui vise à fournir une solution décentralisée résolution de nom système.

D'autres projets visent toujours à s'appuyer sur le réseau Bitcoin. travail lui-même, tirant parti de la grande quantité de valeur placée dans le système et la grande quantité de calcul qui va dans le mécanisme de consensus. Le projet Mastercoin, proposé pour la première fois par [Willett](#_bookmark109) [[2013],](#_bookmark109) vise à construire un monde plus riche protocole impliquant de nombreuses fonctionnalités supplémentaires de haut niveau sur top du protocole Bitcoin grâce à l'utilisation d'un numéro de pièces auxiliaires au protocole de base. Le Coloré Pièces de monnaie

lui-même - ce serait beaucoup trop grand). Ils ponctuent également le série de transactions avec des incitations pour les nœuds à *exploiter* . Cette l'incitation a lieu comme une fonction de transition d'état, ajouter valeur pour un nommé Compte.

L'exploitation minière est le processus qui consiste à consacrer des efforts (travailler) à renforcer une série de transactions (un bloc) par rapport à toute autre potentiel concurrent bloc. Ce est atteint Merci pour un preuve cryptographiquement sécurisée. Ce schéma est connu sous le nom de preuve de travail et est discuté dans détail dans section [11.5.](#_bookmark68)

Officiellement, nous étendre pour:

projet, proposé par [Rosenf eld et al. [2012],](#_bookmark89) prend une approche similaire mais stratégie plus simplifiée, embellissant les règles d'un transaction afin de briser la fongibilité des Bitcoins base devise et Autoriser la création et suivi de jetons par un spécial "chroma-wallet"-conscient du protocole pièce de

(2)

(3)

(4)

***σ*** *t* +1 ≡ [Π (](#_bookmark67) ***σ*** *t , B* )

*B* ≡ ( *...,* ( *T* 0 *, T* 1 *, ...* ) *, ...* )

Π ( ***σ*** *, B* ) ≡ [Ω(](#_bookmark65) *B,* [Υ ( Υ (](#_bookmark0) ***σ*** *, T* 0 ) *, T* 1 ) *...* )

Où [Ω](#_bookmark65) est la bloc-finalisation Etat passage fonction-

Logiciel.

Supplémentaire travail possède a été Fini dans la région avec Jeter- la fondation de la décentralisation ; Ripple, discuté par [Boutellier et Heinzen [2014],](#_bookmark74) a cherché à créer une « éré » système pour devise échanger, effectivement créer un nouveau système de compensation financière. Il a démontré que des gains d'efficacité élevés peuvent être réalisés si la décentralisation prémisse est mis au rebut.

Les premiers travaux sur les contrats intelligents ont été réalisés par [Szabo](#_bookmark103) [[1997]](#_bookmark103) et [Meunier [1997].](#_bookmark86) Environ la années 1990 ce devenu dégager que l'application algorithmique des accords pourrait devenir un important Obliger dans Humain la coopération. Mais non spécifique système a été proposé pour mettre en œuvre un tel système, il a été proposé que l'avenir du droit serait fortement affecté par de tels systèmes. Dans cette optique, Ethereum peut être considéré comme un général la mise en oeuvre de tel un *crypto-loi* système.

Pour un liste de termes utilisé dans cette papier, référer pour Appen- dix [UN.](#_bookmark111)

1. Le Chaîne de blocs Paradigme

Ethereum, pris comme un entier, pouvez être vu comme un basé sur les transactions Etat machine: nous commencer avec un gen- esis état et exécute progressivement des transactions pour transformer dans un état actuel. C'est cet état actuel que nous J'accepte comme la canonique "version" de la monde de Ethereum. L'État peut inclure de telles informations comme compte bal- ances, réputations, confiance arrangements, Les données se rapportant aux informations du monde physique ; bref, n'importe quoi qui peut actuellement être représenté par un ordinateur est admis- possible. Les transactions représentent donc un arc valide entre deux États; la partie 'valide' est importante—il en existe beaucoup plus invalide Etat changements que valide Etat changements. Invalide Etat les changements peuvent, par exemple, être des choses telles que la réduction d'un compte équilibre sans une augmentation égale et opposée ailleurs. UN valide Etat passage est une qui vient à propos par un transaction. Officiellement:

tion (une fonction qui récompense une partie désignée); [*B*](#_bookmark19) est-ce bloc, qui comprend une série de transactions entre certains autre Composants; et [Π](#_bookmark67) est la au niveau du bloc transition d'état une fonction.

C'est la base du paradigme de la blockchain, un modèle qui forme l'épine dorsale non seulement d'Ethereum, mais tout décentralisé basée sur le consensus transaction systèmes pour Date.

* 1. **Valeur.** Afin d'encourager le calcul au sein du réseau, il doit y avoir une méthode convenue pour la transmission valeur ting. Pour résoudre ce problème, Ethereum a une capacité intrinsèque devise, Éther, connu aussi comme EPF et parfois référé pour par \_ la Vieille Anglais ¯ D. Le le plus petit sous- dénomination d' Ether , et donc celui dans lequel toutes les valeurs entières de la monnaie sont comptées, est le Wei. Un éther est défini comme étant 10 18 Wei. Il existe d'autres sous-dénominations de Éther:

Multiplicateur Nom

10 0 Wei

10 12  Szabo

10 15  Finney

10 18  Éther

Tout au long de la cadeau travail, quelconque référence pour valeur, dans le cadre d'Ether, de devise, d'un solde ou d'un paiement, devrait être assumé pour être dénombré dans Wei.

* 1. **Quelle Histoire ?** Puisque le système est décentralisé et toutes les parties ont la possibilité de créer un nouveau bloc sur quelques plus âgée préexistant bloc, la résultant structure est nécessairement un arbre de blocs. Afin de former un consensus quant à quel chemin, de la racine [(le bloc de genèse](#_bookmark153) ) à la feuille (le bloc contenant les transactions les plus récentes) via cette arbre structure, connu comme la chaîne de blocs, là doit être un schéma convenu. Si jamais il y a un désaccord entre les nœuds quant au chemin de la racine à la feuille dans le bloc arbre est la 'meilleur' chaîne de blocs, alors un *fourchette* se produit.

(1) ***σ***

*j* +1

≡ Υ( ***σ*** *t , J* )

Cette voudrais moyenne que passé un donné indiquer dans temps (bloc), plusieurs États de la système mai coexister: quelques nœuds être-

où Υ est la Ethereum Etat passage une fonction. Dans Ethereum, Υ, ensemble avec ***σ*** sont considérablement Suite pou- énervant que quelconque existant comparable système; Υ permet com- composants pour effectuer des calculs arbitraires, tandis que ***σ*** permet Composants pour boutique arbitraire Etat entre transactions. Les transactions sont regroupées en blocs ; les blocs sont enchaînés ensemble en utilisant un cryptographique hacher comme un moyens de référer- ence. Blocs une fonction comme un journal, enregistrement un séries de transactions ensemble avec la précédent bloc et un iden- tifier pour la final Etat (mais faire ne pas boutique la final Etat

croire une bloc pour contenir la canonique transactions, d'autres nœuds croyant qu'un autre bloc est canonique, contenant potentiellement des produits radicalement différents ou incompatibles transactions. Ceci doit être évité à tout prix car l'ONU- la certitude qui s'ensuivrait tuerait probablement toute confiance dans la entier système.

Le schéma que nous utilisons pour générer un consensus est un version simplifiée du protocole GHOST introduite par [Sompolinsky et Zohar [2013].](#_bookmark100) Ce processus est décrit dans détail dans section [dix.](#_bookmark61)

ÉTHERUM : UN SÉCURISE DÉCENTRALISÉ GÉNÉRALISÉ TRANSACTION COMPTABLE **BERLIN VERSION** 3

Parfois, un chemin suit un nouveau protocole d'un hauteur particulière (numéro de bloc). Ce document décrit une version de la protocole, à savoir le *Berlin* version défini par [Beiko et al. [2021b].](#_bookmark72) Afin de suivre en arrière le historique d'un chemin, il faut référencer plusieurs versions de ce document. Voici les numéros de bloc du protocole mises à jour sur la Ethereum principale réseau:

|  |  |
| --- | --- |
| Nom | Première Bloc Numéro |
| Ferme *F* | 1150000 |
| *F* TangerineSifflet | 2463000 |
| *F* SpuriousDragon | 2675000 |
| *Fa* Byzance | 4370000 |
| *F* Constantinople | 7280000 |
| *Saint* -Pétersbourg | 7280000 |
| *F* Istanbul | 9069000 |
| *F* MuirGlacier | 9200000 |
| *F* Berlin | 12244000 |
| *F* Londres | 12965000 |
| Flèche *FGlacier* | 13773000 |

Parfois, les acteurs ne sont pas d'accord sur un changement de protocole, et une fourche permanente se produit. Afin de distinguer être- entre les blockchains divergentes, EIP-155 par [Buterin](#_bookmark79) [[2016b]](#_bookmark79) introduit la concept de chaîne IDENTIFIANT, qui nous dénoter par *β* .

Pour la Ethereum principale réseau

(5) *β* = 1

1. Conventions

Nous utilisons un certain nombre de conventions typographiques pour la notation formelle, dont certaines sont assez particulières à la cadeau travail:

Les deux ensembles de valeurs d'état hautement structurées de « niveau supérieur » ues, sont indiqués par des lettres grecques minuscules en gras. Ils tombent dans ceux d'état-monde, qui sont notés ***σ*** (ou un une variante là-dessus) et ceux de état de la machine, ***µ*** .

Les fonctions en fonctionnement sur très structuré valeurs sont dénoté avec un majuscule grec lettre, par exemple [Υ,](#_bookmark0) la Ethereum Etat passage une fonction.

Pour la plupart des fonctions, une lettre majuscule est utilisée, par exemple *C* , la fonction de coût général. Ceux-ci peuvent être souscrits à désignent des variantes spécialisées, par exemple [*C* SSTORE](#_bookmark146) , la fonction de coût tion pour la [STORE](#_bookmark145) opération. Pour spécialisé et peut-être

fonctions définies en externe, nous pouvons formater comme une machine à écrire texte, par exemple la Keccak-256 hacher une fonction (comme par version 3 de la gagnant entrée pour la SHA-3 concours par [Bertoni](#_bookmark75) [et Al. [2011],](#_bookmark75) plutôt que la final SHA-3 spécification), est

noté KEC (et généralement appelé Keccak ordinaire). Aussi, KEC512 fait référence pour la Keccak-512 hacher une fonction.

Tuples sont typiquement dénoté avec un majuscule lettre,

par exemple *T* , est utilisé pour désigner une transaction Ethereum. Cette symbole mai, si par conséquent défini, être indice pour référer pour un individuel composant, par exemple [*T* n](#_bookmark40) , dénote la nonce de mentionné transaction. Le formulaire de la indice est utilisé pour

désigner son type ; par exemple, les indices majuscules font référence aux tuples avec inscriptible Composants.

Scalaires et séquences d'octets de taille fixe (ou, synonyme, tableaux) sont dénoté avec un Ordinaire minuscule lettre, par exemple *n* est utilisé dans le document pour désigner une [transaction nonce.](#_bookmark40) Ceux avec un notamment spécial sens mai être Grec,

par exemple *δ* , le nombre d'éléments requis sur la pile pour un donné opération.

Les séquences de longueur arbitraire sont généralement désignées par un lettre minuscule en gras, par exemple **o** est utilisé pour désigner l'octet séquence donnée comme données de sortie d'un appel de message. Pour valeurs particulièrement importantes, une lettre majuscule en gras peut être utilisé.

Partout, nous supposons que les scalaires sont des intégraux non négatifs. Gers et Donc appartenir pour la Positionner N. \_ Le Positionner de tout octet séquences est B , officiellement défini dans annexe [B](#_bookmark112) Si tel un ensemble de séquences est restreint à celles d'un longueur, ce est dénoté avec un indice, Donc la Positionner de tout

les séquences d'octets de longueur 32 sont nommées B 32 et l'ensemble des tout non négatif entiers plus petite que 2 256 est nommé N 256 . Cette est officiellement défini dans section [4.3.](#_bookmark19)

Les crochets sont utilisés pour indexer et référencer composants individuels ou sous-séquences de séquences, par exemple ***µ* s** [0] désigne le premier élément de la pile de la machine. Pour sous-séquences, les ellipses sont utilisées pour spécifie le prévu gamme, pour inclure des éléments aux deux limites, par exemple ***µ* m** [0 *..* 31] dénote la première 32 éléments de la Machines Mémoire.

Dans le cas de l'état global ***σ*** , qui est une séquence de comptes, eux-mêmes tuples, les crochets sont utilisés pour référence un individuel Compte.

Lorsque nous considérons des variantes de valeurs existantes, nous suivons la règle que dans un donné portée pour définition, si nous supposons que la valeur 'd'entrée' non modifiée soit désignée par la espace réservé alors la modifié et utilisable valeur est

dénoté comme , et intermédiaire valeurs voudrais être ∗ , ∗∗

&c. Lors d'occasions très particulières, afin de maximiser lisibilité et seulement si le sens est sans ambiguïté, nous pouvons utiliser des indices alphanumériques pour désigner les valeurs intermédiaires, particulièrement ceux de particulier Remarque.

Lors de l'examen de l'utilisation des fonctions existantes, compte tenu d'un fonction *f* , la fonction [*f* ∗](#_bookmark27) désigne un élément similaire version du mappage de fonction à la place entre les séquences. Ce est officiellement défini dans section [4.3.](#_bookmark19)

Nous définissons un certain nombre de fonctions utiles tout au long. Une de la Suite commun est *.€* , qui \_ \_ e v alue pour la dernière article dans la donné séquence:

(6) *.€* ( **x** ) ≡ **x** [ ll **x** ll − 1]

1. Blocs, État et Transactions

Ayant introduit la basique notions derrière Ethereum, nous discuterons de la signification d'une transaction, d'un bloc et la Etat dans Suite détail.

* 1. **Monde État.** Le monde Etat ( *état* ), est un carte- ping entre les adresses (identifiants 160 bits) et le compte états (une structure de données sérialisée en tant que RLP, voir l'annexe [B).](#_bookmark112) Mais ne pas stockée sur la chaîne de blocs, ce est assumé que l'implémentation maintiendra cette cartographie dans une modi- fié Merkle Patricia arbre ( *trie* , voir l'annexe [D).](#_bookmark117) L'essai nécessite une base de données simple qui maintient une carte- ping de tableaux d'octets vers des tableaux d'octets ; nous nommons ce sous-jacent base de données la base de données d'état. Cela a un certain nombre d'avantages; premièrement le noeud racine de cette structure est cryptographiquement dépendant sur tout interne Les données et comme tel son hacher pouvez être utilisé comme une identité sécurisée pour l'ensemble de l'état du système. Deuxièmement, étant une structure de données immuable, elle permet à tout précédent Etat (à qui racine hacher est connu) pour être rappelé en modifiant simplement le hachage racine en conséquence. Depuis que nous stocker tous ces hachages racine dans la blockchain, nous sommes en mesure de trivialement revenir pour vieille États.

ÉTHERUM : UN SÉCURISE DÉCENTRALISÉ GÉNÉRALISÉ TRANSACTION COMPTABLE **BERLIN VERSION** 4

L'état du compte, ***σ*** [ *a* ], comprend les quatre des champs:

**nonce :** une valeur scalaire égale au nombre de trans- Actions expédié depuis cette adresse ou alors, dans la Cas de comptes avec code associé, le nombre de les créations de contrats effectuées par ce compte. Pour ac- compte de l'adresse *a* dans l'état ***σ*** , ce serait pour- mally dénoté ***σ*** [ *une* ] n .

**solde:** UN scalaire valeur égal pour la Numéro de Wei

possédé \_ par \_ cette adresse. Formellement \_ dénoté ***σ*** [ *une* ] b .

**racine de stockage :** Un hachage de 256 bits du nœud racine d'un arbre Merkle Patricia qui encode la configuration de stockage tentes du compte (un mappage entre 256 bits entier valeurs), codé dans la essayer comme un cartographie à partir du hachage Keccak 256 bits de l'entier 256 bits clés aux valeurs entières 256 bits codées RLP. Le hacher est officiellement dénoté ***σ*** [ *une* ] s .

**hachage de code :** Le hacher de la EVM code de cette

compte—ce est la code que obtient réalisé devrait cette adresse recevoir un message appel; ce est immuable et Donc, contrairement à tout autre des champs, ne peux pas être modifié après chantier. Tous ces fragments de code sont contenus dans la base de données de l'État sous leur corres- hachages répondants pour une récupération ultérieure. Ce hachage est officiellement dénoté ***σ*** [ *une* ] c , et Donc la code mai être

dénoté comme **b** , donné que CEC ( **b** ) = ***σ*** [ *une* ] c .

Puisque nous souhaitons généralement ne pas nous référer à la racine du trie hachage mais à l'ensemble sous-jacent de paires clé/valeur stockées dans, nous définir un pratique équivalence:

1. TRIE *L* ∗ je ( ***σ*** [ *une* ] **s** ) ≡ ***σ*** [ *une* ] s

La fonction de réduction pour l'ensemble de paires clé/valeur dans la essayer, *L* ∗ je , est d' amende comme la élément en t transformation de la base une fonction *L* je , donné comme:

1. *L* je ( *k, v* ) ≡ KEC ( *k* ) *,* RLP ( *v* ) où:

(9) *k* ∈ B 32 ∧ *v* ∈ N

Ce devoir être entendu que ***σ*** [ *une* ] **s** est ne pas un 'physique' membre du compte et ne contribue pas à son utilisation ultérieure sérialisation.

Si la **codeHash** domaine est la Keccak-256 hacher de la vider corde, c'est à dire ***σ*** [ *une* ] c = KEC () , alors la nœud représente un Facile Compte, parfois référé pour comme un "hors contrat" Compte.

Ainsi nous mai définir un état-monde s'effondrer une fonction *L* S :

(10) *L* S ( ***σ*** ) ≡ { *p* ( *une* ) : ***σ*** [ *une* ] ∅ }

où

(11) *p* ( *une* ) ≡ KEC ( *une* ) *,* RLP ( ***σ*** [ *une* ] n *,* ***σ*** [ *une* ] b *,* ***σ*** [ *une* ] s *,* ***σ*** [ *une* ] c )

Cette une fonction, *L* S , est utilisé aux côtés de la essayer une fonction pour fournir une courte identité (hachage) de l'état du monde. Nous présumer:

(12) ∀ *une* : ***σ*** [ *une* ] = ∅ ∨ ( *une* ∈ B 20 ∧ *v* ( ***σ*** [ *une* ])) où *v* est la Compte validité une fonction:

(13 ) *v* ( *x* ) ≡ *xn* \_ ∈ N 25 6 ∧ *x* b ∈ N 25 6 ∧ *x* s ∈ B 3 2 ∧ *x* c ∈ B 32

Un Compte est *vider* lorsque ce possède non code, zéro nonce et zéro solde:

(14)

VIDE ( ***σ*** *, une* ) ≡ ***σ*** [ *une* ] c = KEC () ∧ ***σ*** [ *une* ] n = 0 ∧ ***σ*** [ *une* ] b = 0

Même les contrats précompilés appelables peuvent avoir un vide état du compte. C'est parce que leurs états de compte ne habituellement contenir la code décrivant son comportement.

Un compte est *mort* lorsque son état de compte est inexistant ou alors vider:

(15) MORT ( ***σ*** *, un* ) ≡ ***σ*** [ *une* ] = ∅ ∨ VIDE ( ***σ*** *, un* )

* 1. **La transaction.** Une transaction (formellement, *T* ) est un instruction unique signée cryptographiquement construite par un acteur de cinéma extérieurement pour la portée de Ethereum. Le expéditeur d'une transaction ne peut être un contrat. Alors qu'on suppose que l'acteur externe ultime sera de nature humaine, des outils logiciels seront utilisés pour sa construction et sa diffusion. pays [1](#_bookmark10) . EIP-2718 par [Zoltu [2020]](#_bookmark110) introduit la notion de différents types de transactions. À partir de la version berlinoise de le protocole, il existe deux types de transactions : 0 (hérité) et 1 (EIP-2930 par [Butérine et Suède [2020b]).](#_bookmark80) Plus loin, il existe deux sous-types de transactions : celles qui résultent dans les appels de message et ceux qui entraînent la création de nouveaux comptes avec code associé (appelé officieusement "création de contrat"). Tous les types de transactions spécifient un nombre de commun des champs:

**taper:** EIP-2718 transaction taper; officiellement *Tx* . \_

**nonce :** UN scalaire valeur égal pour la Numéro de trans- Actions expédié par la expéditeur; officiellement *Tn* . \_

**prix du gaz :** UN scalaire valeur égal pour la Numéro de

Wei à payer par unité de *gaz* pour tous les calculs les frais occasionnés par l'exécution de la présente transaction; officiellement *T* p .

**Limite de gaz :** UN scalaire valeur égal pour la maximum

quantité de gaz qui doit être utilisée dans l'exécution cette opération. Ceci est payé d'avance, avant tout le calcul est fait et ne peut pas être augmenté plus tard; officiellement *Tg* . \_

**pour:** Le 160 bits adresse de la message appels recette-

ent ou, pour une opération de création de contrat, ∅ , utilisé ici pour dénoter la seul membre de B 0 ; officiellement *T* t .

**valeur:** UN scalaire valeur égal pour la Numéro de Wei pour

être transféré pour la message appels destinataire ou alors, en cas de création de contrat, en tant que dotation pour la nouvellement créé Compte; officiellement *Tv* . \_

**r, s :** Valeurs correspondant pour la Signature de la

transaction et utilisé pour déterminer l'expéditeur de la transaction; formellement *T* r et *Ts* . \_ C'est ex- pané dans annexe [F.](#_bookmark126)

EIP-2930 (taper 1) transactions aussi avoir:

**liste d'accès :** Liste des entrées d'accès à réchauffer ; pour- mally *TA* . **\_** Chaque accéder liste entrée *E* est un tuple de un Compte adresse et un liste de stockage clés:

*E* ≡ ( *E* a *, E* **s** ).

**ID de chaîne :** ID de chaîne ; formellement *T* c . Doit être égal à la réseau chaîne IDENTIFIANT [*β*](#_bookmark1) .

**yParité :** Signature Oui parité; officiellement *T* y .

1 Notamment, ces « outils » pourraient finalement devenir si causalement éloignés de leur initiation basée sur l'homme - ou les humains pourraient devenir si éloignés causalement neutres - qu'il pourrait y avoir un point où ils pourraient à juste titre être considérés comme des agents autonomes. par exemple, les contrats peuvent offrir des primes à humains pour être expédié transactions pour lancer leur exécution.

ÉTHERUM : UN SÉCURISE DÉCENTRALISÉ GÉNÉRALISÉ TRANSACTION COMPTABLE **BERLIN VERSION** 5

Les transactions héritées n'ont pas de liste d' **accès** ( *T* **A** = ()), tandis que **chainId** et **yParity** pour les transactions héritées sont combiné dans un Célibataire valeur:

**w :** un scalaire valeur codage Y parité et possibilité ID de chaîne bly ; formellement *T* w . *T* w = 27 + *T* y ou *T* w = 2 *β* +35+ *T* y (voir EIP-155 par [Butérine [2016b]).](#_bookmark79)

En outre, un Contrat création transaction (indépendamment qu'il s'agisse héritage ou alors EIP-2930) contient:

**initialisation :** Un illimité Taille octet déployer en précisant la Code EVM pour la procédure d'initialisation du compte, officiellement *T* **je** .

**init** est un Code EVM fragment; ce Retour la **corps** , un deuxième fragment de code qui s'exécute à chaque fois que le compte reçoit un message d'appel (soit via un trans- action ou due à l'exécution interne de code). **init** est exécuté une seule fois lors de la création du compte et est supprimé immédiatement après.

Dans contraste, un message appel transaction contient:

**Les données:** Un tableau d'octets de taille illimitée spécifiant saisir Les données de la message appel, officiellement *Td* . **\_**

annexe [F](#_bookmark126) spécifie la fonction, *S* , qui mappe trans- actions à l'expéditeur, et passe par l'ECDSA de la courbe SECP-256k1, en utilisant le hash de la transaction (excepté la dernier Trois Signature des champs) comme la données pour signe. Pour la cadeau nous simplement affirmer que la expéditeur de un donné transaction *J* pouvez être représentée avec *S* ( *T* ).

(16)

f

( *T* n *, T* p *, T* g *, T* t *, Télé ,* \_**p** *, T* w *, T* r *, T* s ) si *T* x = 0

*L*T(*T* ) ≡

(*T , T , T , T , T , T ,* **p***, T*

*, T , T , T* ) if *T*

= 1

**hachage parent :** Le Keccak 256 bits hacher de la parent bloc de entête, dans son intégralité; officiellement *H* p .

**ommersHash :** Le Keccak 256 bits hacher de la om-

mers liste portion de cette bloc; officiellement *H* o .

**Bénéficiaire:** L'adresse 160 bits à laquelle tous les frais collectés à partir de l'exploitation minière réussie de ce bloc être transféré; officiellement *Hc* . \_

**étatRoot :** Le Keccak 256 bits hacher de la racine

nœud du trie d'état, après que toutes les transactions sont réalisé et finalisations appliqué; officiellement *Hr* . \_

**transactionsRoot :** Le Keccak hachage 256 bits de la

racine nœud de la essayer structure peuplé avec chaque transaction dans la partie liste des transactions du bloc; officiellement *H* t .

**reçusRoot :** Le Keccak 256 bits hacher de la racine

nœud de la structure de trie peuplée de la re- reçus de chaque transaction dans la liste des transactions portion de la bloc; officiellement *Il* . \_

**logsBloom :** Le Floraison filtre composé depuis indice-

informations disponibles (adresse de l'enregistreur et rubriques du journal) contenues dans chaque entrée de journal à partir de la réception de chaque transaction dans la liste des transactions ; officiellement *H* b .

**difficulté:** UN scalaire valeur correspondant pour la dif-

niveau de difficulté de ce bloc. Cela peut être calculé depuis la précédent bloc de difficulté niveau et la horodatage ; officiellement *Hd* . \_

**Numéro:** UN scalaire valeur égal pour la Numéro de un-

blocs cestor. Le bloc de genèse a un certain nombre de zéro; officiellement [*Salut*](#_bookmark30) . [\_](#_bookmark30)

**gasLimit:** A scalar value equal to the current limit

où

c n p g t v

**UN** r \_ s x

de gaz dépense par bloc; officiellement *H* l .

**gazutilisé :** UN scalaire valeur égal pour la total gaz utilisé

(17) **p** *T* **je** si *T* t = ∅

f≡

*T* **d** autrement

Ici, nous supposons que tous les composants sont interprétés par le RLP sous forme de valeurs entières, à l'exception de la liste d'accès *TA* **\_** et la arbitraire longueur octet tableaux *T* **je** et *Td* . **\_**

(18) *T* x ∈ { 0 *,* 1 } ∧ *T* c = *β* ∧ *T* n ∈ N 256 ∧ *T* p ∈ N 256 ∧ *T* g ∈ N 256 ∧ *T* v ∈ N 256 ∧ *T* w ∈ N 256 ∧ *T* r ∈ N 256 ∧ *T* s ∈ N 256 ∧ *T* y ∈ N 1 ∧ *T* **d** ∈ B ∧ *T* **je** ∈ B

où

(19) N n = { *P* : *P* ∈ N ∧ *P <* 2 *n* }

Le hachage d'adresse *T* **t** est légèrement différente : il s'agit soit d'un Hachage d'adresse de 20 octets ou, dans le cas d'un contrat- création transaction (et Donc officiellement égal pour ∅ ), ce est

la RLP vider octet séquence et Donc la membre de B 0 :

dans transactions dans cette bloc; officiellement *Hg* . \_

**horodatage :** une valeur scalaire égale à la valeur raisonnable sortie de time() d'Unix au début de ce bloc ; officiellement [*Hs*](#_bookmark32) . [\_](#_bookmark32)

**extraData :** Un arbitraire octet déployer contenant Les données

correspondant à ce bloc. Celui-ci doit être de 32 octets ou moins de; officiellement *Hx* . \_

**mixHash :** UN 256 bits hacher qui, combiné avec la

fois, prouve qu'une quantité suffisante de calcul tation a été effectuée sur ce bloc ; officiellement *Hm* . \_

**nonce :** UN 64 bits valeur qui, combiné avec la mélanger-

hash, prouve qu'une quantité suffisante de calcul tion a été effectuée sur ce bloc ; officiellement [*Hn*](#_bookmark33) . [\_](#_bookmark33)

1. *J*

∈ f B 20 si *T* t = ∅

Le autre deux Composants dans la bloc sont simplement un liste

B 0 sinon

**t**

* 1. **Le Bloc.** Le bloc dans Ethereum est la recueillir- informations pertinentes (connu sous le nom de bloc *header* ), *H* , ainsi que les informations correspondant à les transactions comprises, **T** , et un ensemble d'autres blocs les en-têtes **U** dont on sait qu'ils ont un parent égal au le parent du parent du bloc actuel (ces blocs sont appelés *ommers* [2](#_bookmark20) ). Le bloc entête contient plusieurs des morceaux de information:

de ommer bloc en-têtes (de la même format comme au dessus de),

*BU* **\_** et un séries de la transactions, *BT* . **\_** Officiellement, nous pouvez référer pour un bloc *B* :

1. *B* ≡ ( *BH ,* \_ *B* **T** *, B* **U** )

2 *heures* est un neutre de genre terme pour moyenne "frère et sœur de parent"; voir [https://nonbinary.miraheze.org/wiki/Gender\_neutral\_language\_in\_](https://nonbinary.miraheze.org/wiki/Gender_neutral_language_in_English%23Aunt/Uncle) [Anglais#tante/oncle](https://nonbinary.miraheze.org/wiki/Gender_neutral_language_in_English%23Aunt/Uncle)

ÉTHERUM : UN SÉCURISE DÉCENTRALISÉ GÉNÉRALISÉ TRANSACTION COMPTABLE **BERLIN VERSION** 6

* + 1. *Reçu de transaction.* Pour encoder des informations à propos un transaction concernant qui ce mai être utile pour former une preuve sans connaissance, ou indexer et rechercher, nous encoder un reçu de chaque transaction contenant certaines informations formation dès son exécution. Chaque reçu, noté *B* **R** [ *i* ] pour la *je* e transaction, est placé dans un [trie](#_bookmark118) indexée et la racine enregistré dans la entête comme [*Il*](#_bookmark24) . [\_](#_bookmark24)

Le transaction le reçu, *R* , est un tuple de cinq éléments com-

en prenant : le type de transaction, *R* x , le code de statut de la transaction, *R* z , le gaz cumulé utilisé dans le bloc contenant le reçu de la transaction immédiatement après la transaction a eu lieu, *R* u , l'ensemble des journaux créés par l'exécution de la transaction, [*R* **l**](#_bookmark116)et la floraison filtre composé \_ \_ \_ \_ depuis information dans ceux journaux, [*R* b](#_bookmark114) :

(22) *R* ≡ (*R*x*, R*z*, R*u*, R*b*, R***l**)

* + 1. *Validité holistique.* On peut affirmer la validité d'un bloc si et seul si ce satisfait plusieurs conditions: ce doit être dans- ternellement compatible avec l'ommer et le bloc de transaction hachages et les transactions données *B* **T** (comme spécifié dans sec [11),](#_bookmark63) lorsqu'il est exécuté dans l'ordre sur l'état de base ***σ*** (dérivé à partir de l'état final du bloc parent), entraînent une nouvelle Etat de la identité *Hr* : \_

(33) *H* r ≡ TRIE ( *L S* (Π( ***σ*** *, B* ))) ∧ *H* o ≡ KEC ( RLP ( *L* ∗ *H* ( *B* **U** ))) ∧ *H* t ≡ TRIE ( {∀ *je <* ll *B* **T** ll *, je* ∈ N :

*pT* ( je *, B* **T** [ *je* ]) } ) ∧

*H* e  ≡ TRIE ( {∀ *je <* ll *B* **R** ll *, je* ∈ N :

*p* R ( *je, B* **R** [ *je* ]) } ) ∧

*H*b ≡

**r**∈*B***R**

**r**b

*R* x est égal au [type](#_bookmark4) de l'opération correspondante.

Le une fonction *LR* \_ se prépare un transaction le reçu pour être transformé dans un RLP sérialisé octet déployer:

1. *L* R ( *R* ) ≡ ( *R* z *, R* tu *, R* b *, R* **l** )

Nous affirmer que la statut code *R* z est un non négatif

où *p* T ( *k, v* ) et *p* R ( *k, v* ) sont des transformées RLP par paires- mais avec un traitement spécial pour les transactions EIP-2718. tions :

(34)

f ≡

*p* ( *k, J* ) RLP ( *k* ) *,* RLP ( [*L* T](#_bookmark12) ( *T* )) si *T* x = 0

T

( *T* x ) · RLP ( *L* T ( *T* )) autrement

entier:

1. *R* z ∈ N

Nous affirmer que *R* tu , la cumulatif gaz utilisé, est un non- négatif entier et que la journaux Floraison, *R* b , est un hacher de

et

(35)

*p* R ( *k, R* ) ≡

RLP ( *k* ) *,*

RLP ( [*L* R](#_bookmark26) ( *R* )) si *R* x = 0

( *R* x ) · RLP ( *L* R ( *R* )) autrement

f

Taille 2048 morceaux (256 octets):

1. *R* tu ∈ N ∧ *R* b ∈ B 256

Le séquence *R* **l** est un séries de Journal entrées, ( *O* 0 *, O* 1 *, ...* ). Une entrée de journal, *O* , est un tuple de l'adresse de l'enregistreur, *O a* , a série éventuellement vide de sujets de journal de 32 octets, *O* **t** et certains Numéro de octets de Les données, *O* **d** :

(26) *O* ≡ ( *O* un *,* ( *O* **t** 0 *, O* **t** 1 *, ...* ) *, O* **d** )

(27) *O* un ∈ B 20  ∧ ∀ *x* ∈ *O* **t** : *x* ∈ B 32  ∧ *O* **d** ∈ B

Nous définissons la fonction de filtre de Bloom, *M* , pour réduire un log entrée dans un Célibataire 256 octets hacher:

( · est le \_ \_ enchaînement de par te \_ arr a ys).

En outre:

(36) TRIE ( *L* S ( ***σ*** )) = *P* ( *BH \_* ) *H* r

Ainsi TRIE ( *L* S ( ***σ*** )) est le hachage du nœud racine du Merkle Patricia arbre structure contenant la valeur clé paires de l'état ***σ*** avec des valeurs codées en utilisant RLP, et *P* ( *B* H ) est la parent bloc de *B* , défini directement.

Les valeurs issues du calcul des transactions notamment les [reçus de transaction,](#_bookmark21) *B* **R** , et que défini par la [fonction d'accumulation d'état](#_bookmark67) de la transaction [tion, Π,](#_bookmark67) sont formalisé plus tard dans section [11.4.](#_bookmark64)

* + 1. *Sérialisation.* Le une fonction *LB* \_ et *L* H sont la prépa-

(28) *M* ( *O* ) ≡

*x* ∈{ *O* une }∪ *O* **t**

*M* 3:2048 ( *x* )

ration les fonctions pour un bloc et bloc entête respectivement. Nous affirmer la les types et Commande de la structure pour lorsque

où *M* 3:2048 est un spécialisé Floraison filtre que ensembles Trois

la RLP transformation est obligatoire:

morceaux dehors de 2048, donné un arbitraire octet séquence. Ce Est-ce que

cette par prise la d'ordre inférieur 11 morceaux de chaque de la trois premières paires d'octets dans un hachage Keccak-256 de l'octet séquence. [3](#_bookmark28) Officiellement:

(37)

(38)

*L* H ( *H* ) ≡ ( *H* p *, Oh ,* \_ *H* c *, Hr ,* \_ *H* t *, Il ,* \_ *H* b *, Hd ,* \_

*Salut ,* \_ *Hl ,* \_ *H* g *, Hs ,* \_ *Hx ,* \_ *Hm ,* \_ *H* n )

*L* B ( *B* ) ≡ *L* H ( *B* H ) *, L* ∗ T ( *B* **T** ) *, L* ∗ H ( [*B* **U**](#_bookmark18) )

(29) *M*

f

|  |  |  |
| --- | --- | --- |
| 3:2048 B) | | ≡ |
| (30) | **y** | = |
| (31) | ∀*i* ∈ {0*,* 2*,* 4} | : |
| (32) | *m*(**x***, i*) | ≡ |

( **x** : **X** ∈

**y** : **y** ∈ B 256 où: (0 *,* 0 *, ...,* 0) à l'exception:

où *L* J prend un spécial se soucier de EIP-2718 transactions:

B 2047 − *m* ( **X** *, je* ) ( **y** ) = 1

KEC ( **x** )[ *je, je* + 1] mode 2048

(39)

*L* J

( *T* ) = [*L* T](#_bookmark12) ( *T* ) si *T* x = 0

( *T* x ) · RLP ( *L* T ( *T* )) autrement

où B est la fonction de référence de bit telle que B j ( **x** ) est égal à le bit d'index *j* (indexé à partir de 0) dans le tableau d'octets **x** . Notamment, ce traite **X** comme big-endian (Suite important morceaux

sera avoir plus petite indices).

avec *L* ∗ T et *L* ∗ H être \_ élément en t séquence transformations , Donc:

(40)

*F* ∗ ( *x* 0 *, x1 ,* \_ *...* ) ≡ *F* ( *x* 0 ) *, F* ( *x* 1 ) *, ...* pour quelconque une fonction *F*

3 11 morceaux = 2 204 8 , et la l o w-commande 11 morceaux est la moi dulo \_ 2048 de la opérande , \_ qui \_ \_ \_ est dans cette Cas est " chacun \_ \_ \_ de la première Trois paires d' octets dans un Keccak-256 hacher de la octet séquence."

ÉTHERUM : UN SÉCURISE DÉCENTRALISÉ GÉNÉRALISÉ TRANSACTION COMPTABLE **BERLIN VERSION** 7

Le composant les types sont défini Donc:

(41) *[H](#_bookmark13)* [p](#_bookmark13) ∈ B 32 ∧ *H* o ∈ B 32 ∧ *H* c ∈ B 20 ∧ [*Hr* \_](#_bookmark23) ∈ B 32 ∧ *H* t ∈ B 32 ∧ [*H* e](#_bookmark24) ∈ B 32 ∧ [*H* b](#_bookmark25) ∈ B 256 ∧ *Hd* \_ ∈ N ∧ [*Salut* \_](#_bookmark30) ∈ N ∧ [*H* l](#_bookmark29) ∈ N ∧ *H* g ∈ N ∧ [*H* s](#_bookmark32) ∈ N 256 ∧ [*H* x](#_bookmark15) ∈ B ∧ *H* m ∈ B 32 ∧ [*H* n](#_bookmark17) ∈ B 8

où

(42) B n = { *B* : *B* ∈ B ∧ ll *B* ll = *n* }

Nous disposons désormais d'un cahier des charges rigoureux pour la construc- tion d'une structure de bloc formelle. La fonction RLP RLP (voir annexe [B)](#_bookmark112) fournit la méthode canonique de trans- formant cette structure en une séquence d'octets prêts pour transmission terminé la câble ou alors stockage localement.

[[2016a].](#_bookmark77) Dans la version *Byzance* , avec EIP-649, la glace l'âge a été retardé en créant un faux numéro de bloc, *H* i, qui est obtenu en soustrayant trois millions de la valeur réelle numéro de bloc, ce qui en d'autres termes a réduit *E* et le différence de temps entre les blocs, afin de permettre plus temps pour développer preuve de participation et empêcher la réseau de « geler ». Par la suite, EIP-1234 par [Schoedon](#_bookmark94) [[2018],](#_bookmark94) EIP-2384 par [Conner [2019],](#_bookmark83) EIP-3554 par [Hancock](#_bookmark102) [[2021],](#_bookmark102) et EIP-4345 par [Beiko et Al. [2021a]](#_bookmark71) a augmenté la soustraire *κ* .

La limite canonique des gaz *H* l d'un bloc d'en-tête *H* doit remplir la relation:

* + 1. *Bloc Entête Validité.* On définit *P* ( *B* H parent bloc de *B* , officiellement:

(43) *P* (*H*) ≡ *B*

: KEC(RLP(*B*H )) = [*H*p](#_bookmark13)

) pour être la

(53)

*H* l *< P* ( *H* ) H l

*H*

l

*> P* (*H*)

Hl

+ *P* ( *H* ) H l

1024

∧

*P* ( *H* )

−

Hl

1024

∧

Le bloc Numéro est la parents bloc Numéro augmenter- menté par une:

(44 ) *Salut* ≡ *P* ( *H* ) Salut \_ + 1

Le canonique difficulté de un bloc de entête *H* est défini comme *D* ( *H* ):

(45)

f234 if *H*i = 0

*H* l ;;: 5000

*H* s est la horodatage (dans d'Unix temps()) de bloc *H* et doit remplir la relation:

*D* ( *H* ) ≡

où:

maximum *ré*

min

*, P* ( *H* ) H d

+ *X* × *ς* 2

+ *E* autrement

(54) *H* s *> P* ( *H* ) H s

(46)

(47)

(48)

*D* min ≡ 2 17

*x P* ( *H* ) H d 2048

≡

*ς* 2 ≡ maximum ( *y* −

f≡

*H* s − *P* ( *H* ) H s

9

*,* − 99

Ce mécanisme impose une homéostasie en termes de temps entre blocs; un plus petite période entre la dernière deux blocs entraîne une augmentation du niveau de difficulté et donc Additionnel calcul obligatoire, rallongement la probable suivant période. Inversement, si la période est aussi grande, la

(49)

(50)

(51)

*y* 1 si ll *P* ( *H* ) **U** ll = 0

2 autrement

*E* ≡ 2 l *H* i *t* ÷ 100000 J − 2

*Salut* \_ ≡ max( *H* je − *κ,* 0)

|  |  |  |  |
| --- | --- | --- | --- |
| (52) |  |  | (55) *n H*  d |
| *κ* ≡ | ****3000000 | if  if if if | *F*Byzantium *H*i *< F*Constantinople  *F*Constantinople *H*i *< F*MuirGlacier  *F*MuirGlacier *H*i *< F*London with (*n, m*) = PoW(*H*n*, H F*London *H*i *< F*ArrowGlacier Where *H*n is the new bl |

difficulté, et attendu temps pour la suivant bloc, est réduit.

Le non fois, [*H* n](#_bookmark17) , doit satisfaire la rapports:

2 256

5000000

****

9000000

9700000

****

  10700000 si *Salut* \_

;;: *F*

FlècheGlacier

∧m *\_* = *H* m

n *,* **d** ).

ock's header *H* , mais *sans* le nonce et mélange-hachage Composants, **ré** être la courant GAD, un grande Les données Positionner avait besoin pour calculer la mélange-hachage, et PoW

Le *Propriété* difficulté paramètre, *ς* 2 , est utilisé pour affecter

un dynamique homéostasie de temps entre blocs, comme la temps entre les blocs varie, comme indiqué ci-dessous, tel que mis en œuvre dans EIP-2 par [Butérine](#_bookmark76) [[2015].](#_bookmark76) Dans la Propriété Libération, le symbole de difficulté exponentielle, *E* provoque la difficulté à augmenter lentement (tous les 100 000 blocs) à un rythme exponentiel taux, et augmentant ainsi la différence de temps de bloc, et mettre une pression de temps sur la transition vers la preuve de participation. Cet effet, connu sous le nom de "bombe de difficulté", ou "ère glaciaire", a été expliqué dans EIP-649 par [Schoedon et Butérine [2017]](#_bookmark95)

et retardé et mis en œuvre plus tôt dans EIP-2. *ς* 2 était aussi modifié dans EIP-100 avec l'utilisation de *x* , l'ajustement facteur ci-dessus, et le dénominateur 9, afin de cibler la moyenne bloc temps y compris oncle blocs par [Butérine](#_bookmark77)

est la preuve de travail une fonction (voir section [11.5):](#_bookmark68) cette évaluer-

ates à un tableau avec le premier élément étant le mix-hash, à prouver que un Corriger DAG possède a été utilisé, et la seconde Objet être un pseudo-aléatoire Numéro cryptographiquement dépendant de *H* et **d** . Étant donné un approximativement uniforme Distribution dans le plage [0 *,* 2 64 ), la temps prévu pour trouver un Solution est proportionnel pour la difficulté, *Hd* . \_

C'est le fondement de la sécurité de la blockchain et est la raison fondamentale pour laquelle un nœud malveillant peut- ne pas propager les blocs nouvellement créés qui, autrement, écraser ("réécrire") l'historique. Parce que le nonce doit s'asseoir satisfait à cette exigence, et parce que sa satisfaction dépend sur le contenu du bloc et à son tour sa composition transactions, créer Nouveau, valide, blocs est difficile et,

ÉTHERUM : UN SÉCURISE DÉCENTRALISÉ GÉNÉRALISÉ TRANSACTION COMPTABLE **BERLIN VERSION** 8

terminé temps, a besoin environ la total calculer Puissance de la fiable portion de la exploitation minière pairs.

Ainsi nous sont capable pour définir la bloc entête validité une fonction *V* ( *H* ):

256

2

1. *V* ( *H* ) ≡ *n H* ∧ *m* = *H* m ∧

d

*Hd* \_ = *ré* ( *H* ) ∧

*H* g ≤ *H* l  ∧

∧

* 1. Transaction Exécution

L'exécution d'une transaction est la partie la plus complexe du protocole Ethereum : il définit la transition d'état fonction [Υ.](#_bookmark0) Il est supposé que toutes les transactions exécutées première passer la initial essais de intrinsèque validité. Celles-ci inclure:

* 1. La transaction est bien formé RLP, avec pas d'addi- national traînant octets ;
  2. la transaction Signature est valide;

*H* l *< P* ( *H* ) H l

+ *P* ( *H* ) H l

1024

*P* ( *H* ) H l

*H*l *> P* (*H*)Hl −

1024

* 1. la [transaction nonce](#_bookmark40) est valide (équivalent pour la [expéditeur comptes courant nonce);](#_bookmark3)
  2. la expéditeur Compte possède non Contrat code déployé

∧

(see EIP-3607 by [Feist et al. [2021]);](#_bookmark93)

*H* l ;;: 5000 ∧

*H* s *> P* ( *H* ) H s ∧

*Salut* \_ = *P* ( *H* ) Salut \_ + 1 ∧

ll *H* x ll ≤ 32

où ( *n, m* ) = PoW ( *H* n *, H* n *,* **ré** )

Notant aditionellement que **extraData** doit être à plus 32 octets.

5. Gaz et paiement

Afin d'éviter les problèmes d'abus de réseau et d'éviter les inévitables questions issues de Turing complètent- ness, tous les calculs programmables dans Ethereum sont soumis pour frais. Le frais programme est spécifié dans unités de *gaz* (voir Ap- appendice [g](#_bookmark131) pour la frais associé avec divers calcul). Ainsi tout fragment donné de calcul programmable (cette comprend créer contrats, fabrication message appels, utiliser et accéder au stockage du compte et exécuter des op- rations sur la machine virtuelle) a un consensus universel Coût dans termes de gaz.

Chaque transaction a une quantité spécifique de gaz associée é avec ce: **GasLimit** . Cette est la montant de gaz qui est implicitement acheté à partir du solde du compte de l'expéditeur. L'achat a lieu au prix du **gaz correspondant** , aussi spécifié dans la transaction. Le transaction est consi- ered invalide si la Compte solde ne peux pas Support tel un achat. Ce est nommé **GasLimit** puisque quelconque inutilisé gaz à la finir de la transaction est remboursé (à la même taux d'achat) sur le compte de l'expéditeur. Le gaz ne existent en dehors de l'exécution d'une transaction. Ainsi pour comptes avec de confiance code associé, un relativement haute gaz limite mai être Positionner et la gauche seule.

En général, Ether achetait du gaz qui n'est pas recyclé. financé est livré à l' adresse du *bénéficiaire* , l'adresse de un Compte typiquement en dessous de la contrôler de la mineur. Transacteurs sont libre pour spécifier quelconque **gazPrix** que elles ou ils souhaite, cependant les mineurs sont libres d'ignorer les transactions comme ils choisissent. Un prix du gaz plus élevé sur une transaction avant Coût la expéditeur Suite dans termes de Éther et livrer une plus grande valeur pour le mineur et sera donc plus susceptible d'être sélectionné pour être inclus par plus de mineurs. Les mineurs, en général, choisira d'annoncer le prix minimum du gaz pour lequel ils exécuteront des transactions et les commerçants seront libres de sonder ces prix pour déterminer le prix du gaz à offre. Puisqu'il y aura une distribution (pondérée) de min- prix du gaz acceptables, les opérateurs devront nécessairement ont un compromis à faire entre la baisse du prix du gaz et maximiser les chances que leur transaction soit miné dans un opportun manière.

* 1. la gaz limite est non plus petite que la intrinsèque gaz,

*g* 0 , utilisé par la transaction; et

* 1. la expéditeur Compte solde contient à moins la Coût, *v* 0 , obligatoire dans à l'avant Paiement.

Officiellement, nous considérer la une fonction [Υ,](#_bookmark0) avec *J* être un transaction et ***σ*** la Etat:

(57 ***)*** = Υ( ***σ*** *, J* )

Ainsi ***σ*** est l'état post-transactionnel. Nous définissons également [Υg \_](#_bookmark42) pour évaluer pour la montant de gaz utilisé dans la exécution d'une transaction, [Υ **l**](#_bookmark42) à évaluer par rapport aux charges à payer de la transaction éléments de log et [Υ z](#_bookmark42) pour évaluer le code d'état résultant depuis la transaction. Celles-ci sera être officiellement défini plus tard.

* 1. **Sous-état.** Tout au long de l'exécution de la transaction, nous assurons Crue certaines informations qui sont traitées immédiatement suite à l'opération. Nous appelons cela la *transaction accumulée. sous-état de tion* , ou sous- *état accumulé* en abrégé, et représentent ce comme *Un* , qui est un tuple :

(58) *Un* ≡ ( *Un* **s** *,* **Al** *, \_ A* **t** *, A* r *, Un* **un** *,* **AK** ) *\_*

Le contenu du tuple inclut *A* **s** , l'ensemble d'autodestruction : un ensemble de comptes qui seront éliminés suite à la trans- l'achèvement de l'action. *A* **l** est la série log : il s'agit d'une série de archivé et indexable 'points de contrôle' dans VM code exécution qui permettent aux appels contractuels d'être facilement suivis par onlook- ers externes au monde Ethereum (tels que décentralisés frontaux d'application). *A* **t** est l'ensemble des comptes touchés, de qui la vider ceux sont supprimé à la finir de un transac- tion. *Un* r est le solde du remboursement, augmenté grâce à l'utilisation la

[STORE](#_bookmark145) instruction dans Commande pour réinitialiser Contrat stockage pour

zéro à partir d'une valeur non nulle. Bien que pas immédiatement remboursé, ce est autorisé pour partiellement décalage la total exécution frais. Enfin, EIP-2929 par [Buterin et Swende [2020a]](#_bookmark81) introduit *A* **a** , l'ensemble des adresses de compte consultées, et *A* **K** , l'ensemble des clés de stockage accédées (plus précisément, chaque élément **de** *AK*est un tuple d'une adresse de compte de 20 octets et un 32 octets stockage insérer).

Nous définissons le sous-état accumulé vide *A* 0 ne pas avoir s'autodétruit, non journaux, non touché comptes, zéro rembourser bal-

ance, tous les contrats précompilés dans les adresses consultées, et non accédé stockage:

(59) *Un* 0 ≡ ( ∅ *,* () *,* ∅ *,* 0 *, π,* ∅ )

où [*π*](#_bookmark51)est la Positionner de tout précompilé adresses.

ÉTHERUM : UN SÉCURISE DÉCENTRALISÉ GÉNÉRALISÉ TRANSACTION COMPTABLE **BERLIN VERSION** 9

* 1. **Exécution.** Nous définissons le gaz intrinsèque *g* 0 , la quantité de gaz que cette transaction nécessite d'être payé avant l'exécution, comme suit :

accumulé sous-état *UN* et statut code *z* : (66)

 **** [Λ](#_bookmark44) 4 ( ***σ*** 0 *, A* ∗ *, S* ( *T* ) *, S* ( *T* ) *, g ,*

(***σ***

P

*, g, A, z*) ≡ 

*T*p*, T*v*, T***i***,* 0*,* ∅*,* T) if *T*t = ∅

∗

*g* 0 ≡

(60)

f [*G* txdatazero](#_bookmark136)  si *je* = 0

[Θ](#_bookmark47) 4 ( ***σ*** 0 *, UN , S* ( *T* ) *, S* ( *T* ) *, T* t *,*

 *T* t *, g , T* p *, Télé ,* \_ *Télé ,* \_ *T* **d** *,* 0 *,* T ) autrement

****

*je* ∈ *T* **je** *,T* **ré**

f

[*G* txdatanonzero](#_bookmark137) autrement

où

+ [*G* txcreate](#_bookmark135) si *T* t = ∅

0 sinon

(67)

(68)

*Un* ∗ ≡ *Un* 0 à l'exception

*Un* ∗ **un** ≡ *Un* 0 ∪ { *S* ( *T* ) } ∪ *E* ∈ *T*

{ [*E*](#_bookmark9) une }

+ [transaction*G*](#_bookmark138)

*TA* **\_** − 1

(69)

**un**

*A* ∗ **K** ≡ L

**UN**

{ ∀ *je <* ll *E* ll *,* **\_** *je* ∈ N : ( *E* a *, E* **s** [ *je* ]) }

+

*j* =0

Adresse de la liste d'accès *G* + ll *T* **A** [ *j* ] **s** ll *G* accessliststorage

*E* ∈ *TA* **\_**

et *g* est la montant de gaz restant après déduire

où *T* **je** *, T* **d** signifie la série d'octets de la transaction associé Les données et initialisation code EVM, en fonction, dépendemment sur qu'il s'agisse la transaction est pour création de contrat ou alors message-

appel. *G* txcreate est ajoutée si la transaction est Contrat- création, mais pas si un résultat de code EVM. [Adresse de la liste d'accès *G*](#_bookmark132) et [*G* accessliststorage](#_bookmark133) sont les frais d'échauffement et stockage accéder, respectivement. *g* est pleinement défini dans Ap-

appendice [G.](#_bookmark131)

Le à l'avant Coût *v* 0 est calculé comme:

(61) *contre* 0 ≡ [*T* g](#_bookmark6) [*T* p](#_bookmark5) + [*Télévision* \_](#_bookmark7)

Le validité est déterminé comme:

la basique montant obligatoire pour Payer pour la existence de la transaction:

1. *g* ≡ *T* g − *g* 0

Notez que nous utilisons [Θ](#_bookmark47) 4 et [Λ](#_bookmark44) 4 pour indiquer le fait que seul le les quatre premières composantes des valeurs des fonctions sont prises ; la finale représente la valeur de sortie de l'appel de message (un octet déployer) et est inutilisé dans la le contexte de transaction évaluer-

tion.

Après le traitement de l'appel de message ou de la création du contrat, le compteur de remboursement doit être incrémenté pour les comptes que ont été auto-détruit tout au long de son invocation.

1. *Un* r ≡ *Un* r + *R* autodestruction

*je* ∈ *A* **s**

(62) *S* ( *T* ) = ∅

***σ***[*S*(*T* )]c = KEC ()

∧

Puis la Etat est finalisé par déterminer la montant

*T* n = ***σ*** [ *S* ( *T* )] n ∧

∧

*g* 0 *T* g ∧

*v* 0 ***σ*** [ *S* ( *T* )] b ∧

to be refunded, *g*∗ from the remaining gas, *g*, plus some

allocation depuis la rembourser compteur, pour la expéditeur à la original taux.

*T* g *B* H l − [*.€*](#_bookmark2) ( *B* **R** ) u

Noter la final état; la somme de la transactions gaz

1. *g* ∗ ≡ *g*

+ min

*T* g − *g*

2

*, Un* r

limite, *T* g , et la gaz utilisé dans cette bloc avant, donné par

[*.€*](#_bookmark2) ( *B* **R** ) u , je dois être \_ non plus grand que la bl o c k's **GasLimi t** , *B* Hl . Aussi, avec un léger abuser de de notation, nous présumer que ***σ*** [ *S* ( *T* )] c = KEC () , ***σ*** [ *S* ( *T* )] n = 0, et ***σ*** [ *S* ( *T* )] b = 0 si ***σ*** [ *S* ( *T* )] = ∅ .

Le exécution de un valide transaction commence avec un changement irrévocable apporté à l'état : le [nonce de la](#_bookmark3) [compte de l'expéditeur,](#_bookmark3) *S* ( *T* ), est incrémenté de un et le solde est réduit par partie de la à l'avant Coût, *T* g *T* p . Le

Le total remboursable montant est la légitimement rester-

gaz *g*, ajoutée à [*A* r](#_bookmark37) , cette dernière composante étant plafonné à un maximum de la moitié (arrondi à l'inférieur) du quantité totale utilisée *T* g − *g*. Donc, *g* ∗ est le gaz total que restes après la transaction possède a été réalisé.

Le Éther pour la gaz est donné pour la mineur, à qui adresse est spécifié comme la Bénéficiaire de la cadeau bloc

*B.* \_ Alors nous définir la pré-finale Etat ***σ*** ∗ dans termes de la

provisoire Etat ***σ*** P :

gaz disponible pour la procéder calcul, *g* , est défini

comme *T* g − *g* 0 . Le calcul, qu'il s'agisse Contrat création

***σ*** ∗ ≡ ***σ*** P à l'exception

ou un appel de message, aboutit à un état éventuel (qui peut juridiquement équivalent à l'état actuel), le passage à qui est déterministe et jamais invalide : il ne peut y avoir invalide transactions depuis cette indiquer.

***σ*** ∗ [ *S* ( *T* )] b  ≡ ***σ*** P [ *S* ( *T* )] b + *g* ∗ *T* p

***σ*** ∗ [ *m* ] b  ≡ ***σ*** P [ *m* ] b + ( *T* g − *g* ∗ ) *T* p *m* ≡ *BHC* \_

Nous définir la point de contrôle Etat ***σ*** 0 :

Le final Etat, ***σ***, est atteint après suppression tout comptes

que Soit apparaître dans la auto-destruction Positionner ou alors sont touché

(63)

(64)

***σ*** 0 ≡ ***σ*** à l'exception:

***σ*** 0 [ *S* ( *T* )] b  ≡ ***σ*** [ *S* ( *T* )] b − *T* g *T* p

et vider:

***σ*** ≡ ***σ*** ∗ à l'exception

(65)

***σ*** 0 [ *S* ( *T* )] n ≡ ***σ*** [ *S* ( *T* )] n + 1

∀ *je* ∈ *Un* **s** : ***σ*** [ *je* ] = ∅

Évaluation ***σ*** P

depuis ***σ*** 0

dépend sur la transaction taper;

∀ *je* ∈ *Un* **t** : ***σ*** [ *je* ] = ∅ si MORT ( ***σ*** ∗ *, je* )

Soit Contrat création ou alors message appel; nous définir la tuple de post-exécution provisoire Etat ***σ*** P , restant gaz *g*,

Et finalement, nous spécifier Υ g , la total gaz utilisé dans cette transaction Υ **l** , la journaux créé par cette transaction et

ÉTHERUM : UN SÉCURISE DÉCENTRALISÉ GÉNÉRALISÉ TRANSACTION COMPTABLE **BERLIN VERSION** 10

f

Υ z , la statut code de cette transaction:

(90)

***σ*** ∗ [ *une* ] = 1 *, v* + *v,* [TRIE](#_bookmark118) ( ∅ ) *,* KEC ()

Υ g ( ***σ*** *, J* ) ≡ *T* g − *g* ∗

Υ **l** ( ***σ*** *, J* ) ≡ [*A* **l**](#_bookmark38)

(91)

***σ*** ∗ [ *s* ] =

∅ si ***σ*** [ *s* ] = ∅ ∧ *v* = 0

**un** ∗ autrement

Υ z ( ***σ*** *, J* ) ≡ *z*

(92)

**un** ∗  ≡ ( ***σ*** [ *s* ] n *,* ***σ*** [ *s* ] b − *v ,* ***σ*** [ *s* ] **s** *,* ***σ*** [ *s* ] c )

Ceux-ci sont utilisés pour aider à définir le [reçu de transaction](#_bookmark21) et sont aussi utilisé [plus tard](#_bookmark66) pour Etat et nonce validation.

* 1. Contrat Création

Il existe un certain nombre de paramètres intrinsèques utilisés lorsque créer un compte: expéditeur ( *s* ), agent d'origine [4](#_bookmark46) ( *o* ), gaz disponible ( *g* ), prix du gaz ( *p* ), dotation ( *v* ) ensemble avec un arbitraire tableau d'octets de longueur, **je** , l'initialisation EVM code, la cadeau profondeur de la message-appel/contrat- pile de création ( *e* ), le sel pour l'adresse du nouveau compte ( *ζ* ) et enfin l' autorisation d'apporter des modifications au Etat ( *w* ). Le sel *ζ* pourrait être disparu ( *ζ* = ∅ ); officiellement,

(83) *ζ* ∈ B 32 ∪ B 0

où *v* est la valeur préexistante du compte, dans le cas où ce a été précédemment dans existence:

(93) *v* 0 si ***σ*** [ *une* ] = ∅

f≡

***σ*** [ *une* ] b autrement

Enfin, le compte est initialisé par l'exécution du code EVM d'initialisation **i** en fonction de l'exécution modèle (voir section [9).](#_bookmark52) L'exécution du code peut affecter plusieurs événements qui ne sont pas internes à l'état d'exécution : la le stockage du compte peut être modifié, d'autres comptes peuvent être créés et d'autres appels de messages peuvent être passés. En tant que tel, la fonction d'exécution de code [Ξ](#_bookmark57) s'évalue à un tuple du résultant Etat ***σ*** ∗∗ , disponible gaz restant *g* ∗∗ , la concernant- sultant accumulé sous-état *Un* ∗∗ et la corps code de la Compte **o** .

Si la création a été causé par [CRÉER2](#_bookmark150) , alors *ζ* ∅ .

On définit formellement la fonction de création comme la fonction Λ, qui s'évalue à partir de ces valeurs, ainsi que Etat ***σ*** et la accumulé sous-état *Un* , pour la tuple contenant le nouvel état, le gaz restant, le nouveau sous-état accumulé, le statut code et production ( ***σ****, g, UN, z,* **o** ):

(84) ( ***σ****, g, UN, z,* **o** ) ≡ Λ ( ***σ*** *, UN, s, o, g, p, v,* **je** *, e, , \_ w* )

L'adresse du nouveau compte est définie comme étant le le plus à droite 160 morceaux de la Keccak-256 hacher de la [RLP](#_bookmark113) encodage de la structure ne contenant que l'expéditeur et le [compte nonce.](#_bookmark3) Pour [CREATE2](#_bookmark150) la règle est différente et est décrit dans EIP-1014 par [Buterin](#_bookmark82) [[2018].](#_bookmark82) Combiner les deux cas, nous définissons l'adresse résultante pour le nouveau Compte *un* :

(94) ( ***σ*** ∗∗ *, g* ∗∗ *, A* ∗∗ *,* **o** ) ≡ Ξ( ***σ*** ∗ *, g, A* ∗ *, je* )

où *je* contient la paramètres de la [exécution environ-](#_bookmark53) [hommes](#_bookmark53) , que est:

(85) *un* ≡ ADR ( *s,* ***σ*** [ *s* ] n − 1 *, , \_* **je** )

|  |  |  |  |
| --- | --- | --- | --- |
| (95) | *I*a | ≡ | *a* |
| (96) | *I*o | ≡ | *o* |
| (97) | *I*p | ≡ | *p* |
| (98) | *I***d** | ≡ | () |
| (99) | *I*s | ≡ | *s* |
| (100) | [*I*v](#_bookmark56) | ≡ | *v* |
| (101) | *I***b** | ≡ | **i** |
| (102) | *I*e | ≡ | *e* |
| (103) | *I*w | ≡ | *w* |

(86) ADR ( *s, n, , \_* **je** ) ≡ B 96 *..* 255 KEC *LA* ( *s* , *n, , \_* **je** )

f

*je* **d** évalue au tuple vide car il n'y a pas de données d'entrée pour cette appel. *Je* H possède non spécial traitement et est déterminé

(87)

*LA* ( *s* , *n, ζ,* **je** ) ≡

RLP ( *s, n* ) si *ζ* = ∅

(255) · *s* · *ζ* · KEC ( **i** ) autrement

depuis la chaîne de blocs.

Code exécution épuise gaz, et gaz mai ne pas aller au dessous de zéro, Donc exécution mai sortie avant que la code possède viens

où · est la concaténation de tableaux d'octets, B *a..b* ( *X* ) évaluant correspond à une valeur binaire contenant les bits d'indices dans le Portée [ *un, b* ] de la binaire Les données *X* , et ***σ*** [ *x* ] est la adresse Etat de *X* , ou alors ∅ si rien existe. Noter nous utiliser une moins de que

la valeur nonce de l'expéditeur ; nous affirmons que nous avons augmenté mentionné le nonce du compte de l'expéditeur avant cet appel, et donc la valeur utilisée est le nonce de l'expéditeur au début de la responsable transaction ou alors VM opération.

L'adresse du nouveau compte est ajoutée à l'ensemble des accédé comptes:

1. *Un* ∗ ≡ *UN* à l'exception *Un* ∗ **un** ≡ *Un* **un** ∪ { *un* }

Le nonce du compte est initialement défini comme un, le bal- ance comme la valeur passé, la stockage comme vider et la hachage de code en tant que hachage Keccak 256 bits de la chaîne vide ; le solde de l'expéditeur est également réduit de la valeur transmise. Ainsi la muté Etat devient ***σ*** ∗ :

1. ***σ*** ∗ ≡ ***σ*** à l'exception:

à un état d'arrêt naturel. Dans cette (et plusieurs autres) ex- cas exceptionnels, nous disons qu'une exception de manque de gaz (OOG) a s'est produit: Le évalué Etat est défini comme être la vider Positionner, ∅ , et la entier créer opération devrait avoir non effet sur l'état, le laissant effectivement tel qu'il était immédiatement avant pour tenter la création.

Si le code d'initialisation se termine avec succès, un dernier création de contrat Coût est payé, la dépôt de code Coût, *c* , proportionnel pour la Taille de la créé contrat de code:

(104) *c* ≡ Dépôt de code *G* × tout **va** bien

Si là est ne pas suffisant gaz restant pour Payer cette, c'est à dire

*g* ∗∗ *< c* , alors nous aussi déclarer un panne d'essence exception.

Le gaz restant sera nul dans de tels cas exceptionnels. condition, c'est-à-dire si la création a été conduite comme la réception tion d'une transaction, cela n'affecte pas le paiement de le coût intrinsèque de la création de contrats ; il est payé quoi qu'il en soit. Toutefois, la valeur de la transaction n'est pas transférée à la avorté contrat de adresse lorsque nous sont panne d'essence.

4 qui pouvez différer depuis la expéditeur dans la Cas de un message appel ou alors Contrat création ne pas directement déclenché par un transaction mais à venir depuis la exécution de Code EVM

ÉTHERUM : UN SÉCURISE DÉCENTRALISÉ GÉNÉRALISÉ TRANSACTION COMPTABLE **BERLIN VERSION** 11

Si tel un exception Est-ce que ne pas se produire, alors la restant le gaz est remboursé à l'initiateur et à l'état maintenant modifié est autorisé à persister. Ainsi formellement, on peut préciser résultant Etat, gaz, accumulé sous-état et statut code comme ( ***σ****, g, UN, z* ) où:

(105)

*g* 0 si *F*

f≡

*g* ∗∗ − *c* autrement

(106)

****

***σ*** si *F* ∨ ***σ*** ∗∗ = ∅

****  ***σ*** sauf :

∗∗

destinataire ( *r* ), le compte dont le code doit être exécuté ( *c* , généralement le même que le destinataire), gaz disponible ( *g* ), valeur ( *v* ) et le prix du gaz ( *p* ) avec un octet de longueur arbitraire déployer, **ré** , la saisir Les données de la appel, la cadeau profondeur de la appel-message/création-contrat pile ( *e* ) et finalement la autorisation pour Fabriquer modifications pour la Etat ( *w* ).

De côté depuis évaluer pour un Nouveau Etat et accumulé transaction sous-état, message appels aussi avoir un En plus composant - les données de sortie désignées par le tableau d'octets **o** . Cette est ignoré lorsque exécution transactions, pourtant mes- sauge appels pouvez être initié exigible pour Code VM exécution et dans cette Cas cette information est utilisé.

(110)

( ***σ****, g**, UN**, z ,* **o** ) ≡ Θ ( ***σ*** *, UN, s, o, r , c, g , p, v , v* ˜ *,* **ré** *, e, w* )

***σ*** ≡

(107)

***σ***[ *un* ] = ∅ si MORT ( ***σ*** ∗∗ *, un* )

***σ*** ∗∗ à l'exception:

****

  ***σ***[ *une* ] c = KEC ( **o** ) autrement

Noter que nous avoir besoin pour différencier entre la valeur que

est pour être transféré, *v* , depuis la valeur apparent dans la exécution co n t e x t, *v* ˜ , pour la DELEG A TECALL instruction. Nous définir ***σ*** 1 , la première de transition Etat comme la origine- finale Etat mais avec la valeur transféré depuis expéditeur pour

*A A* ∗  si *F* ∨ ***σ*** ∗∗ = ∅

f

≡

*Un* ∗∗ autrement

(108)

f≡

*z* 0 si *F* ∨ ***σ*** ∗∗ = ∅

1 autrement

où (109)

*F* ≡ ***σ*** [ *une* ] = ∅ ∧ ***σ*** [ *une* ] *c* = KEC () ∨ ***σ*** [ *une* ] *n* = 0 ∨

( ***σ*** ∗∗ = ∅ ∧ **o** = ∅ ) ∨

*g* ∗∗ *< c* ∨

destinataire:

(111) ***σ*** 1 [ *r* ] b ≡ ***σ*** [ *r* ] b + *v* ∧ ***σ*** 1 [ *s* ] b ≡ ***σ*** [ *s* ] b − *v*

sauf si *s* = *r* .

Tout au long de la cadeau travail, ce est assumé que si ***σ*** 1 [ *r* ] n'était pas défini à l'origine, il sera créé en tant que compte sans code ni état et solde nul et nonce. Ainsi la précédent équation devrait être pris pour moyenne:

(112) ***σ*** 1 ≡ ***σ*** 1 à l'exception:

ll **o** ll *>* 24576

(113) ***σ*** 1 [ *s* ] ≡ f ∅

si ***σ***1 [ *s* ] = ∅

∧ *v* = 0

L'exception dans la détermination de ***σ*** dicte que **o** , la séquence d'octets résultante de l'exécution de la code d'initialisation, spécifie le code de corps final pour le nouvellement créé Compte.

Notez que l'intention est que le résultat soit soit une réussite nouveau contrat créé avec succès avec sa dotation, ou pas Nouveau Contrat avec non transférer de valeur. Dans une addition, ob-

servir que si l'exécution du code d'initialisation [revient](#_bookmark151) ( ***σ*** ∗∗ = ∅ ∧ **o** = ∅ ), le gaz résultant *g* n'est pas épuisé (à condition de là a été non autre exception), mais non Nouveau ac-

compter est créé.

7.1. **Subtilités.** Notez que tant que le code d'initialisation est en cours d'exécution, l'adresse nouvellement créée existe mais avec pas de code corporel intrinsèque [5](#_bookmark49) . Ainsi tout message d'appel reçu pendant ce temps, aucun code ne sera exécuté. Si l'exécution de l'initialisation se termine par un SELFDESTRUCT instruction, la question est sans objet puisque le compte sera supprimé avant la fin de la transaction. Pour une normale ARRÊT code, ou si la code revenu est autrement vide, alors l'état se retrouve avec un compte zombie, et tout ce qui reste solde sera être fermé à clé dans la Compte pour toujours.

* 1. Message Appel

Dans le cas de l'exécution d'un appel de message, plusieurs paramètres eters sont obligatoire: expéditeur ( *s* ), transaction auteur ( *o* ),

**un** 1 autrement

(114) **à** 1 ≡ ***σ***1 [ *s* ] n *,* ***σ***1 [ *s* ] b − *v ,* ***σ***1 [ *s* ] **s** *,* ***σ***1 [ *s* ] c

(115) et ***σ*** 1 ≡ ***σ*** à l'exception:

(116)



***σ***1 [ *r* ] ≡ (0 *, v ,* TRIE ( ∅ ) *,* CEC (())) si ***σ*** [ *r* ] = ∅ ∧ *v*  = 0

***σ***1 [ *r* ] ≡ ∅ si ***σ*** [ *r* ] = ∅ ∧ *v* = 0



σ ***\_***1 [ *r* ] ≡ **un**1  sinon

(117) **un**1 ≡ ( ***σ*** [ *r* ] n *,* ***σ*** [ *r* ] b + *v ,* ***σ*** [ *r* ] **s** *,* ***σ*** [ *r* ] c )

Le code associé au compte (identifié comme le fragment ment dont le hachage Keccak-256 est ***σ*** [ *c* ] c ) est exécuté selon au modèle d'exécution (voir section [9).](#_bookmark52) Tout comme avec con- tract création, si la exécution s'arrête dans un exceptionnel mode (c'est-à-dire en raison d'une alimentation en gaz épuisée, sous- flux, destination de saut invalide ou instruction invalide), puis aucun gaz n'est remboursé à l'appelant et l'état revient à la indiquer immédiatement avant pour solde transférer (c'est à dire ***σ*** ).

5 Pendant initialisation code exécution, EXTCODETAILLE sur la adresse devrait retourner zéro, qui est la longueur de la code de la Compte pendant que

CODETAILLE devrait retourner la longueur de la initialisation code (comme défini dans [H.2).](#_bookmark142)

ÉTHERUM : UN SÉCURISE DÉCENTRALISÉ GÉNÉRALISÉ TRANSACTION COMPTABLE **BERLIN VERSION** 12

où

1. Ξ ≡

et

***σ σ*** si ***σ*** ∗∗ = ∅

***σ*** ∗∗ autrement



f

|  |  |  |
| --- | --- | --- |
| (118) |  | ≡ |
| (119) | *g* | ≡ |
| (120) |  | ≡ |
| (121) | *z* | ≡ |

0 si ***σ*** ∗∗ = ∅ ∧

**o** = ∅

f

*g* ∗∗ autrement

*UNE UNE* si ***σ*** ∗∗ = ∅

*Un* ∗∗ autrement

f 0 si ***σ*** ∗∗ = ∅

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| (122) (***σ***∗∗*, g*∗∗*, A*∗∗*,* | **o**) | ≡ | Ξ | 1 | otherwise |
| (123) | *I*a | ≡ | *r* |  |  |
| (124) | *I*o | ≡ | *o* |  |  |
| (125) | *I*p | ≡ | *p* |  |  |
| (126) | *I***d** | ≡ | **d** |  |  |
| (127) | *I*s | ≡ | *s* |  |  |
| (128) | *I*v | ≡ | *v*˜ |  |  |
| (129) | *I*e | ≡ | *e* |  |  |
| (130) | *I*w | ≡ | *w* |  |  |

Ξ ECREC ( ***σ*** 1 *, g, A, je* ) si *c* = 1 Ξ SHA256 ( ***σ*** 1 *, g, A, je* ) si *c* = 2 Ξ RIP160 ( ***σ*** 1 *, g, A, je* ) si *c* = 3 Ξ ID ( ***σ*** 1 *, g, UN, je* ) si *c* = 4 Ξ EXPMOD ( ***σ*** 1 *, g, A, je* ) si *c* = 5 ΞBN \_ AJOUTER ( ***σ*** 1 *, g, A, je* ) si *c* = 6 ΞBN \_ MUL ( ***σ*** 1 *, g, A, je* ) si *c* = 7 Ξ SNARKV ( ***σ*** 1 *, g, A, je* ) si *c* = 8 Ξ BLAKE2 F ( ***σ*** 1 *, g, UN, je* ) si *c* = 9

****

****

****

****

****

Ξ( ***σ*** 1 *, g, UN, je* ) sinon

modèle formel d'une machine à états virtuelle, connu comme le Machine virtuelle Ethereum (EVM). C'est un *quasi-* Turing- machine complète ; la *quasi -* qualification vient de la fait que le calcul est intrinsèquement borné par un paramètre, *gas* , qui limite la quantité totale de tation Fini.

* 1. **Bases.** L'EVM est une architecture simple basée sur la pile ture. La taille du mot de la machine (et donc la taille de la pile éléments) est de 256 bits. Cela a été choisi pour faciliter le Keccak- 256 schéma de hachage et calculs de courbes elliptiques. Le Le modèle de mémoire est un simple tableau d'octets adressé par mot. Le pile a une taille maximale de 1024. La machine a également un indépendant stockage maquette; cette est similaire dans concept à la mémoire mais plutôt qu'un tableau d'octets, c'est un mot- tableau de mots adressable. Contrairement à la mémoire, qui est volatile, le stockage est non volatil et est maintenu dans le cadre du système Etat. Tout Emplacements dans tous les deux stockage et Mémoire sont bien défini initialement comme zéro.

La machine ne suit pas la norme von Neu- homme architecture. Plutôt que stocker programme code dans généralement accessible Mémoire ou alors stockage, ce est stockée séparément dans un virtuel ROM interactif seul par un spécialisé instruction.

La machine peut avoir une exécution exceptionnelle pendant plusieurs raisons, y compris les débordements de pile et les instructions non valides tions. Comme l'exception de panne de gaz, ils ne partent pas changements d'état intacts. Au contraire, la machine s'arrête immédiatement. rapidement et signale le problème à l'agent d'exécution (soit la transaction processeur ou alors, récursivement, la frai exécution environnement) qui sera accord avec ce séparément.

* 1. **Frais Aperçu.** Frais (dénomination dans gaz) sont inculpé dans trois circonstances distinctes, toutes trois comme préalable à l'exécution d'une opération. La première et le plus courant est la redevance intrinsèque au calcul de l'opération (voir annexe [G).](#_bookmark131) Deuxièmement, le gaz peut être déduit dans Commande pour formulaire la Paiement pour un subalterne

1. KEC ( *I* **b** ) = ***σ*** [ *c* ] c

Ce est assumé que la client sera avoir stockée la paire ( KEC ( *je* **b** ) *, je* **b** ) à quelques indiquer avant dans Commande pour Fabriquer la détermination de *je* **b** réalisable.

Comme pouvez être vu, là sont neuf exceptions pour la usage du cadre général d'exécution Ξ pour l'évaluation des appel de messages : ce sont des contrats dits 'précompilés', conçu comme une pièce préliminaire de l'architecture qui peut plus tard deviennent *des extensions natives* . Les contrats aux adresses 1 à 9 exécuter la fonction de récupération de clé publique à courbe elliptique, le schéma de hachage SHA2 256 bits, le hachage RIPEMD 160 bits schéma, la fonction d'identité, précision arbitraire modulaire exponentiation, addition de courbe elliptique, scalaire de courbe elliptique multiplication, un elliptique courbe jumelage Chèque, et la BLAKE2 fonction de compression F respectivement. Leur plein définition formelle est dans l'annexe [E.](#_bookmark119) Nous désignons l'ensemble des la adresses de la précompilé contrats par *π* :

(133 *)* ≡ { 1 *,* 2 *,* 3 *,* 4 *,* 5 *,* 6 *,* 7 *,* 8 *,* 9 }

* 1. Exécution Modèle

Le modèle d'exécution spécifie comment l'état du système est modifié étant donné une série d'instructions de bytecode et un petit tuple de environnemental Les données. Cette est spécifié par un

appel de message ou création de contrat ; cela fait partie du Paiement pour CRÉER , CRÉER2 , APPEL et CODE D'APPEL . Pour terminer, gaz mai être payé exigible pour un augmenter dans la usage de la Mémoire.

Terminé un comptes exécution, la total frais pour Mémoire- usage payable est proportionnel pour le plus petit plusieurs de 32 octets que sont obligatoire tel que tout Mémoire indices (qu'il s'agisse pour lis ou alors écrivez) sont inclus dans la Portée. Cette est payé pour sur un juste à temps base; comme tel, référencement un région de Mémoire à moins 32 octets plus grand que quelconque pré- une mémoire correctement indexée se traduira certainement par une Mémoire usage frais. Exigible pour cette frais ce est très peu probable adresses sera déjà aller au dessus de 32 bits bornes. Que mentionné, implémentations doit être capable pour faire en sorte cette éventualité. Stockage frais avoir un légèrement nuancé comportement - à dans- centiviser minimisation de la utiliser de stockage (qui cor- répond directement pour un plus grande Etat base de données sur tout nœuds), la exécution frais pour un opération que efface un entrée dans la stockage est ne pas seul renoncé, un qualifié rembourser est donné; dans fait, cette rembourser est effectivement payé à l'avant puisque la initial usage de un stockage emplacement frais substantiellement Suite

que Ordinaire usage.

Voir annexe [H](#_bookmark139) pour un rigoureux définition de la EVM gaz Coût.

ÉTHERUM : UN SÉCURISE DÉCENTRALISÉ GÉNÉRALISÉ TRANSACTION COMPTABLE **BERLIN VERSION** 13

* 1. **Environnement d'exécution.** En plus du système état tem ***σ*** , le gaz restant pour le calcul *g* , et le sous- état accumulé *A* , il existe plusieurs éléments importants informations utilisées dans l'environnement d'exécution que l'agent d'exécution doit fournir ; ceux-ci sont contenus dans le tuple *Je* :
     + *je* un , la adresse de la Compte qui possède la code que est exécution.
     + *je* o , la expéditeur adresse de la transaction que origine- iné cette exécution.
     + *je* p , la le prix de gaz dans la transaction que d'origine- noué cette exécution.
     + *je* , **\_** la octet déployer que est la saisir Les données pour cette exécution; si la exécution agent est un transaction,

devrait arrêt.

(135) Ξ( ***σ*** *, g, UN, je* ) ≡ ( ***σ****,* ***µg*** *,* \_ *UN,* **o** )

(136) ( ***σ****,* ***µ****, UN, ...,* **o** ) ≡ *X* ( ***σ*** *,* ***µ*** *, UN, je* )

|  |  |  |  |
| --- | --- | --- | --- |
| (139) | ***µ*m** | ≡ | (0*,* 0*, ...*) |
| (140) | ***µ***i | ≡ | 0 |
| (141) | ***µ*s** | ≡ | () |
| (142) | ***µ*o** | ≡ | () |
| (143) |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| (137) | ***µg*** \_ | ≡ | *g* |
| (138) | ***µpc*** \_ | ≡ | 0 |

  ∅ *,* ***µ*** *, UN, je ,* ∅ si *Z* ( ***σ*** *,* ***µ*** *, UN, je* )

cette voudrais être la transaction Les données.

* + - *je* suis , la adresse de la Compte qui causé la

*X* ( ***σ*** *,* ***µ*** *, UN, je* )

≡ ****  ∅ *,* ***µ****, UN, je ,* **o** si *w* =

REVENIR

**** ·

code pour être exécuter ; si la exécution agent est un transaction, cette voudrais être la transaction expéditeur.

* + - *je* v , la valeur, dans Wei, passé pour cette Compte comme partie de la même procédure comme exécution; si la

l'agent d'exécution est une transaction, ce serait le transaction valeur.

* + - *je* **b** , la octet déployer que est la machine code pour être

réalisé.

* + - *Je* H , la bloc entête de la cadeau bloc.

où (144)

(145)

(146)

*O* ( ***σ*** *,* ***µ*** *, UN, je* ) **o** si **o** = ∅

 *X O* ( ***σ*** *,* ***µ*** *, A, je* ) autrement

**o** ≡ *H* ( ***µ*** *, je* )

( *un, b, c, ré* ) · *e* ≡ ( *un, b, c, ré, e* )

***µ*** ≡ ***µ*** à l'exception:

* + - *je* e , la profondeur de la cadeau message-appel ou alors

création de contrat (c'est à dire la nombre de APPELS \_ ou alors

CRÉER(2) s être réalisé à cadeau).

(147)

***µg*** \_

≡ ***µg*** \_

* *C* ( ***σ*** *,* ***µ*** *, UN, je* )
  + *je* w , la autorisation pour Fabriquer modifications pour la

Etat.

Le modèle d'exécution définit la fonction Ξ, qui peut calculer l'état résultant ***σ***, le gaz restant *g*, la sous- état accumulé résultant *A* et la sortie résultante, **o** , compte tenu de ces définitions. Pour le contexte actuel, nous allons définir ce comme:

(134) ( ***σ****, g, UN,* **o** ) ≡ Ξ( ***σ*** *, g, UN, je* )

où nous sera rappelles toi que *Un* , la accumulé sous-état, est défini dans section [6.1.](#_bookmark36)

* 1. **Présentation de l'exécution.** Il faut maintenant définir le Ξ une fonction. Dans la plupart des implémentations pratiques, ce sera modelé comme un itératif progression de la paire comprenant l'état du système complet, ***σ*** et l'état de la machine, ***µ*** . Pour- mally, nous le définissons récursivement avec une fonction *X* . Cette les usages un itérateur une fonction *O* (qui définit la résultat de un

Noter que, lorsque nous évaluer Ξ, nous laissez tomber la Quatrième

élément *je* et extrait la restant gaz ***µg*** \_ depuis la résultant machine Etat ***µ***.

*X* est donc cyclé (récursivement ici, mais les implémentations sont généralement censés utiliser une simple boucle itérative) jusqu'à ce que soit [*Z*](#_bookmark59)devient vrai indiquant que l'état actuel est exceptionnel et que la machine doit être arrêtée et tout changements mis au rebut ou alors jusqu'à [*H*](#_bookmark60)devient un séries (plutôt que l'ensemble vide) indiquant que la machine a atteint un contrôlé arrêt.

* + 1. *Machine État.* Le machine Etat ***µ*** est défini comme la tuple ( *g, ordinateur personnel,* **moi** *, je,* **s** ) qui sont la gaz disponible, la

compteur de programme *pc* ∈ N 256 , le contenu de la mémoire, la actif Numéro de mots dans Mémoire (compte en continu

à partir de la position 0), et le contenu de la pile. La mémoire Contenu ***µ*** sont un séries de des zéros de Taille 2 256 .

**m**

Pour la facilité de en train de lire, la instruction mnémotechnique,

écrit en petites majuscules (par exemple ADD ), doit être interprété comme leur numérique équivalents ; la plein table de des instructions et leur détails est donné dans annexe [H](#_bookmark139)

Pour la fins de définir *Z* , *H* et *Ô* , nous définir *w*

comme la courant opération pour être réalisé:

Célibataire cycle de la Etat machine) ensemble avec les fonctions

[*Z*](#_bookmark59)qui détermine si la cadeau Etat est un [exceptionnel](#_bookmark59)

(148) *w* ≡

f *je* **b** [ ***µ*** p c

] si ***µpc*** \_

*<* ll *je* **b** ll

[arrêt](#_bookmark59) de la machine et [*H*](#_bookmark60) , spécifiant la sortie données de l'instruction si et seulement si l'état actuel est un [Ordinaire s'arrêter](#_bookmark60) Etat de la machine.

La séquence vide, notée (), n'est pas égale à la ensemble vide, noté ∅ ; c'est important lors de l'interprétation la sortie de *H* , qui s'évalue à ∅ lorsque l'exécution doit Continuez mais un séries (potentiellement vider) lorsque exécution

[ARRÊTER](#_bookmark143)  sinon

Nous supposons également les quantités fixes de *δ* et *α* , en spécifiant la pile éléments supprimé et ajoutée, tous les deux inscriptible sur l'instruction et une fonction de coût d'instruction *C* évaluant uating pour la plein Coût, dans gaz, de exécution la donné instruction.

ÉTHERUM : UN SÉCURISE DÉCENTRALISÉ GÉNÉRALISÉ TRANSACTION COMPTABLE **BERLIN VERSION** 14

* + 1. *Exceptionnel Arrêt.* La fonction d'arrêt exceptionnelle tion *Z* est défini comme:

(149)

*Z* ( ***σ*** *,* ***µ*** *, UN, je* ) ≡ ***µg*** *\_ < C* ( ***σ*** *,* ***µ*** *, UN, je* ) ∨

9.4.4. *Arrêt normal.* La fonction d'arrêt normale *H* est défini :

(154)



  *H* RETOUR ( ***µ*** ) \_ si *w* ∈ { [RETOUR](#_bookmark149) *,* REVE R T }

*δ w* = ∅ ∨ ll ***µ* s** ll *< δw \_* ∨

( *w* = SAUTER ∧ ***µs [* 0** ] ∈ */ D* ( *je* **b** )) ∨

*H* ( ***µ*** *, je* ) ≡

() si *w* ∈ { [ARRÊTER](#_bookmark143) *,* [AUTODESTRUCTION](#_bookmark152) }

 ∅ sinon



où

( *w* = JUMPI ∧ ***µs [* 1** ] = 0 ∧

***µs [* 0** ] ∈ */ D* ( *je* **b** )) ∨

( *w* = RETOURDATACOPIE ∧

***µ* s** [1] + ***µ* s** [2] *>* ll ***µ* o** ll ) ∨ ll ***µ* s** ll − *δ w* + *α w >* 1024 ∨ ( ¬ *je* w ∧ *O* ( *w,* ***µ*** )) ∨

( *w* = STORE ∧ ***µg*** *\_* Allocation d'appel *G* )

Les opérations d'arrêt de retour de données, [RETOURNER](#_bookmark149) et REVERT , ont une fonction spéciale *H* RETURN . A noter également la différence entre la séquence vide et la séquence vide Positionner comme discuté [ici.](#_bookmark58)

* 1. **Le Exécution Cycle.** Les éléments de la pile sont ajoutés ou supprimé de la partie la plus à gauche et la plus basse de l'index séries; tout autre éléments rester inchangé:

(150) *O* ( *w,* ***µ*** ) ≡ *w* ∈ { CRÉER *,* CRÉER2 *,* STORE *,*

AUTODESTRUCTION } ∨

LOG0 ≤ *w* ∧ *w* ≤ LOG4 ∨

*w* = APPEL ∧ ***µs* [** 2] = 0

(155)

(156)

(157)

(158)

*O* ( ***σ*** *,* ***µ*** *, UN, Je* ) ≡ ( ***σ****,* ***µ****, UN, je* )

∆ ≡ *αw \_* − *δw \_*

ll ***µ* s** ll ≡ ll ***µ* s** ll + ∆

∀ *x* ∈ [ *α w ,* ll ***µ* s** ll ) : ***µs*** [ *x* ] **\_** ≡ ***µs*** [ *x* **\_**− ∆]

Celui-ci précise que l'exécution est dans un arrêt exceptionnel. l'état de fonctionnement s'il n'y a pas assez de gaz, si l'instruction est invalide (et donc son indice *δ* est indéfini), s'il y a sont insuffisant pile éléments, si un SAUTER / SAUTER destination n'est pas valide, la nouvelle taille de la pile serait supérieure à 1024 ou Etat modification est tenté pendant un statique appel. Le comme-

Le gaz est réduit par la des instructions gaz Coût et pour la plupart des instructions, le compteur de programme s'incrémente sur chaque cycle, pour les trois exceptions, on suppose une fonction *J* , indice par une de deux des instructions, qui évalue pour la selon valeur:

Tut lecteur sera prendre conscience de que cette implique que non instruction

peut, à travers son exécution, cause un exceptionnel arrêt. Aussi, la exécution est dans un exceptionnel s'arrêter Etat si la gaz

(159)

***µ*** g ≡ ***µg*** \_ − *C* ( ***σ*** *,* ***µ*** *, UN, je* )

   [*J* SAUT P](#_bookmark147) ( ***µ*** ) si *w* = SAUTER

laissé avant l'exécution d'une instruction [SSTORE](#_bookmark145) est inférieur à ou alors égal pour la appel traitement [*G* appelle l'allocation](#_bookmark134) . Le dernière état a été introduit dans EIP-1706 par [Forshtat et Weiss [2019]](#_bookmark96)

(160)

***µ*** pc ≡

[*J* SAUTPI](#_bookmark148) ( ***µ*** ) si *w* = JUMPI

  *N* ( ***µ*** p c *, w* ) sinon

(partie de EIP-2200 par [Soie](#_bookmark105) [ [2019]).](#_bookmark105)

Dans général, nous présumer la Mémoire, accumulé sous-état

et système Etat faire ne pas monnaie:

* + 1. *Validité de la destination de saut.* Nous utilisions auparavant *D* comme fonction pour déterminer l'ensemble des destinations de saut valides. nations étant donné le code qui est exécuté. Nous définissons ceci comme quelconque position dans la code occupé par un JUMPDEST instruction.

(161)

(162)

(163)

(164)

***µ* m** ≡ ***µm* \_ *µ*** je ≡ ***µ*** je

*Un* ≡ *UN*

***σ*** ≡ ***σ***

Tous ces postes doivent être sur instruction valide lié- bélier, plutôt que de rester assis dans la partie données de PUSH opérations et doit apparaître dans le cadre explicitement défini portion de la code (plutôt que dans la implicitement défini [ARRÊT](#_bookmark143) opérations que Piste ce).

Officiellement:

(151) *ré* ( **c** ) ≡ *ré* J ( **c** *,* 0) où:

(152)

****

{} si *je* ;;: ll **c** ll

*ré* ( **c** *, je* ) { *je* } ∪ *DJ* ( c *,* **\_** *N* ( *je,* **c** [ *je* ]))

J

≡

si **c** [ *je* ] = JUMPDEST

****



*DJ* ( c *,* **\_** *N* ( *je,* **c** [ *je* ])) sinon

où *N* est la suivant valide instruction position dans la code, saut la Les données de un POUSSER instruction, si quelconque: (153)



  *je* + *w* − PUSH1 + 2

Cependant, les instructions modifient généralement un ou plusieurs composantes de ces valeurs. Composants modifiés répertoriés par instruction sont c'est noté dans annexe [H,](#_bookmark139) aux côtés de valeurs pour *α* et *δ* et un officiel la description de la gaz conditions.

* 1. Blocktree pour Chaîne de blocs

La blockchain canonique est un chemin de la racine à la feuille dans toute l'arborescence des blocs. Afin d'avoir un consensus terminé qui chemin ce est, conceptuellement nous identité la chemin qui a eu le plus de calculs effectués dessus, ou, le chemin *le plus lourd .* De toute évidence, un facteur qui aide à déterminer la le plus lourd chemin est la bloc Numéro de la feuille, équivalent pour la Numéro de blocs, ne pas compte la non miné [genèse](#_bookmark153) [bloquer,](#_bookmark153) dans le chemin. Plus le chemin est long, plus la total de l'effort minier qui doit avoir été fait pour arriver à la feuille. Cela s'apparente aux régimes existants, tels que que employé dans Dérivé de Bitcoin protocoles.

Puisqu'un en-tête de bloc inclut la difficulté, l'en-tête seule est suffisant pour valider la calcul Fini. Quelconque

*N* ( *je, w* ) ≡

∈

si *w* [ APPUYER1 *,* PUSH32 ]

  *je* + 1 sinon

bloc contribue vers la total calcul ou alors *total difficulté* de un chaîne.

ÉTHERUM : UN SÉCURISE DÉCENTRALISÉ GÉNÉRALISÉ TRANSACTION COMPTABLE **BERLIN VERSION** 15

comme:

Ainsi nous définir la total difficulté de bloc *B* récursivement

Si là sont collisions de la Bénéficiaire adresses être- interpolation ommers et la bloc (c'est à dire deux ommers avec la

(165)

(166)

*B* t  ≡ *B* t + *B* d *B* ≡ *P* ( *BH* ) \_

même Bénéficiaire adresse ou alors un ommer avec la même béné- adresse ficiaire comme bloc actuel), les ajouts sont appliqués cumulativement.

Comme tel donné un bloc *B* , *B* t est son total difficulté, *B* est son parent bloc et *B* d est son difficulté.



Nous définir la bloc récompense dans Wei : (176)

1. Bloc Finalisation

Le processus de finalisation un bloc implique quatre étapes:

* + - 1. Valider (ou alors, si exploitation minière, déterminer) ommers ;

Bloc *R* = 10 18 ×

5 si *Salut* \_ *< Fa* Byzance

3 si *Fa* Byzance *Salut <* \_ *F* Constantinople

  2 si *Salut* \_ ;;: *F* Constanti nople \_ \_

* + - 1. valider (ou alors, si exploitation minière, déterminer) transactions;
      2. appliquer récompenses;
      3. Vérifier (ou alors, si exploitation minière, calculer un valide) Etat et [bloc nonce.](#_bookmark17)

11.1. **Ommer Validation.** Le validation de [ommer](#_bookmark18) les [en-têtes](#_bookmark18) ne signifient rien de plus que de vérifier que chaque om- mer entête est tous les deux un valide entête et satisfait la relation

de *N* ème génération ommer pour la cadeau bloc où *N* ≤ 6.

Le maximum de ommer en-têtes est deux. Officiellement:

11.4. **État & Nonce Validation.** Nous mai à présent définir

la fonction, Γ, qui mappe un bloc *B* à son état d'initiation : (177)

f≡

Γ( *B* ) ***σ*** 0 si *P* ( *BH* ) \_ = ∅

***σ*** je : [TRIE](#_bookmark118) ( *L* S ( ***σ*** je )) = *P* ( *B* H ) H r autrement

Ici, TRIE ( *L* S ( ***σ*** je )) moyens la hacher de la racine nœud de un trie d'état ***σ*** i ; on suppose que les implémentations seront boutique cette dans la Etat base de données, qui est banal et efficace

puisque la essayer est par nature un immuable Les données structure.

(167) ll *B* **U**

ll 2

**U** ∈ *B* **U**

[*V* ( **U**](#_bookmark35) ) ∧ *k* ( **U** *, P* ( **B**

**H** ) **H**

*,* 6)

Et enfin nous définissons Φ, la fonction de transition de bloc, qui Plans un incomplet bloc *B* pour un Achevée bloc *B*:

où *k* dénote la "est-parent" biens:

(168)



(178) Φ( *B* ) ≡ *B* : *B* = *B* ∗ à l'exception:

*k* ( *U, H, n* ) ≡

*faux* si *n* = 0

*s* ( *U, H* )

(179)

*B* n  = *n* : *x*

2 256

*[Hd](#_bookmark31)* [\_](#_bookmark31)

  ∨ *k* ( *U , P* ( *H* ) H *, n* − 1) autrement

and *s* denotes the “is-sibling” property:

(180) *B* m = *m* avec ( *x, m* ) = PoW ( *B* [n ∗](#_bookmark34) *, n,* **ré** )

(181)

*B*∗ ≡ *B* except: [*B*r∗](#_bookmark21) = [*r*](#_bookmark128)[(Π(Γ(](#_bookmark67)*B*)*, B*))

(169)

*s* ( *U , H* ) ≡ ( *P* ( *H* ) = *P* ( *U* ) ∧ *H U* ∧ *tu* ∈ */ B* ( *H* ) **U** )

où *B* ( *H* ) et *P* ( *H* ) sont la bloc et la parent bloc de la correspondant entête *H* respectivement.

* 1. **Transaction Validation.** Le donné **gazUtilisé** doit correspondre fidèlement aux transactions répertoriées : [*B* Hg](#_bookmark14) , la total gaz utilisé dans la bloc, doit être égal pour la accumulé gaz utilisé selon pour la final transaction:

(170) [*B* H](#_bookmark14) g = [*.€*](#_bookmark2) ( [**R** ) u](#_bookmark22)

* 1. **Demande de récompense.** L'application des récompenses à un bloc consiste à relever le solde des comptes de l'adresse bénéficiaire du bloc et de chaque ommer par un un certain montant. Nous élevons le compte bénéficiaire du bloc par \_ [bloc *R*](#_bookmark62) ; [\_ \_ \_](#_bookmark62) pour chacun \_ \_ ommer, nous \_ augmenter la bl o c k's b bénéficiaire d'un complément 1 de la récompense globale et du bénéfice ciaire de l'ommer est récompensé en fonction du bloc nombre . Officiellement nous définir la une fonction Ω :

32

(171)

Ω( *B ,* ***σ*** ) ≡ ***σ*** : ***σ*** = ***σ*** sauf :

Avec **ré** être un base de données comme spécifié dans annexe [J](#_bookmark154)

Comme précisé au début du présent travail, [Π](#_bookmark67) est la fonction de transition d'état, qui est définie en termes de [Ω,](#_bookmark65) la fonction de finalisation du bloc et [Υ,](#_bookmark0) la transaction- évaluation une fonction, tous les deux à présent bien défini.

Comme précédemment détaillé, **R** [ *n* ] z , **R** [ *n* ] **l** et **R** [ *n* ] u sont le *n* ième code d'état correspondant, les journaux et le gaz cumulé utilisé après chacun \_ \_ transaction ( **R** [ *n* ] b , il \_ Quatrième com p one n t dans le tuple, a déjà été défini en termes de logs). Nous définissons également le *n* ième état ***σ*** [ *n* ], qui est défini simplement comme la Etat résultant depuis appliquer la correspondant

transaction à l'état résultant de la transaction précédente. action (ou l'état initial du bloc dans le cas du premier tel transaction):

f

(182) ***σ*** [ *n* ] = [Γ(](#_bookmark64) *B* ) si *n <* 0

[Υ(](#_bookmark0) ***σ*** [ *n* − 1] *, B* **T** [ *n* ]) autrement

Dans la Cas de *B* **R** [ *n* ] u , nous \_ prendre \_ \_ un similaire approche \_ \_ définir \_ chaque Objet comme la gaz utilisé dans évaluer la correspondre- ing transaction sommé avec la précédent Objet (ou alors zéro, si ce est la première), donnant nous un course total:

(172) ***σ*** [ **B** ] = ***σ*** [ **B** ] + ( 1 + ll **B U** ll *R*

Hc b

Hc b

32

block

(183) **R**[*n*]

=

Υ*g*(***σ***[*n* − 1]*, B*

[*n*])

   0 si *n <* 0

(173)

u

**T**

∀ **U** ∈ **BU \_** :

  + **R** [ *n* − 1] tu sinon

f ∅ si ***σ*** [ **U** ] = ∅ ∧ *R* = 0

Pour **R** [ *n* ] **l** , nous utiliser la [Υ](#_bookmark42) une fonction que nous conver-

***σ*** [ **U** c ] =

[**l**](#_bookmark42)

c

**un** autrement

gentiment défini dans la transaction exécution une fonction.

(184) **R** [ *n* ] **l** = Υ **l** ( ***σ*** [ *n* − 1] *, B* **T** [ *n* ])

(174) **un** ≡ ( ***σ*** [ *U* c ] n *,* ***σ*** [ *U* c ] b + *R ,* ***σ*** [ *U* c ] **s** *,* ***σ*** [ *U* c ] c )

(175) *R* ≡ ( 1 + 1 ( *U* − *B* ) *R*

8 i Hi

bloc

Nous définir [**R** [ *n* ] z](#_bookmark42) dans un similaire manière.

(185) **R** [ *n* ] z = Υ z ( ***σ*** [ *n* − 1] *, B* **T** [ *n* ])

ÉTHERUM : UN SÉCURISE DÉCENTRALISÉ GÉNÉRALISÉ TRANSACTION COMPTABLE **BERLIN VERSION** 16

Enfin, nous définissons Π comme le nouvel état étant donné le bloc re- fonction ward [Ω](#_bookmark65) appliquée à la résultante de la transaction finale Etat, [*.€*](#_bookmark2) ( ***σ*** ):

(186) Π( ***σ*** *, B* ) ≡ Ω( *B , .€* ( ***σ*** ))

Ainsi, le mécanisme complet de transition de bloc est dé- une amende, à l'exception pour PoW , la preuve de travail une fonction.

11.5. **Exploitation minière Preuve de travail.** Le exploitation minière preuve de- travail (PoW) existe comme un cryptographiquement sécurise nonce que prouve au-delà raisonnable doute que un particulier quantité de calcul a été dépensée dans la détermination mination d'une valeur de jeton *n* . Il est utilisé pour faire respecter la chaîne de blocs Sécurité par donnant sens et foi pour la notion de difficulté (et, par extension, total dif- difficulté). Cependant, depuis l'extraction de nouveaux blocs vient avec une récompense attachée, la preuve de travail ne fonctionne pas seulement comme un méthode de sécurisation confiance que la chaîne de blocs restera canonique dans le futur, mais aussi comme une richesse Distribution mécanisme.

Pour ces deux raisons, il y a deux objectifs importants du fonction de preuve de travail ; premièrement, il doit être aussi accessible que possible au plus grand nombre. L'exigence de, ou récompense de, matériel spécialisé et peu commun devrait être minimisé. Cela rend le modèle de distribution comme ouvrir comme possible, et, idéalement, fait du la loi de exploitation minière un échange simple de l'électricité à l'éther à peu près au même taux pour personne environ la monde.

Deuxièmement, ce devrait ne pas être possible pour Fabriquer super-linéaire profits, et surtout pas avec une barrière initiale élevée. Tel un mécanisme permet un bien financé adversaire pour Gain un gênant montant de la du réseau total exploitation minière Puissance et comme tel donne eux un super-linéaire récompense (Donc biaiser Distribution dans leur favoriser) comme Bien comme réduire la réseau Sécurité.

L'un des fléaux du monde Bitcoin est les ASIC. Ceux-ci sont pièces spécialisées de matériel de calcul qui n'existent que pour faire une seule tâche [(Smith [1997]).](#_bookmark98) Dans le cas de Bitcoin, la tâche est la fonction de hachage SHA256 [(Courtois et al. [2014]).](#_bookmark84) Tandis que Les ASIC existent pour une fonction de preuve de travail, les deux objectifs sont mis en danger. Pour cette raison, une fonction de preuve de travail résistant aux ASIC (c'est-à-dire difficile ou économiquement inefficace à mettre en œuvre dans du matériel de calcul spécialisé) possède a été identifié comme la proverbial argent balle.

Deux directions exister pour ASIC résistance; d'abord Fabriquer il séquentiel à mémoire dure, c'est-à-dire concevoir la fonction telle que la détermination de la [nonce](#_bookmark17) a besoin un terrain de mem- ory et bande passante tel que la Mémoire ne peux pas être utilisé en parallèle pour découvrir plusieurs nonces simultanément. Le deuxième est de faire le type de calcul dont il aurait besoin pour faire à usage général ; la signification de « matériel spécialisé » pour un polyvalent tâche Positionner est, naturellement, général but Matériel et comme tel produit de base bureau des ordinateurs sont probable pour être jolie proche pour "spécialisé Matériel" pour la tâche. Pour Ethereum 1.0 nous avoir choisi la première chemin.

Suite officiellement, la preuve de travail une fonction prend la formulaire de PoW :

(187)

256

2

*m* = [*H* m](#_bookmark16)  ∧ *n* [*H*](#_bookmark31) avec ( *moi, n* ) = PoW ( [*H* n](#_bookmark34) *, H* n *,* **ré** )

[d](#_bookmark31)

Où *H* n est l'en-tête du nouveau bloc mais *sans* la composants nonce et mix-hash ; *H* n est le nonce du entête; **d** est un grand ensemble de données nécessaire pour calculer le mix- Hacher et *Hd* \_ est la Nouveau bloc de difficulté valeur (c'est à dire la

bloquer la difficulté de la section [10).](#_bookmark61) PoW est la preuve de travail fonction qui évalue un tableau avec le premier élément étant le mixHash et le deuxième élément étant un pseudo- nombre aléatoire cryptographiquement dépendant de *H* et **d** . L'algorithme sous-jacent s'appelle Ethash et est décrit au dessous de.

11.5.1. *Éthash.* Éthash est la PoW algorithme pour Ethereum

1.0. C'est la dernière version de Dagger-Hashimoto, intro- réalisé par [Buterin](#_bookmark78) [[2013b]](#_bookmark78) et [Dryja](#_bookmark90) [[2014],](#_bookmark90) bien qu'il ne peut plus s'appeler ainsi puisque beaucoup de les caractéristiques originales des deux algorithmes ont été radicalement modifié avec R&D de février 2015 au 4 mai 2015 [(Jentzs ch](#_bookmark106) [[2015]).](#_bookmark106) L'itinéraire général que l'algorithme prend est comme suit :

Il existe une graine qui peut être calculée pour chaque bloc en parcourant les en-têtes de bloc jusqu'à ce que indiquer. A partir de la graine, on peut calculer un pseudo-aléatoire cachette, [*J* cacheinit](#_bookmark155) octets en taille initiale. Magasin clients légers le cache. De la cachette, nous pouvons générer un jeu de données, [*J* datasetinit](#_bookmark156) octets en taille initiale, avec la propriété que chaque l'élément de l'ensemble de données ne dépend que d'un petit nombre de éléments de la cache. Les clients complets et les mineurs stockent le base de données. Le base de données grandit linéairement avec temps.

L'exploitation minière consiste à saisir des tranches aléatoires de l'ensemble de données et les hacher ensemble. La vérification peut se faire avec mémoire faible en utilisant le cache pour régénérer le éléments de l'ensemble de données dont vous avez besoin, vous n'avez donc qu'à boutique la cache. Le grande base de données est mis à jour une fois tous

[*J* époque](#_bookmark157) blocs, donc le vaste majorité de un mineur l'effort sera être en train de lire la base de données, ne pas fabrication changements pour ce. Le

les paramètres mentionnés ainsi que l'algorithme sont expliqués dans détail dans annexe [J](#_bookmark154)

1. Exécution Contrats

Il existe plusieurs modèles d'ingénierie contractuelle qui permettre des comportements utiles particuliers ; deux d'entre eux que nous sera brièvement discuter sont Les données flux et Aléatoire Nombres.

* 1. **Flux de données.** Un contrat de flux de données est un contrat qui pro- fournit un service unique : il donne accès aux informations de le monde extérieur au sein d'Ethereum. La précision et l'actualité de ces informations n'est pas garantie et il est la tâche d'un auteur de contrat secondaire - le contrat qui utilise la Les données nourrir—à déterminer comment beaucoup confiance pouvez être mis dans quelconque Célibataire Les données nourrir.

Le schéma général implique un contrat unique au sein de Ethereum qui, lorsqu'il reçoit un appel de message, répond avec quelques opportun information concernant un externe phénomène- non. Un exemple pourrait être la température locale de New York. Cela serait mis en œuvre sous la forme d'un contrat qui renvoyé cette valeur d'un point connu dans le stockage. De bien sûr ce point de stockage doit être maintenu avec le corriger une telle température, et donc la deuxième partie de la modèle voudrais être pour un externe serveur pour Cours un Ethereum nœud, et immédiatement sur Découverte de un Nouveau bloc, crée une nouvelle transaction valide, envoyée au contrat, mettant à jour ladite valeur en stockage. Le code du contrat accepterait de telles mises à jour seul depuis la identité contenu sur mentionné serveur.

* 1. **Nombres aléatoires.** Fournir des nombres aléatoires au sein d'un système déterministe est naturellement impossible tâche. Cependant, on peut approximer avec un pseudo-aléatoire Nombres par utilisant Les données qui est en général inconnaissable

ÉTHERUM : UN SÉCURISE DÉCENTRALISÉ GÉNÉRALISÉ TRANSACTION COMPTABLE **BERLIN VERSION** 17

au moment de la transaction. Ces données pourraient inclure le le hachage du bloc, l'horodatage du bloc et le bénéfice du bloc adresse ciaire. Afin de rendre la tâche difficile aux mineurs malveillants pour contrôler ces valeurs, il faut utiliser le [BLOCKHASH](#_bookmark144) opération dans Commande pour utiliser hachages de la précédent 256 blocs sous forme de nombres pseudo-aléatoires. Pour une série de tels nombres, une solution triviale serait d'ajouter une quantité constante et hachage la résultat.

1. Futur les directions

La base de données d'état ne sera pas obligée de conserver toutes les données passées Etat [essayer](#_bookmark118) structures dans la futur. Ce devrait maintenir un âge pour chaque nœud et finalement Jeter nœuds que ne sont ni assez récents ni points de contrôle. Points de contrôle, ou un ensemble de nœuds dans la base de données qui permettent à un trie d'état du bloc à traverser, pourrait être utilisé pour placer un limite maximale de la quantité de calcul nécessaire dans Commande pour récupérer quelconque Etat tout au long de la chaîne de blocs.

La consolidation de la blockchain pourrait être utilisée afin de re- duc la montant de blocs un client voudrais avoir besoin pour Télécharger pour agir comme un nœud minier complet. Une archive compressée du structure triée à des moments donnés (peut-être un sur chaque 10 000e bloc) pourrait être maintenu par le réseau de pairs, refondre efficacement le [bloc de genèse.](#_bookmark153) Cela réduirait le montant à télécharger sur une seule archive plus un dur maximum limite de blocs.

Enfin, la compression de la blockchain pourrait peut-être être con- canalisé : nœuds dans l'état trie qui n'ont pas envoyé/reçu un transaction dans une quantité constante de blocs pourrait être jeté dehors, réduire tous les deux Fuite d'éther et la croissance de la Etat base de données.

13.1. **Évolutivité. L'évolutivité reste une** préoccupation éternelle . Avec un généralisé Etat passage une fonction, ce devient dif- difficile de partitionner et de paralléliser les transactions pour appliquer la stratégie de diviser pour mieux régner. Sans réponse, la dynamique la plage de valeurs du système reste essentiellement fixe et la moyen transaction valeur augmente, la moins de valeur de ils deviennent ignorés, étant économiquement inutile d'in- inclure dans le grand livre principal. Cependant, plusieurs stratégies existent qui peuvent potentiellement être exploitées pour fournir un Suite évolutif protocole.

Une certaine forme de structure hiérarchique, réalisée soit par consolider les petites chaînes plus légères dans le principal bloc ou alors immeuble la principale bloc par la incrémental- combinaison et adhésion (par preuve de travail) de des ensembles de transactions plus petits peuvent permettre la parallélisation des trans- combinaison d'action et construction de blocs. Le parallélisme pourrait proviennent également d'un ensemble prioritaire de chaînes de blocs parallèles , consolider chaque bloc et avec dupliquer ou alors invalide transactions jeté dehors par conséquent.

Enfin, un calcul vérifiable, s'il est fait généralement capable et suffisamment efficace, peut fournir un itinéraire pour permettre au preuve de travail pour être la vérification de final Etat.

1. Conclusion

Nous avons introduit, discuté et formellement défini la protocole d'Ethereum. Grâce à ce protocole, le lecteur peut implémenter un nœud sur le réseau Ethereum et rejoindre d'autres dans un système d'exploitation social sécurisé décentralisé. Contrats mai être auteur dans Commande pour algorithmiquement spécifier et de manière autonome imposer règles de interaction.

1. Remerciements

Beaucoup Merci pour Aéron Buchanan pour création la Révisions de *Homestead* , Christoph Jentzsch pour l'auteur du Algorithme Ethash et Yoichi Hirai pour avoir fait la plupart des Modifications EIP-150. Entretien important, correction utile tions et suggestions ont été à condition de par un Numéro de autres de l'organisation Ethereum DEV et Ethereum com- m uni t y un t grande y compris Rafale a v Simonson, P a w e l w Bylica, Jutta Steiner, Nick \_ \_ Épargnants , \_ \_ Viktor T r ´ on, Marquez \_ \_ Sim o vic, Giacomo Tazzari et, de cours, Vitalik Butérine.

1. Disponibilité

Le la source de cette papier est entretenu à [https:](https://github.com/ethereum/yellowpaper/)

[//github.com/ethereum/yellowpaper/](https://github.com/ethereum/yellowpaper/) . Un auto- Le PDF généré se trouve sur [https://ethereum.github.](https://ethereum.github.io/yellowpaper/paper.pdf) [io/yellowpaper/paper.pdf](https://ethereum.github.io/yellowpaper/paper.pdf) .

Références

Jacob Aron. Le logiciel BitCoin trouve une nouvelle vie. *Nouveau scientifique* , 213(2847):20, 2012. URL [http://www.sciencedirect.](http://www.sciencedirect.com/science/article/pii/S0262407912601055) [com /science/article/pii/S0262407912601055](http://www.sciencedirect.com/science/article/pii/S0262407912601055) .

Adam Arrière. Haschcash - Amortissable Publiquement Auditable

Fonctions de coût, 2002. URL [http://www.hashcash.org/](http://www.hashcash.org/papers/amortizable.pdf) [papiers/amortissable.pdf](http://www.hashcash.org/papers/amortizable.pdf) .

Tim Beiko, James Hancock et Thomas Jay Rush. EIP- 4345 : Retard de la bombe de difficulté à juin 2022, octobre 2021a. URL <https://eips.ethereum.org/EIPS/eip-4345>.

Tim Beiko et Al. Berlin réseau améliorer spécifique-

tion, 2021b. URL [https://github.com/ethereum/](https://github.com/ethereum/eth1.0-specs/blob/master/network-upgrades/mainnet-upgrades/berlin.md) [eth1.0-specs/blob/master/network-upgrades/](https://github.com/ethereum/eth1.0-specs/blob/master/network-upgrades/mainnet-upgrades/berlin.md) [mainnet-upgrades/berlin.md](https://github.com/ethereum/eth1.0-specs/blob/master/network-upgrades/mainnet-upgrades/berlin.md) .

Guide Bertoni, Jeanne Démons, Michal Peter, et Gilles Van Assche. Le KECCAK SHA-3 sous- mission, 2011. URL [https://keccak.team/files/](https://keccak.team/files/Keccak-submission-3.pdf)

[Keccak-submission-3.pdf](https://keccak.team/files/Keccak-submission-3.pdf) .

romain Boutellier et mareike Heinzen. Pirates, Pio- neers, innovateurs et imitateurs. En *croissance jusqu'à Innover* , pages 85–96. Springer, 2014. URL [https:](https://www.springer.com/gb/book/9783319040158)

[//www.springer.com/gb/book/9783319040158](https://www.springer.com/gb/book/9783319040158) .

Vitalik Buterin. Ethereum : Une Smart de nouvelle génération Contrat et Décentralisé Application Plateforme,

2013a. URL [https://github.com/ethereum/wiki/](https://github.com/ethereum/wiki/wiki/White-Paper) [wiki/Livre blanc](https://github.com/ethereum/wiki/wiki/White-Paper) .

Vitalik Buterin. Dagger : une mémoire difficile à calculer, Memory-Easy to Verify Scrypt Alternative, 2013b. URL <http://www.hashcash.org/papers/dagger.html>.

Vitalik Butérine. EIP-2 : Propriété hard-fork changements,

2015. URL <https://eips.ethereum.org/EIPS/eip-2>. Vitalik Butérine. EIP-100 : Changer difficulté ajustement pour cibler moyenne bloc temps y compris oncles, Avril 2016a.

URL <https://eips.ethereum.org/EIPS/eip-100>. Vitalik Butérine. EIP-155 : Simple rejouer attaque protéger

tion, Octobre 2016b. URL [https://eips.ethereu m.](https://eips.ethereum.org/EIPS/eip-155) [org/EIPS/eip-155](https://eips.ethereum.org/EIPS/eip-155) .

Vitalik Buterin. EIP-1014 : Skinny CREATE2, avril 2018. URL <https://eips.ethereum.org/EIPS/eip-1014>.

Vitalik Butérine et Martin Swende. EIP-2929 : Gaz Coût dans-

plis pour Etat accéder les opcodes, Septembre 2020a. URL

<https://eips.ethereum.org/EIPS/eip-2929>.

Vitalik Buterin et Martin Swende. EIP-2930 : Accompagnement en option listes d'accès, août 2020b. URL [https://eips.ethereum.](https://eips.ethereum.org/EIPS/eip-2930) [org/EIPS/eip-2930](https://eips.ethereum.org/EIPS/eip-2930) .

ÉTHERUM : UN SÉCURISE DÉCENTRALISÉ GÉNÉRALISÉ TRANSACTION COMPTABLE **BERLIN VERSION** 18

Éric Conner. EIP-2384 : Muir Glacier difficulté b omb retard, Novembre 2019. URL [https://eips.ethereum.](https://eips.ethereum.org/EIPS/eip-2384) [org/EIPS/eip-2384](https://eips.ethereum.org/EIPS/eip-2384) .

Nicolas T Courtoise, marek Grajek, et Rahul Naïk.

*Optimisation SHA256 dans Bitcoin Exploitation minière* , pages 131–

144. Springer Berlin Heidelberg, Berlin, Heidel- berg, 2014. ISBN 978-3-662-44893-9. est ce que je: dix. 1007/978-3-662-44893-9 12. URL [https://doi.org/10.](https://doi.org/10.1007/978-3-662-44893-9_12) [1007/978-3-662-44893-9\_12](https://doi.org/10.1007/978-3-662-44893-9_12) .

BA Davey et HA Prisley. *Introduction pour treillis et Commande. 2e éd.* Cambridge : Université de Cambridge Presse, 2e éd. édition, 2002. ISBN 0-521-78451-4/pbk. Thaddée Dryja. Hashimoto : E/S lié preuve de travail, 2014. URL [http://diyhpl.us/ ~ bryan/papers2/](http://diyhpl.us/~bryan/papers2/bitcoin/meh/hashimoto.pdf)

[bitcoin/meh/hashimoto.pdf](http://diyhpl.us/~bryan/papers2/bitcoin/meh/hashimoto.pdf) .

Cynthia Dwork et Moni Naour. Tarification passant par pro- arrêter ou combattre le courrier indésirable. Dans *In 12th An- nual International Cryptology Conference* , pages 139– 147, 1992. URL [http://www.wisdom.weizmann.ac.il/](http://www.wisdom.weizmann.ac.il/~naor/PAPERS/pvp.pdf)

[~ naor/PAPERS/pvp.pdf](http://www.wisdom.weizmann.ac.il/~naor/PAPERS/pvp.pdf) .

Dankrad Feist, Dmitri Khovratovitch, et Marius van der

Wijden. EIP-3607 : Rejeter transactions depuis expéditeurs avec déployé code, Juin 2021. URL [https://eips.](https://eips.ethereum.org/EIPS/eip-3607) [ethereum.org/EIPS/eip-3607](https://eips.ethereum.org/EIPS/eip-3607) .

Alexandre Forshtat et Yoav Weiss. EIP-1706 : Désactiver sstore

avec gaz gauche plus bas que appel traitement, 2019. URL [https:](https://eips.ethereum.org/EIPS/eip-1706)

[//eips.ethereum.org/EIPS/eip-1706](https://eips.ethereum.org/EIPS/eip-1706) .

Phong Vo Glenn Fowler, Landon Curt Noll. Fo wlerNol- Fonction de hachage lVo, 1991. URL [http://www.isthe.com/](http://www.isthe.com/chongo/tech/comp/fnv/index.html) [chongo/tech/comp/fnv/index.html](http://www.isthe.com/chongo/tech/comp/fnv/index.html) .

Nils Goura, Arun Patel, Arvinderpal Errer, Hans Eberle ,

et Sheueling Chang Shantz. Comparer la courbe elliptique cryptographie et RSA sur les processeurs 8 bits. En *cryptographie Matériel et Systèmes Embarqués-CHES 2004* , pages 119–132. Springer, 2004. URL [https://www.iacr.org/](https://www.iacr.org/archive/ches2004/31560117/31560117.pdf) [archive/ches2004/31560117/31560117.pdf](https://www.iacr.org/archive/ches2004/31560117/31560117.pdf) .

James Hancock. EIP-3554 : Difficile retard de la bombe à De- décembre 2021, mai 2021. URL [https://eips.ethereum.](https://eips.ethereum.org/EIPS/eip-3554) [org/EIPS/eip-3554](https://eips.ethereum.org/EIPS/eip-3554) .

Tjaden Hesse, Mat Luongo, Piotr Dyraga, et James

Hancock. EIP-152 : Ajouter la fonction de compression BLAKE2 tion 'F' précompiler, Octobre 2016. URL [https://eips.](https://eips.ethereum.org/EIPS/eip-152) [ethereum.org/EIPS/eip-152](https://eips.ethereum.org/EIPS/eip-152) .

Christophe Jentzsch. Commettre Date pour éthash, 2015. URL

[https://github.com/ethereum/yellowpaper/commit/](https://github.com/ethereum/yellowpaper/commit/77a8cf2428ce245bf6e2c39c5e652ba58a278666" \l "commitcomment-24644869) [77a8cf2428ce245bf6e2c39c5e652ba58a278666#](https://github.com/ethereum/yellowpaper/commit/77a8cf2428ce245bf6e2c39c5e652ba58a278666#commitcomment-24644869) [commitcomment-24644869](https://github.com/ethereum/yellowpaper/commit/77a8cf2428ce245bf6e2c39c5e652ba58a278666#commitcomment-24644869) .

Enfiler Johnson, Alfred Ménezès, et Scott Van- calcul. Le Elliptique Courbe Signalisation numérique- ture Algorithme (ECDSA), 2001. URL https:

//web.archive.org/web/20170921160141/http://

cs.ucsb.edu/ ~ koc/ccs130h/notes/ecdsa- cert.pdf . Accédé 21 Septembre 2017, mais la original lien a été

inaccessible sur 19 Octobre 2017. Faire référence à section 6.2 pour ECDSAPUBKEY, et section 7 pour ECDSASIGN et

ECDSARECOVER.

Sergio Démian Lerner. Stricte Mémoire Dur Hachage Fonction- tions, 2014. URL [http://www.hashcash.org/papers/](http://www.hashcash.org/papers/memohash.pdf) [memohash.pdf](http://www.hashcash.org/papers/memohash.pdf) .

Marquer Meunier. Le Futur de Droit. Dans *Pennsylvanie-*

*par livré à la conférence Extra 3 (août 9)* , 1997. URL [https://drive.google.com/file/d/](https://drive.google.com/file/d/0Bw0VXJKBgYPMS0J2VGIyWWlocms/edit?usp=sharing) [0Bw0VXJKBgYPMS0J2VGIyWWlocms/edit?usp=partage](https://drive.google.com/file/d/0Bw0VXJKBgYPMS0J2VGIyWWlocms/edit?usp=sharing) .

Satoshi Nakamoto. Bitcoin : UN d'égal à égal électronique espèces

système, 2008. URL [http://www.bitcoin.org/bitcoin.](http://www.bitcoin.org/bitcoin.pdf) [pdf](http://www.bitcoin.org/bitcoin.pdf) .

Meni Rosenfeld, Yoni Assie, Vitalik Butérine, m li- orhakiLior, Oded Leïba, Assaf Shomer, et Eli- couru Zach. Coloré Pièces de monnaie Protocole Spécification, 2012. URL [https://github.com/Colored-Coins/](https://github.com/Colored-Coins/Colored-Coins-Protocol-Specification)

[Spécification du protocole des pièces de couleur](https://github.com/Colored-Coins/Colored-Coins-Protocol-Specification) .

Markku-Juhani Saarinen et Jean Philippe Aumasson. RFC 7693 : Le hachage cryptographique BLAKE2 et mes- code d'authentification sage (MAC), novembre 2015. URL <https://tools.ietf.org/html/rfc7693>.

Afrique Schoedon. EIP-1234 : Constantinople difficulté bombe

retard et bloc récompense ajustement, 2018. URL [https:](https://eips.ethereum.org/EIPS/eip-1234)

[//eips.ethereum.org/EIPS/eip-1234](https://eips.ethereum.org/EIPS/eip-1234) .

Afri Schoedon et Vitalik Buterin. EIP-649 : Métropole- retard de bombe de difficulté de lis et réduction de récompense de bloc, Juin 2017. URL [https://eips.ethereum.org/EIPS/](https://eips.ethereum.org/EIPS/eip-649)

[eip-649](https://eips.ethereum.org/EIPS/eip-649) .

Michael John Sébastien Forgeron. *Spécifique à l'application Intégré Circuits* . Addison-Wesley, 1997. ISBN 0201500221.

Yonatan Sompolinsky et Aviv Zohar. Accélération peu- pièces de monnaie transaction En traitement. vite de l'argent grandit sur des arbres, ne pas Chaînes, 2013. URL [https://eprint.iacr.org/ \_](https://eprint.iacr.org/2013/881)

[2013/881](https://eprint.iacr.org/2013/881) .

Simon Sprankel. Technique Base de Numérique Devises, 2013. URL [http://www.coderblog.de/wp-content/](http://www.coderblog.de/wp-content/uploads/technical-basis-of-digital-currencies.pdf) [téléchargements/ base-technique-des-monnaies-numériques.](http://www.coderblog.de/wp-content/uploads/technical-basis-of-digital-currencies.pdf) [pdf](http://www.coderblog.de/wp-content/uploads/technical-basis-of-digital-currencies.pdf) .

pseudo Szabo. Formaliser et sécurisation relation-

navires sur le public réseaux. *Première Lundi* , 2(9), 1997. URL [http://firstmonday.org/ojs/index.php/](http://firstmonday.org/ojs/index.php/fm/article/view/548) [fm/article/vue/548](http://firstmonday.org/ojs/index.php/fm/article/view/548) .

Wei Soie. EIP-2200 : structuré définitions pour rapporter gaz moi-

tering, 2019. URL [https://eips.ethereum.org/EIPS/](https://eips.ethereum.org/EIPS/eip-2200) [eip-2200](https://eips.ethereum.org/EIPS/eip-2200) .

Vivek Vishnumurthy, Sangeeth Chandrakumar, et Emin Gn Sirer. KARMA : Un cadre économique sécurisé pour d'égal à égal Ressource partage, 2003. URL [https://](https://www.cs.cornell.edu/people/egs/papers/karma.pdf)

[www.cs.cornell.edu/people/egs/papers/karma.pdf](https://www.cs.cornell.edu/people/egs/papers/karma.pdf) .

J R Willette. MasterCoin Complet Spécification, 2013. URL <https://github.com/mastercoin-MSC/spec>.

Michée Zoltu. EIP-2718 : Tapé transaction enveloppe,

Juin 2020. URL [https://eips.ethereum.org/EIPS/](https://eips.ethereum.org/EIPS/eip-2718) [eip-2718](https://eips.ethereum.org/EIPS/eip-2718) .

annexe UN. Terminologie

**Acteur externe :** Une personne ou une autre entité capable de s'interfacer avec un nœud Ethereum, mais extérieure au monde de Ethereum. Il peut interagir avec Ethereum en déposant des transactions signées et en inspectant la blockchain et associé Etat. Possède une (ou alors Suite) intrinsèque Comptes.

**Adresse:** UN 160 bits code utilisé pour identifier Comptes.

**Compte:** Comptes avoir un intrinsèque solde et transaction compter entretenu comme partie de la Ethereum Etat. Ils aussi avoir quelques (peut-être vider) EVM Code et un (peut-être vider) Stockage État associé avec eux.

ÉTHERUM : UN SÉCURISE DÉCENTRALISÉ GÉNÉRALISÉ TRANSACTION COMPTABLE **BERLIN VERSION** 19

Bien qu'homogène, il est logique de distinguer deux types pratiques de comptes : ceux avec vide Code EVM associé (ainsi le solde du compte est contrôlé, le cas échéant, par une entité externe) et ceux avec Code EVM associé non vide (le compte représente donc un Objet Autonome). Chaque compte a un Célibataire Adresse que identifie ce.

**Transaction:** Une donnée, signée par un Acteur Externe. Il représente soit un Message soit une nouvelle Autonome Objet. Transactions sont enregistré dans chaque bloc de la chaîne de blocs.

**Objet autonome :** un objet fictif existant uniquement dans l'état hypothétique d'Ethereum. A une valeur intrinsèque adresse et donc un compte associé ; le compte aura un code EVM associé non vide. Incorporé seul comme la Stockage État de que Compte.

**État de stockage :** Les informations propres à un compte donné qui sont conservées entre les moments où le Comptes associé EVM Code court.

**Message:** Les données (sous la forme d'un ensemble d'octets) et la valeur (spécifiée comme Ether) qui sont transmises entre deux comptes, soit par le fonctionnement déterministe d'un Objet Autonome ou la signature cryptographiquement sécurisée du Transaction.

**Appel de message :** action de transmettre un message d'un compte à un autre. Si le compte de destination est associé avec non vide EVM Code, puis le La MV va démarrer avec le état de ledit Objet et le message a agi sur. Si l'expéditeur du message est un objet autonome, l'appel transmet toutes les données renvoyées par la VM opération.

**Gaz:** L'unité de coût de réseau fondamentale. Payé exclusivement par Ether (à partir de PoC-4), qui est converti librement en et du gaz selon les besoins. Le gaz n'existe pas en dehors du moteur de calcul interne d'Ethereum ; son prix est Positionner par la Transaction et mineurs sont libre pour ignorer Transactions à qui Gaz le prix est aussi bas.

**Contrat:** Terme informel utilisé pour désigner à la fois un morceau de code EVM qui peut être associé à un compte ou un Autonome Objet.

**Objet:** Synonyme pour Autonome Objet.

**Application :** Un visible par l'utilisateur final application hébergé dans la Ethereum Navigateur.

**Navigateur Ethereum :** (alias Ethereum Reference Client) Une interface graphique multiplateforme d'une interface similaire à une interface simplifiée navigateur (à la Chrome) capable d'héberger des applications en bac à sable dont le backend est purement sur l'Ethereum protocole.

**Ethereum Virtuel Machine:** (alias EVM) Le virtuel machine que formes la clé partie de la exécution maquette pour un Comptes associé EVM Code.

**Environnement d'exécution Ethereum :** (alias ERE) L'environnement fourni à un objet autonome s'exécutant dans l'EVM. Inclut l'EVM mais aussi la structure de l'état mondial sur laquelle s'appuie l'EVM pour certain E/S des instructions y compris APPEL & CRÉER.

**Code EMV :** Le bytecode que la EVM pouvez nativement exécuter. Utilisé pour officiellement spécifier la sens et ramifications de un message pour un Compte.

**EVM Assemblée:** Le lisible par l'homme formulaire de Code EVM.

**JE VAIS:** Le Lisp-like Niveau faible Langue, un inscriptible par l'homme Langue utilisé pour création Facile contrats et général niveau faible Langue boîte à outils pour trans-compilation pour.

annexe B Récursif Longueur Préfixe

Cette est un sérialisation méthode pour codage arbitrairement structuré binaire Les données (octet tableaux). Nous définir la Positionner de possible structures T :

(188)

(189)

(190)

J ≡ L \± B

L ≡ { **t** : **t** = ( **t** [0] *,* **t** [1] *, ...* ) ∧ ∀n *\_ <* ll **ll** ll : **t** [ *n* ] ∈ T }

B ≡ { **b** : **b** = ( **b** [0] *,* **b** [1] *, ...* ) ∧ ∀n *\_ <* ll **b** ll : **b** [ *n* ] ∈ O }

Où O est la Positionner de (8 bits) octets. Ainsi B est la Positionner de tout séquences de octets (autrement connu comme octet tableaux, et une feuille si on l'imagine comme un arbre), L est l'ensemble de toutes les (sous-)structures arborescentes qui ne sont pas une seule feuille (un nœud de branche si imaginé comme un arbre) et J est la Positionner de tout octet tableaux et tel de construction séquences. Le disjoint syndicat \± est avait besoin seul

pour distinguer la vider octet déployer () ∈ B depuis la vider liste () ∈ L , qui sont codé différemment comme défini au dessous de; comme

commun, nous sera abuser de notation et laisser la disjoint syndicat indices implicite, inférable depuis le contexte.

Nous définissons la fonction RLP comme RLP à travers deux sous-fonctions, la première gérant l'instance lorsque la valeur est un octet déployer, la seconde lorsque ce est un séquence de plus loin valeurs:

≡ f

(191) RLP ( **x** ) *R* b ( **x** ) si **X** ∈ B

*R* l ( **x** ) sinon

Si la valeur pour être sérialisé est un octet déployer, la RLP sérialisation prend une de Trois formes:

* Si la octet déployer contient uniquement un Célibataire octet et que Célibataire octet est moins que 128, alors la saisir est exactement égal pour la production.
* Si la octet déployer contient moins de que 56 octets, alors la production est égal pour la saisir préfixé par la octet égal pour la longueur de la octet déployer plus 128.

ÉTHERUM : UN SÉCURISE DÉCENTRALISÉ GÉNÉRALISÉ TRANSACTION COMPTABLE **BERLIN VERSION** 20

* Autrement, la production est égal pour la saisir, à condition de que ce contient moins de que 2 octets, préfixé par la

64

longueur minimale octet déployer qui lorsque interprété comme un big-endian entier est égal pour la longueur de la saisir octet

déployer, qui est lui-même préfixé par la Numéro de octets obligatoire pour fidèlement encoder cette longueur valeur plus 183.

Tableaux d'octets contenant 2 64 ou plusieurs octets ne peuvent pas être encodés. Cette restriction garantit que le premier octet de l'encodage de un octet déployer est toujours au dessous de 192, et Donc ce pouvez être facilement distingué depuis la encodages de séquences dans L. \_

Formellement , \_ \_ nous \_ définir *R* b :

(192)

*R* b ( **x** ) ≡

**x** si ll **x** ll = 1 ∧ **×** [0] *<* 128

(128 + ll **x** ll ) · **x** sinon si ll **x** ll *<* 56

**** 183 + BE(ll**x**ll) · BE(ll**x**ll) · **x** else if ll**x**ll *<* 264

****

∅ sinon



**b** − 1

(193)

ÊTRE ( *x* ) ≡ ( *b* 0 *, b* 1 *, ...* ) : *b* 0

0 ∧ *X* =

*n* =0

*b n* · 256 **b** − 1 − *n*

(194) ( *x* 1 *, ..., x n* ) · ( *y* 1 *, ..., y m* ) = ( *x* 1 *, ..., x n , y* 1 *, ..., y m* )

Ainsi ÊTRE est la une fonction que se développe un non négatif entier valeur pour un big-endian octet déployer de minimal longueur et la point opérateur effectue séquence enchaînement.

Si Au lieu, la valeur pour être sérialisé est un séquence de autre éléments alors la RLP sérialisation prend une de deux formes:

* Si la concaténé sérialisations de chaque contenu Objet est moins que 56 octets dans longueur, alors la production est égal pour que enchaînement préfixé par la octet égal pour la longueur de cette octet déployer plus 192.
* Autrement, la production est égal pour la concaténé sérialisations, à condition de que elles ou ils contenir moins de que 2 octets,

64

préfixé par la longueur minimale octet déployer qui lorsque interprété comme un big-endian entier est égal pour la longueur

de la concaténé sérialisations octet déployer, qui est lui-même préfixé par la Numéro de octets obligatoire pour fidèlement encoder cette longueur valeur plus 247.

Séquences à qui concaténé sérialisé éléments contenir 2 64 ou alors Suite octets ne peux pas être codé. Cette restriction assure que la première octet de la codage Est-ce que ne pas dépasser 255 (autrement ce voudrais ne pas être un octet).

Ainsi nous terminer par officiellement définir *R* l :

(192 + ll*s*(**x**)ll) · *s*(**x**) if *s*(**x**) = ∅ ∧ ll*s*(**x**)ll *<* 56

(195)

(196)

*R* l ( **x** ) ≡

*s*(**x**) ≡

247 + BE ( ll *s* ( **x** ) ll ) · BE ( ll *s* ( **x** ) ll ) · *s* ( **x** ) autre si *s* ( **x** ) = ∅ ∧ ll *s* ( **x** ) ll *<* 2 64

F

∅ otherwise

∅

RLP(**x**[0]) · RLP(**x**[1]) · *...* if ∀*i* : RLP(**x**[*i*])

∅ otherwise

Si RLP est utilisé pour encoder un scalaire, défini seul comme un non négatif entier (dans N , ou alors dans N *x* pour quelconque *x* ), ce doit être codé comme le tableau d'octets le plus court dont l'interprétation gros-boutiste est le scalaire. Ainsi, le RLP d'un entier non négatif *i* est défini comme:

(197) RLP ( *je* : *je* ∈ N ) ≡ RLP ( BE ( *i* ))

Lors de l'interprétation des données RLP, si un fragment attendu est décodé en tant que scalaire et que des zéros non significatifs sont trouvés dans l'octet séquence, les clients sont tenus de la considérer comme non canonique et de la traiter de la même manière que des données RLP autrement invalides, congédier ce totalement.

Là est non spécifique canonique codage format pour signé ou alors point flottant valeurs.

annexe C Préfixe hexadécimal Codage

Le codage par préfixe hexadécimal est une méthode efficace pour coder un nombre arbitraire de quartets sous la forme d'un tableau d'octets. Il est capable de stocker un indicateur supplémentaire qui, lorsqu'il est utilisé dans le contexte du trie (le seul contexte dans lequel il est utilisé), lève l'ambiguïté entre nœud les types.

Il est défini comme la fonction HP qui mappe à partir d'une séquence de quartets (représentés par l'ensemble Y ) avec un booléen valeur pour un séquence de octets (représentée par la Positionner B ):

(198)

PS ( **x** *, t* ) : **x** Y (16 *f* ( *t* ) *,* 16 **fois** [0] + **x** [1] *,* 16 **×** [2] + **x** [3] *, ...* ) si ll **x** ll est même

(16( *f* ( *t* ) + 1) + **x** [0] *,* 16 **fois** [1] + **x** [2] *,* 16 **fois** [3] + **x** [4] *, ...* ) autrement

f∈ ≡

(199)

*F* ( *t* ) ≡ f 2 si *t* 0

0 autrement

Ainsi, le quartet haut du premier octet contient deux drapeaux ; le bit le plus bas codant l'impair de la longueur et le deuxième codage le plus bas du drapeau *t* . Le quartet bas du premier octet est nul dans le cas d'un nombre pair de quartets et le premier quartet dans le cas d'un nombre impair. Tous les quartets restants (maintenant un nombre pair) s'insèrent correctement dans le reste octets.

ÉTHERUM : UN SÉCURISE DÉCENTRALISÉ GÉNÉRALISÉ TRANSACTION COMPTABLE **BERLIN VERSION** 21

annexe RÉ. Modifié Merkel Patricia Arbre

L'arbre Merkle Patricia modifié (trie) fournit une structure de données persistante pour mapper entre binaire de longueur arbitraire données (tableaux d'octets). Il est défini en termes de structure de données mutable pour mapper entre des fragments binaires de 256 bits et données binaires de longueur arbitraire, généralement implémentées sous forme de base de données. Le noyau du trie, et sa seule exigence en termes de la protocole spécification, est pour fournir un Célibataire valeur que identifie un donné Positionner de valeur clé paires, qui mai être Soit une séquence de 32 octets ou la séquence d'octets vide. Il est laissé comme une considération de mise en œuvre pour stocker et maintenir le structure de la essayer dans un manière que permet efficace et efficace la concrétisation de la protocole.

Officiellement, nous présumer la saisir valeur je , un Positionner contenant paires de octet séquences avec unique clés:

(200) je = { ( **k** 0 ∈ B *,* **v** 0 ∈ B ) *,* ( **k** 1 ∈ B *,* **v** 1 ∈ B ) *,* }

Lors de l'examen d'une telle séquence, nous utilisons la notation d'indice numérique commune pour faire référence à la clé ou à la valeur d'un tuple, Donc:

(201) ∀ *je* ∈ Je : *je* ≡ ( *je* 0 *, je* 1 )

Quelconque séries de octets mai aussi trivialement être vu comme un séries de grignote, donné un spécifique à endian notation; ici nous présumer big-endian. Ainsi:

(202)

(203)

∀n *\_* : ∀ *je <* 2 ll **k** *n*

*y* ( je ) = { ( **k** 0 ∈ Oui *,* **v** 0 ∈ B ) *,* ( **k** 1 ∈ Oui *,* **v** 1 ∈ B ) *, ...* }

**k** *n* [ *je* 2] 16si *je* est même

fl ÷ ÷ Jll : **k** [*i*] ≡

*n*

**k** *n* [ l *je* ÷ 2 J ] mod \_ \_ 16 autrement

Nous définissons la fonction TRIE , qui évalue à la racine du trie qui représente cet ensemble lorsqu'il est encodé dans ce structure:

(204) TRIE ( I ) ≡ KEC RLP ( *c* ( je *,* 0))

Nous supposons également une fonction *n* , la fonction de cap de nœud du trie. Lors de la composition d'un nœud, nous utilisons RLP pour coder le structure. Comme un moyens de réduire stockage complexité, nous boutique nœuds à qui composé RLP est moins de que 32 octets directement; pour les plus grands, nous affirmons la prescience du tableau d'octets dont le hachage Keccak-256 s'évalue à notre référence. Ainsi nous définissons dans termes de *c* , la nœud composition une fonction:

 () si je = ∅

(205) *n*(I*, i*) ≡ *c*(I*, i*) if ll RLP *c*(I*, i*) ll *<* 32



KEC RLP ( *c* ( je *, je* )) sinon



D'une manière similaire à un arbre de base, lorsque le trie est parcouru de la racine à la feuille, on peut construire une seule paire clé-valeur. La clé est accumulée tout au long du parcours, en acquérant un seul quartet à partir de chaque nœud de branche (tout comme avec un arbre de base). Contrairement à un base arbre, dans la Cas de plusieurs clés partage la même préfixe ou alors dans la Cas de un Célibataire clé ayant un unique suffixe, deux nœuds d'optimisation sont fournis. Ainsi, lors de la traversée, on peut potentiellement acquérir plusieurs quartets de chaque de la autre deux nœud les types, extension et feuille. Là sont Trois sortes de nœuds dans la essayer :

**Feuille :** une structure à deux éléments dont le premier élément correspond aux quartets dans la clé qui ne sont pas déjà pris en compte par le accumulation de clés et de branches parcourues depuis la racine. La méthode de codage par préfixe hexadécimal est utilisée et le seconde paramètre pour la une fonction est obligatoire pour être 1.

**Extension:** Une structure à deux éléments dont le premier élément correspond à une série de quartets de taille supérieure à un qui sont partagées par au moins deux clés distinctes passé le cumul des clés de quartets et des clés de branches comme traversé à partir de la racine. La méthode de codage par préfixe hexadécimal est utilisée et le deuxième paramètre de la fonction est obligatoire pour être 0.

**Branche :** une structure de 17 éléments dont les seize premiers éléments correspondent à chacune des seize valeurs de quartet possibles pour la clés à cette indiquer dans leur traversée. Le 17ème Objet est utilisé dans la Cas de cette être un terminateur nœud et Donc un clé être terminé à cette indiquer dans son traversée.

Une branche n'est alors utilisée qu'en cas de nécessité ; aucun nœud de branche ne peut exister qui ne contienne qu'une seule entrée non nulle. Nous mai officiellement définir cette structure avec la de construction composition une fonction *c* :

**** 0 ll 0ll − 1 ll ll ∃ ∈

HP ( *je* [ *je..* ( *je* 1)] *,* 1) *, je* si je = 1 où *je* : *je* je

HP ( *I* 0 [ *i..* ( *j* − 1)] *,* 0) *, n* ( je *, j* ) si *je* = *j* où *j* = max { *x* : ∃ **l** : ll **ll** ll = *x* ∧ ∀ *je* ∈ Je : *je* 0 [0 *..* ( *x* − 1)] = **l** }

****

(206) *c* ( je *, je* ) ≡ **** ( *u* (0) *, tu* (1) *, ..., tu* (15) *, v* ) sinon où *tu* ( *j* ) ≡ *n* f ( { *je* : *je* ∈ je ∧ *je* 0 [ *je* ] = *j* } *, je* + 1)

*v* =

****

*I*1 if ∃*I* : *I* ∈ I ∧ ll*I*0ll = *i*

() otherwise

D.1. **Essayer Base de données.** Ainsi non explicite hypothèses sont fabriqué concernant quelle Les données est stockée et quelle est ne pas, puisque que est un spécifique à la mise en œuvre considération; nous simplement définir la identité une fonction cartographie la valeur clé Positionner je pour un 32 octets hacher et affirmer que seul un Célibataire tel hacher existe pour quelconque je , qui mais ne pas strictement vrai est précis dans acceptable précision donné la Keccak hachis collision résistance. Dans réalité, un sensible la mise en oeuvre sera ne pas pleinement recalculer la essayer racine hacher pour chaque Positionner.

Une implémentation raisonnable maintiendra une base de données de nœuds déterminée à partir du calcul de divers essais ou, Suite officiellement, ce sera mémoire la une fonction *c* . Cette stratégie les usages la nature de la essayer pour tous les deux facilement rappeler la Contenu de

ÉTHERUM : UN SÉCURISE DÉCENTRALISÉ GÉNÉRALISÉ TRANSACTION COMPTABLE **BERLIN VERSION** 22

tout ensemble de valeurs-clés précédent et de stocker plusieurs de ces ensembles de manière très efficace. En raison de la relation de dépendance, Merkle-preuves mai être construit avec un *O* (log *N* ) espace exigence que pouvez démontrer un particulier feuille doit exister dans un essayer de un donné racine hacher.

annexe E. Précompilé Contrats

Pour chaque précompilé Contrat, nous Fabriquer utiliser de un modèle une fonction, Ξ PRE , qui met en oeuvre la panne d'essence vérification.

f≡

(207) Ξ

PRÉ

( ***σ*** *, g, UN, Je* ) ( ∅ *,* 0 *, UN,* ()) si *g < g* r

( ***σ*** *, g* − *g* r *, UN,* **o** ) autrement

Les contrats précompilés utilisent chacun ces définitions et fournissent des spécifications pour le **o** (les données de sortie) et *g* r , le gaz conditions.

On définit Ξ ECREC sous forme de contrat précompilé pour la clé publique de l'algorithme de signature numérique à courbe elliptique (ECDSA) fonction de récupération (ecrecover). Voir l'annexe [F](#_bookmark126) pour la définition de la fonction ECDSARECOVER . Nous définissons également **d** comme étant les données d'entrée, bien définies pour une longueur infinie en ajoutant des zéros si nécessaire. En cas de signature invalide ( ECDSARECOVER ( *h, v, r, s* ) = ∅ ), nous retourner non production.

(208)

(209)

(210)

Ξ ECREC ≡ Ξ PRÉ où:

*g* r = 3000

fll ll

**o** = 0 si ECDSARECOVER ( *h, v, r, s* ) = ∅

32 autrement

(211)

(212)

(213)

(214)

si tout **va** bien = 32 :

**o** [0 *..* 11] = 0

**o** [12 *..* 31] = KEC ECDSARECOVER ( *h, v, r, s* ) [12 *..* 31] où:

**d** [0 *..* ( ll [*I* **d**](#_bookmark54) ll − 1)] = *je* **d**

|  |  |  |  |
| --- | --- | --- | --- |
| (215) | **d** [ ll *je* **d** ll *..* ] | = | (0 *,* 0 *, ...* ) |
| (216) | *h* | = | **d** [0 *..* 31] |
| (217) | *v* | = | **d** [32 *..* 63] |
| (218) | *r* | = | **d** [64 *..* 95] |
| (219) | *s* | = | **d** [96 *..* 127] |

Nous définir Ξ SHA256 et Ξ RIP160 comme précompilé contrats exécution la SHA2-256 et RIPEMD-160 hacher les fonctions respectivement. Leur gaz usage est dépendant sur la saisir Les données Taille, un facteur arrondi en haut pour la la plus proche Numéro de mots.

(220) Ξ SHA256 ≡ Ξ PRÉ où:

l

(221)

*g* = 60 + 12 ll *je* **d** ll

32

r

(222) **ou** [0 *..* 31] = SHA256 ( *je* **d** )

(223) Ξ RIP160 ≡ Ξ PRÉ où:

l

(224)

*g* = 600 + 120 ll *je* **d** ll

32

r

(225) **ou** [0 *..* 11] = 0

(226) **ou** [12 *..* 31] = RIPEMD160 ( *I* **d** )

Pour la fins ici, nous présumer nous avoir bien défini la norme cryptographique les fonctions pour RIPEMD-160 et SHA2-256 de la formulaire:

(227)

(228)

SHA256 ( **je** ∈ B ) ≡ *o* ∈ B 32

RIPEMD160 ( **je** ∈ B ) ≡ *o* ∈ B 20

Le Quatrième Contrat, la identité une fonction Ξ ID simplement définit la production comme la saisir:

(229) Ξ ID ≡ Ξ PRÉ où:

l

(230)

*g* = 15 + 3 ll *je* **d** ll

32

r

1. **o** = *je* **d**

Le cinquième contrat \_ \_ effectue \_ précision arbitraire ex p on n tiation en dessous de moi dulo . Tiens , \_ 0 0 est pris \_ \_ pour être \_ une, et *X* mod \_ \_ 0 est nul pour tout *x* . Le premier mot de l'entrée spécifie le nombre d'octets occupés par le premier entier non négatif *B.* Le deuxième mot de l'entrée spécifie le nombre d'octets occupés par le deuxième entier non négatif *E.* Le troisième mot dans la saisir précise la Numéro de octets que la troisième non négatif entier *M* occupe. Celles-ci Trois mots sont

ÉTHERUM : UN SÉCURISE DÉCENTRALISÉ GÉNÉRALISÉ TRANSACTION COMPTABLE **BERLIN VERSION** 23

suivi par *B* , *E* et *M* . Le le repos de la saisir est mis au rebut. Chaque fois que la saisir est aussi court, la disparu octets sont considéré pour être zéro. Le production est codé big-endian dans la même format comme *M* 's.

1. Ξ EXPMOD ≡ Ξ PRÉ à l'exception:

*G*

r

*g* = maximum 200 *,* je *F*  max( *.€* M *, .€* B ) max( *.€*E *,* 1)

*G* quaddiviseur ≡ 3

*F* ( *x* ) ≡ *X* l 2

8

****

quaddiviseur

*.€* E  =

0 si *.€* E ≤ 32 ∧ *E* = 0

l log 2 ( *E* ) J si \_ *.€* E ≤ 32 ∧ *E*  = 0

8( *.€* E − 32) + l lo g 2 ( *je* [(96 + *.€* B ) *..* (127 + *.€* B )]) J si 32 *< . €* E ∧ *je* [(96 + *.€* B ) *..* (127 + *.€* B )] = 0

****

8( *.€* E − 32) sinon

1. **o** = *B E* mod \_ \_ *M*  ∈ N 8 *£*

M

|  |  |  |  |
| --- | --- | --- | --- |
| (238) | *.€* B | ≡ | *je* [0 *..* 31] |
| (239) | *.€* E | ≡ | *je* [32 *..* 63] |
| (240) | *.€* M | ≡ | *je* [64 *..* 95] |
| (241) | *B* | ≡ | *je* [96 *..* (95 + *.€* B )] |
| (242) | *E* | ≡ | *je* [(96 + *.€* B ) *..* (95 + *.€* B + *.€* E )] |
| (243) | *M* | ≡ | *je* [(96 + *.€* B + *.€* E ) *..* (95 + *.€* B + *.€* E + *.€* M )] |

(244)

*je* [ *x* ] *je* **ré** [ *x* ] si *X <* ll *je* **d** ll

0 sinon

f≡

* 1. **zkSNARK Lié Précompilé Contrats.** Nous choisir deux Nombres, tous les deux de qui sont prime.

(245) *p* ≡ 2188824287183927522246405745257275088696311157297823662689037894645226208583

(246) *q* ≡ 2188824287183927522246405745257275088548364400416034343698204186575808495617

Depuis *p* est un prime Numéro, { 0 *,* 1 *, . . . , p* − 1 } formes un domaine avec une addition et multiplication modulo *p* . Nous appel cette domaine *Fp* . \_

Nous définir un Positionner *C* 1 avec

(247) *C* 1 ≡ { ( *X, Oui* ) ∈ *F* p × *F* p | *Oui* 2 = *X3* \_ + 3 } ∪ { (0 *,* 0) }

Nous définir un binaire opération + sur *C* 1 pour distinct éléments ( *X* 1 *, Y* 1 ) *,* ( *X* 2 *, Oui* 2 ) avec

1

1

2

2

(0*,* 0) otherwise

(248)

( *X , Oui* ) + ( *X , Oui* ) ≡ f ( *X , Oui* ) si *X* 1 = *X2* \_

*λ Y* 2 − *Oui* 1

≡

*X2* \_ − *X* 1

*X* ≡ *λ* 2 − *X* 1 − *X2* \_

*Oui* ≡ *λ* ( *X* 1 − *X* ) − *Oui* 1

Dans la Cas où ( *X* 1 *, Oui* 1 ) = ( *X* 2 *, Y* 2 ), nous définir + sur *C* 1 avec

(249) ( *X , Oui* ) + ( *X , Oui* ) ≡ f ( *X , Oui* ) si *Oui* 1 0

1 1 2 2

*λ* ≡

(0 *,* 0) sinon

3 *X* 2

1

2 *ans* 1

*X* ≡ *λ* 2 − 2 *X* 1

*Oui* ≡ *λ* ( *X* 1 − *X* ) − *Oui* 1

( *C* 1 *,* +) est connu pour formulaire un grouper. Nous définir scalaire multiplication · avec

(250) *n* · *P* ≡ (0 *,* 0) + *P* + · · · + *P*

..,- .,

*n*

pour un Naturel Numéro *n* et un indiquer *P* dans *C1* . \_

Nous définir *P* 1 pour être un indiquer (1 *,* 2) sur *C1* . \_ Laisser *G* 1 être la sous-groupe de ( *C* 1 *,* +) généré par *P1* . \_ *G* 1 est connu pour être un cyclique grouper de Commande *Q.* \_ Pour un indiquer *P* dans *G* 1 , nous définir journal *P* 1 ( *P* ) pour être la le plus petit Naturel Numéro *n* satisfaisant *n* · *P* 1 = *P* . journal *P* 1 ( *P* ) est à plus *q* − 1.

Laisser *F p* 2 être \_ un domaine *F p* [ *je* ] */* ( *je* 2 + 1). Nous \_ définir un Positionner *C2* \_ avec

(251) *C* 2 ≡ { ( *X, Oui* ) ∈ *F p* 2 × *F p* 2 | *Oui* 2 = *X3* \_ + 3( *je* + 9) − 1 } ∪ { (0 *,* 0) }

ÉTHERUM : UN SÉCURISE DÉCENTRALISÉ GÉNÉRALISÉ TRANSACTION COMPTABLE **BERLIN VERSION** 24

Nous définir un binaire opération + et scalaire multiplication · avec la même équations [(248),](#_bookmark121) [(249)](#_bookmark122) et [(250).](#_bookmark123) ( *C2 ,* \_+) est aussi connu pour être un grouper. Nous définir *P2* \_ dans *C2* \_ avec

(252)

*P* 2 ≡ (11559732032986387107991004021392285783925812861821192530917403151452391805634 × *je*

+10857046999023057135944570762232829481370756359578518086990519993285655852781 *,*

4082367875863433681332203403145435568316851327593401208105741076214120093531 × *je*

+8495653923123431417604973247489272438418190587263600148770280649306958101930)

Nous définir *G* 2 pour être la sous-groupe de ( *C2 ,* \_+) généré par *P2* . \_ *G* 2 est connu pour être la seul cyclique grouper de Commande *q* sur *C2* . \_ Pour un indiquer *P* dans *G* 2 , nous définir journal *P* 2 ( *P* ) être la le plus petit Naturel Numéro *n* satisfaisant *n* · *P2* \_ = *P* . Avec cette définition, journal *P* 2 ( *P* ) est à plus *q* − 1.

Laisser *GT* être la multiplicatif abélien grouper sous-jacent *F q* 12 . Ce est connu que un non dégénéré bilinéaire carte *e* : *G* 1 × *G* 2 → *G T* existe. Cette application bilinéaire est un appariement de type trois. Il existe plusieurs cartes bilinéaires de ce type, ce n'est pas le cas question qui est choisi pour être *e* . Laisser *PT \_* = *e* ( *P* 1 *, P* 2 ), *un* être un Positionner de *k* points dans *G* 1 , et *b* être un Positionner de *k* points dans *G* 2 . Ce suit depuis la définition de un jumelage que la Suivant sont équivalent

(253)

(254)

log *P* 1 \_ ( *un* 1 ) × log *P* 2 \_ ( *b* 1 ) + · · · + log *P* 1 \_ ( *un k* ) × log *P* 2 \_ ( *bk* ) *\_* ≡ 1 mod \_ \_ *q*

*k*

*e* ( *un je , b je* ) = *PT \_*

*je* =0

Ainsi la jumelage opération fournit un méthode pour Vérifier [(253).](#_bookmark124)

UN 32 octet Numéro **X** ∈ **P256** \_ pourrait et pourrait ne pas représenter un élément de *Fp* . \_

(255) *δ* ( **x** ) ≡ f **x** si **X** *< p*

p

∅ autrement

UN 64 octet Les données **X** ∈ **B** 512 pourrait et pourrait ne pas représenter un élément de *G* 1 .

(256)

*δ* ( **x** ) ≡ f *g* 1 si *g* 1 ∈ *G* 1

(257)

1

∅ sinon

1. ( *x, y* ) si *X* = ∅ ∧ *y* ∅

1

∅ sinon

(258) *x* ≡ *δ* p ( **x** [0 *..* 31])

f≡

(259) *y* ≡ *δ* p ( **x** [32 *..* 63])

UN 128 octet Les données **X** ∈ **B** 1024 pourrait et pourrait ne pas représenter un élément de *G* 2 .

(260)

f≡

*δ* ( **x** ) ≡ f *g* 2 si *g* 2 ∈ *G* 2

(261)

∅ sinon

*g* (( *x* 0 *je* + *y* 0 ) *,* ( *x* 1 *je* + *y* 1 )) si *x* 0 = ∅ ∧ *y* 0 = ∅ ∧ *x1* \_

2

∅ sinon

∅ ∧ *y* 1 = ∅

2

|  |  |  |  |
| --- | --- | --- | --- |
| (262) | *x* 0 | ≡ | *δ* p ( **x** [0 *..* 31]) |
| (263) | *y* 0 | ≡ | *δ* p ( **x** [32 *..* 63]) |
| (264) | *x1* \_ | ≡ | *δ* p ( **x** [64 *..* 95]) |
| (265) | *y* 1 | ≡ | *δ* p ( **x** [96 *..* 127]) |

Nous définir Ξ SNARKV comme un précompilé Contrat qui chèques si [(253)](#_bookmark124) tient, pour prévu utiliser dans zkSNARK vérification.

ÉTHERUM : UN SÉCURISE DÉCENTRALISÉ GÉNÉRALISÉ TRANSACTION COMPTABLE **BERLIN VERSION** 25

1. Ξ SNARKV ≡ Ξ PRÉ à l'exception:
2. Ξ SNARKV ( ***σ*** *, g, UN, je* ) = ( ∅ *,* 0 *, UN,* ()) si *F*

*F* ≡ ( ll *je* **d** ll mod \_ \_ 192 = 0 ∨ ( ∃ *j. un* j = ∅ ∨ *b* j = ∅ ))

*k* = ll *je* **d** ll

192

*g* r = 34000k *\_* + 45000

f≡

**o** [0 *..* 31] 0 *x* 0000000000000000000000000000000000000000000000000000000000001 si *v* ∧ ¬F *\_*

0 *x* 00000000000000000000000000000000000000000000000000000000000000 si ¬ *v* ∧ ¬ *F*

*v* ≡ ( log *P* 1 ( *un* 1 ) × log *P* 2 \_ ( *b* 1 ) + · · · + log *P* 1 \_ ( *un k* ) × log *P* 2 \_ ( *bk* ) *\_* ≡ 1 mod \_ \_ *q* ) *une* 1  ≡ *δ* 1 ( *I* **d** [0 *..* 63])

*b* 1  ≡ *δ* 2 ( *I* **d** [64 *..* 191])

.

*un k* ≡ *δ* 1 ( *je* **ré** [( ll *je* **ré** ll − 192) *..* ( ll *je* **d** ll − 129)])

*b k* ≡ *δ* 2 ( *je* **ré** [( ll *je* **ré** ll − 128) *..* ( ll *je* **d** ll − 1)])

Nous définir un précompilé Contrat pour une addition sur *G* 1 .

1. (277)

ΞBN \_ AJOUTER ≡ ΞBN \_ PRÉ à l'exception:

ΞBN \_ AJOUTER ( ***σ*** *, g, UN, je* ) = ( ∅ *,* 0 *, UN,* ()) si *X* = ∅ ∨ *y* = ∅

*g* r = 150

**o** ≡ *δ* − 1 ( *x* + *y* ) où + est la grouper op ération \_ dans *G* 1 *x δ* 1 *Je* ¯ **d** [0 *..* 63]

≡

1

*y* ≡ *δ* 1 *je* ¯ **d** [64 *..* 127]

f*I* [*x*] if *x <* ll*I* ll**d d**¯*I* [*x*] ≡

**ré**

0 sinon

Nous \_ définir un précompilé contrat \_ \_ pour scalaire m ultiplication sur *G* 1 , où *je* ¯ **d** est défini \_ dans [(283).](#_bookmark125)

ΞBN \_ MU ≡ Ξ PRÉ à l'exception:

ΞBN \_ MUL ( ***σ*** *, g, UN, je* ) = ( ∅ *,* 0 *, UN,* ()) si *X* = ∅

*g* r = 6000

**o** ≡ *δ* − 1 ( *n* · *x* ) où · est la scalaire m ultiplication dans *G 1x* \_≡ *δ* 1 *Je* ¯ **d** [0 *..* 63]

1

(289) *n* ≡ *Je* ¯ **d** [64 *..* 95]

* 1. **BLAKE2 Précompilé Contrat.** EIP-152 par [Hess et al.](#_bookmark104) [[2016]](#_bookmark104) définit Ξ BLAKE2 F sous forme de contrat pré-compilé implémentant la fonction de compression F utilisée dans l'algorithme de hachage cryptographique BLAKE2. La compression F une fonction est spécifié dans RFC 7693 par [Saarinen et Aumasson [2015].](#_bookmark92)

(290)

(291)

(292)

(293)

Ξ BLAKE2 F ≡ Ξ PRE à l'exception:

Ξ BLAKE2 F ( ***σ*** *, g, UN, je* ) = ( ∅ *,* 0 *, UN,* ()) si ll *je* **d** ll =

*g* r = *r*

**o** ≡ LE 8 ( *h* 0 ) · *...* · LE 8 ( *h* 7 )

213 ∨ *F* ∈ */* { 0 *,* 1 }

|  |  |  |  |
| --- | --- | --- | --- |
| (294) | ( *h* 0 *, . . . , h* 7 ) | ≡ | F ( *h, moi, t* bas *, t* élevé *, F* ) avec *r* Les manches et *w* = 64 |
| (295) | BE 4 ( *r* ) | ≡ | *Je* **d** [0 *..* 4] |
| (296) | LE 8 ( *h* 0 ) | ≡ | *Je* **d** [4 *..* 12] |
| (297) |  | *. . .* |  |
| (298) | LE 8 ( *h* 7 ) | ≡ | *Je* **d** [60 *..* 68] |
| (299) | LE 8 ( *m* 0 ) | ≡ | *je* [68 **..** *76* ] |
| (300) |  | *. . .* |  |
| (301) | LE 8 ( *m* 15 ) | ≡ | *Je* **d** [188 *..* 196] |
| (302) | LE 8 ( *t* faible ) | ≡ | *Je* **d** [196 *..* 204] |
| (303) | LE 8 ( *t* élevé ) | ≡ | *Je* **d** [204 *..* 212] |
| (304) | *F* | ≡ | *je [* **212** ] |

ÉTHERUM : UN SÉCURISE DÉCENTRALISÉ GÉNÉRALISÉ TRANSACTION COMPTABLE **BERLIN VERSION** 26

où *r* ∈ B 32 , ∀ *je* ∈ 0 *..* 7 : *h je* ∈ B 64 , ∀ *je* ∈ 0 *..* 15 : *m je* ∈ B 64 , *t* faible ∈ B 64 , *t* élevé ∈ B 64 , *f* ∈ B 8 , BE *k* est le *k* -octet big-endian représentation—comparer [avec(193):](#_bookmark115)

*k* - 1

*k* 1 *n*

(305) BE *k* ( *x* ) ≡ ( *b* 0 *, b* 1 *, ..., b k* - 1 ) : *X* = *b n* · 256 − −

*n* =0

et LE *k* est la *k* -octet petit endian représentation:

*k* - 1

(306) LE *k* ( *x* ) ≡ ( *b* 0 *, b* 1 *, ..., b k* - 1 ) : *X* = *b n* · 256 *n*

*n* =0

annexe F. Signature Transactions

Les transactions sont signées à l'aide de signatures ECDSA récupérables. Cette méthode utilise la courbe SECP-256k1 comme décrit par [Courtois et Al. [2014],](#_bookmark84) et est mis en œuvre De même pour comme décrit par [Goura et Al.](#_bookmark99) [ [2004]](#_bookmark99) sur p. 9 de 15, para. 3.

Ce est assumé que la expéditeur possède un valide privé clé *p* r , qui est un au hasard choisi positif entier (représentée comme un octet déployer de longueur 32 dans big-endian formulaire) dans la Portée [1 *,* secp256k1n − 1].

Nous présumer la existence de les fonctions ECDSAPUBKEY , ECDSASIGN et ECDSARECOVER . Celles-ci sont officiellement défini dans la

Littérature, par exemple par [Johnson et Al. [2001].](#_bookmark108)

(307)

(308)

(309)

ECDSAPUBKEY ( *p* r ∈ B 32 ) ≡ *p* tu ∈ B 64

ECDSASIGN ( *e* ∈ B 32 *, p* r ∈ B 32 ) ≡ ( *v* ∈ B 1 *, r* ∈ B 32 *, s* ∈ B 32 )

ECDSARECOVER ( *e* ∈ B 32 *, v* ∈ B 1 *, r* ∈ B 32 *, s* ∈ B 32 ) ≡ *p* tu ∈ B 64

Où *tu* es est la clé publique, supposée être un tableau d'octets de taille 64 (formé de la concaténation de deux entiers chaque *<* 2 256 ), *p* r est la privé clé, un octet déployer de Taille 32 (ou alors un Célibataire positif entier dans la susmentionné Portée) et *e* est la hacher de la transaction, [*h* ( *T*](#_bookmark130) ). Ce est assumé que *v* est la 'récupération identifiant'. Le récupération identifiant

est une valeur de 1 octet spécifiant la parité et la finitude des coordonnées du point de courbe pour lequel *r* est la valeur x ; cette la valeur est dans la plage de [0 *,* 3], cependant nous déclarons les deux possibilités supérieures, représentant des valeurs infinies, invalides. Le valeur 0 représente un même *y* valeur et 1 représente un étrange *y* valeur.

Nous déclarer que un ECDSA Signature est invalide sauf si tout la Suivant conditions sont vrai:

(310)

(311)

(312)

où: (313)

0 *<**r <* secp256k1n

0 *< s <* secp256k1n ÷ 2 + 1

[*v*](#_bookmark127) ∈ { 0 *,* 1 }

secp256k1n = 115792089237316195423570985008687907852837564279074904382605163141518161494337

Pour un donné privé clé, *p* r , la Ethereum adresse *A* ( *p* r ) (un 160 bits valeur) pour qui ce correspond est défini comme la le plus à droite 160 bits de la Keccak-256 hacher de la correspondant ECDSA Publique clé:

(314) *A* ( *p* r ) = B 96 *..* 255 KEC ECDSAPUBKEY ( *p* r )

Le message hacher, *h* ( *T* ), pour être signé est la Keccak-256 hacher de la transaction. Trois différent les saveurs de signature régimes sont disponible:



(315)

*LX* ( *T* \_) ≡

où

( *T* n *, T* p *, T* g *, T* t *, Télé ,* \_**p** ) si *T* x = 0 ∧ *T* w ∈ { 27 *,* 28 }

( *T* n *, T* p *, T* g *, T* t *, Télé ,* \_**p** *, β,* () *,* ()) si *T* x = 0 ∧ *T* w ∈ { 2 *β* + 35 *,* 2 *β* + 36 }



 ( *T* c *, T* n *, T* p *, T* g *, T* t *, Télé ,* \_**p** *, T* **A** ) si *T* x = 1

f≡

(316)

**p** *T* **je** si *T* t = ∅

*T* **d** autrement

f≡

*h* ( *T* ) KEC ( RLP ( *L* X ( *T* ))) si *T* x = 0

KEC ( *T* x · RLP ( *L* X ( *T* ))) autrement

Le signé transaction *G* ( *T, p* r ) est défini comme: (317)

*G* ( *T, p* r ) ≡ *T* sauf :

(318)

[Réitérant depuis précédemment:](#_bookmark8) (319)

(320)

( *T* y *, T* r *, Ts* ) \_ = ECDSASIGN ( *h* ( *T* ) *, p* r )

*T* r = [*r*](#_bookmark128) *Ts* \_ = [*s*](#_bookmark129)

et [*T* w](#_bookmark11) de héritage transactions est Soit 27 + *T* y ou alors 2 [*β*](#_bookmark1) + 35 + *T* y .

ÉTHERUM : UN SÉCURISE DÉCENTRALISÉ GÉNÉRALISÉ TRANSACTION COMPTABLE **BERLIN VERSION** 27

|  |  |
| --- | --- |
| Nous mai alors définir la expéditeur une fonction *S* de la transaction comme: | |
| (321) | *S* ( *T* ) ≡ B 96 *..* 255 KEC ECDSARECOVER ( *h* ( *T* ) *, v, T* r *, Ts* ) \_ |
| (322) |    *T* w − 27 si *T* x = 0 ∧ *T* w ∈ { 27 *,* 28 }  *v* ≡ ( *T* w − 35) mode 2 si *T* x = 0 ∧ *T* w ∈ { 2 *β* + 35 *,* 2 *β* + 36 }    *T* y si *T* x = 1 |
| Le affirmation que la expéditeur de un signé transaction équivaut à la adresse de la signataire devrait être va de soi :  (323 *)* ∀T: ∀ *p* r : *S* ( *G* ( *T , p* r )) ≡ *A* ( *p* r )  annexe G. Frais Programme  Le frais programme *g* est un tuple de scalaire valeurs correspondant pour la relatif frais, dans gaz, de un Numéro de abstrait opérations que un transaction mai effet. | |

|  |  |  |
| --- | --- | --- |
| Nom | Valeur | La description |
| *G* zéro | 0 | Rien payé pour opérations de la Positionner *W* zéro . |
| *G* jumpdest | 1 | Montant de gaz pour Payer pour un JUMPDEST opération. |
| Base *G* | 2 | Montant \_ \_ de gaz pour payer \_ \_ pour op érations \_ de la ensemble \_ \_ *W* base e . |
| *Sol* très bas | 3 | Montant \_ \_ de gaz pour payer \_ \_ pour op érations \_ de la ensemble \_ \_ *W* très bas . \_ \_ |
| *sol* bas | 5 | Montant de gaz pour Payer pour opérations de la Positionner *W* faible . |
| *sol* moyen | 8 | Montant \_ \_ de gaz pour payer \_ \_ pour op érations \_ de la ensemble \_ \_ *W* mi d . |
| *Sol* aigu | dix | Montant \_ \_ de gaz pour payer \_ \_ pour op érations \_ de la Positionner *W* haut h . |
| *G* warmaccess | 100 | Coût de un chauffer Compte ou alors stockage accéder. |
| Adresse de la liste d'accès *G* | 2400 | Coût de échauffement en haut un Compte avec la accéder liste. |
| Stockage des listes d'accès *G* | 1900 | Coût de échauffement en haut un stockage avec la accéder liste. |
| *G* coldaccountacces | 2600 | Coût de un du froid Compte accéder. |
| *G* charge froide | 2100 | Coût de un du froid stockage accéder. |
| *G* sset | 20000 | Payé pour un STORE opération lorsque la stockage valeur est Positionner pour non nul depuis zéro. |
| *G* sréinitialiser | 2900 | Payé pour un STORE opération lorsque la stockage valeurs le zéro restes inchangé ou alors est Positionner pour zéro. |
| *R* clair | 15000 | Rembourser donné (ajoutée dans rembourser compteur) lorsque la stockage valeur est Positionner pour zéro depuis  non nul. |
| *R* autodestruction | 24000 | Rembourser donné (ajoutée dans rembourser compteur) pour autodestructeur un Compte. |
| *G* autodestruction | 5000 | Montant de gaz pour Payer pour un AUTO-DESTRUCTION opération. |
| *G* créer | 32000 | Payé pour un CRÉER opération. |
| Dépôt de code *G* | 200 | Payé par octet pour un CRÉER opération pour réussir dans placement code dans Etat. |
| Valeur d'appel *G* | 9000 | Payé pour un non nul valeur transférer comme partie de la APPEL opération. |
| Allocation d'appel *G* | 2300 | UN traitement pour la appelé Contrat soustrait depuis Valeur d'appel *G* pour un non nul valeur transférer. |
| *G* nouveau compte | 25000 | Payé pour un APPEL ou alors AUTO-DESTRUCTION opération qui crée un Compte. |
| *G* exp | dix | Partiel Paiement pour un EXP opération. |
| *G* expbyte | 50 | Partiel Paiement lorsque multiplié par la Numéro de octets dans la exposant pour la EXP opération. |
| Mémoire *G* | 3 | Payé pour tous Additionnel mot lorsque expansion Mémoire. |
| *G* txcreate | 32000 | Payé par tout création de contrat transactions après la *Propriété* passage. |
| *G* txdatazero | 4 | Payé pour tous zéro octet de Les données ou alors code pour un transaction. |
| *G* txdatanonzero | 16 | Payé pour tous non nul octet de Les données ou alors code pour un transaction. |
| transaction*G* | 21000 | Payé pour tous transaction. |
| Journal *G* | 375 | Partiel Paiement pour un JOURNAL opération. |
| Données de journal *G* | 8 | Payé pour chaque octet dans un JOURNAL l'opération Les données. |
| Sujet de journal *G* | 375 | Payé pour chaque sujet de un JOURNAL opération. |
| *G* keccak256 | 30 | Payé pour chaque KECCAK256 opération. |
| *G* keccak256word | 6 | Payé pour chaque mot (arrondi en haut) pour saisir Les données pour un KECCAK256 opération. |
| Copie *G* | 3 | Partiel Paiement pour \*COPIE opérations, multiplié par mots copié, arrondi en haut. |
| *G* bloc de hachage | 20 | Paiement pour chaque BLOCHASH opération. |

annexe H Virtuel Machine spécification

Lorsque interprétant 256 bits binaire valeurs comme entiers, la représentation est big-endian.

Lorsqu'une donnée machine 256 bits est convertie vers et à partir d'une adresse ou d'un hachage 160 bits, la droite (ordre inférieur pour BE) 20 octets sont utilisés et les 12 les plus à gauche sont ignorés ou remplis de zéros, ainsi les valeurs entières (lorsque les octets sont interprété comme big-endian) sont équivalent.

ÉTHERUM : UN SÉCURISE DÉCENTRALISÉ GÉNÉRALISÉ TRANSACTION COMPTABLE **BERLIN VERSION** 28

* 1. **Gaz Coût.** Le général gaz Coût une fonction, *C* , est défini comme:

(324)

*C* ( ***σ*** *,* ***µ*** *, UN, je* ) ≡ *C* mém ( ***µ*** i ) − *C* mém ( ***µ*** i ) +

****

*C* SSTORE ( ***σ*** *,* ***µ*** *, UN, je* ) si *w* = STORE

*G* exp si *w* = EXP ∧ ***µs [* 1** ] = 0

****

****

*G* exp + *G* expoctet × (1 + l log 256 ( ***µ* s** [1]) J ) si *w* = EXP ∧ ***µs [* 1** ] *>* 0 *Sol* très bas + Copie *G* × r ***µs* [** 2] ÷ 32 1 si *w* ∈ *W* copie *C* aaccès ( ***µ* s** [0] mode 2 160 *, A* ) + Copie *G* × r ***µs* [** 3] ÷ 32 1 si *w* = EXTCODECOPIE *C* aaccès ( ***µ* s** [0] mode 2 160 *, A* ) si *w* ∈ *W* excompte

****

****

****

Journal *G* + Données de journal *G* × ***µs [* 1** ] si *w* = LOG0

****

Journal *G* + Données de journal *G* × ***µs [* 1** ] + *G* logtopic si *w* = LOG1

Journal *G* + Données de journal *G* × ***µs [* 1** ] + 2 *G* logtopic si *w* = LOG2

****

Journal *G* + Données de journal *G* × ***µs [* 1** ] + 3 *G* logtopic si *w* = LOG3

Journal *G* + Données de journal *G* × ***µs [* 1** ] + 4 *G* logtopic si *w* = LOG4

****

*C* APPEL ( ***σ*** *,* ***µ*** *, A* ) si *w* ∈ *W* appel

*C* AUTODESTRUCTION ( ***σ*** *,* ***µ*** ) si *w* = AUTO-DESTRUCTION

*G* créer si *w* = CRÉER

****

*G* créer + *G* keccak256word × r ***µs* [** 2] ÷ 32 1 si *w* = [CRÉER2](#_bookmark150) *G* keccak256 + *G* keccak256word × r ***µs [* 1** ] ÷ 32 1 si *w* = KECCAK256 *G* jumpdest si *w* = JUMPDEST

****

*C* SLOAD ( ***µ*** *, A, I* ) si *w* = SLOAD

****

*G* zéro si *w* ∈ *W* zéro

Base *G* si *w* ∈ socle en *W*

****

*G* très faible si *w* ∈ *W* très bas

*G* bas si *w* ∈ *W* faible

****

*G* mi si *w* ∈ *W* mi

*G* élevé si *w* ∈ *W* haut

 *G* blockhash si *w* = [BLOCHASH](#_bookmark144)

f≡

1. *w je* **b** [ ***µ*** pc ] si ***µpc*** \_ *<* ll *je* **b** ll

ARRÊTER sinon

où:

1. *C* mém ( *un* ) ≡ Mémoire *G* · *un* +

*un* 2

512

1. *C*

un accès

( *x, A* ) *G* warmaccess si *x* ∈ [*Un* **un**](#_bookmark39)*G* coldaccountacces autrement

avec *C* CALL , *C* SELFDESTRUCT , *C* SLOAD et *C* SSTORE comme spécifié dans la section appropriée ci-dessous. Nous définissons ce qui suit sous-ensembles de des instructions:

f≡

*W* zéro = { ARRÊTER , RETOUR , RETOUR }

socle en *W* = { ADRESSE , ORIGINE , APPELANT , CALLVALUE , CALLDATASIZE , CODESIZE , GASPRICE , COINBASE , Horodatage , NOMBRE , DIFFICULTE , LIMITE DE GAZ , CHAINID , RETURNDATASIZE , POP , PC , TAILLE , GAZ }

*W* très bas = { AJOUTER , SOUS , PAS , LT , GT , SLT , SGT , égaliseur , ISZÉRO , ET , OU , XOR , OCTET , SHL , SHR , RAS , CALLDATALOAD , CHARGEMENT , MSTORE , MSTORE8 , POUSSER\* , DUP\* , ÉCHANGE\* }

*W* bas = { MUL , DIV , SDIV , MOD , SMOD , SIGNEXTEND , ÉQUILIBRE AUTONOME }

*W* mi = { ADDMOD , MULMOD , SAUT }

*W* élevé = { JUMPI }

*W* copie = { CALLDATACOPY , CODECOPIE , RENVOYER UNE COPIE } *W* appel = { APPEL , CODE APPEL , DELEGATECALL , STATICCALL } *W* excompte = { ÉQUILIBRE , EXTCODETAILLE , EXTCODEHACHE }

Notez le composant de coût de la mémoire, donné comme le produit de la mémoire *G* et du maximum de 0 et du plafond du nombre nombre de mots en taille que la mémoire doit dépasser du nombre de mots courant, ***µ*** i pour que tous les accès référencent valide Mémoire qu'il s'agisse pour lis ou alors écrivez. Tel accède doit être pour non nul Numéro de octets.

Référencement un zéro longueur Portée (par exemple par tenter pour passer ce comme la saisir Portée pour un APPEL) Est-ce que ne pas exiger Mémoire pour être élargi pour la début de la Portée. ***µ*** je est défini comme cette Nouveau maximum Numéro de mots de actif Mémoire; cas spéciaux sont donné où celles-ci deux sont ne pas égal.

ÉTHERUM : UN SÉCURISE DÉCENTRALISÉ GÉNÉRALISÉ TRANSACTION COMPTABLE **BERLIN VERSION** 29

Notez également que *C* mem est la fonction de coût mémoire (la fonction d'expansion étant la différence entre le coût avant et après). C'est un polynôme, avec le coefficient d'ordre supérieur divisé et plancher, et donc linéaire jusqu'à 724B de mémoire utilisé, après qui ce frais substantiellement Suite.

Tandis que définir la instruction Positionner, nous défini la extension de mémoire pour Portée une fonction, *M* , Donc:

1. *M* ( *s, F, l* ) *s* si *je* = 0 max( *s,* r ( *f* + *l* ) ÷ 32 1 ) autrement

f≡

Une autre utile une fonction est "tout mais une 64e » une fonction *L* défini comme:

(329) *L* ( *n* ) ≡ *n* − l *n/* 64 J

* 1. **Jeu d'instructions.** Comme précisé précédemment dans la section [9,](#_bookmark52) ces définitions prennent place dans le contexte final ici. Dans particulier nous présumer *O* est la EVM progression d'état une fonction et définir la termes se rapportant pour la suivant cycle Etat ( ***σ*** *,* ***µ*** ) tel que:

(330) *O* ( ***σ*** *,* ***µ*** *, UN, je* ) ≡ ( ***σ*** *,* ***µ*** *, UN , je* ) avec exceptions, comme c'est noté

Voici les différentes exceptions aux règles de transition d'état données dans la section [9](#_bookmark52) spécifiées pour chaque instruction, ainsi que les définitions supplémentaires spécifiques aux instructions de *J* et *C.* \_ Pour chaque instruction, est également spécifié *α* , le Additionnel éléments mis sur la pile et *δ* , la éléments supprimé depuis pile, comme défini dans section [9.](#_bookmark52)

ÉTHERUM : UN SÉCURISE DÉCENTRALISÉ GÉNÉRALISÉ TRANSACTION COMPTABLE **BERLIN VERSION** 30

# 0 : Arrêt et Arithmétique Opérations

Tout arithm e tic est moi dulo \_ 2 256 sauf si sinon \_ c'est noté. Le zéro-ième puissance \_ \_ \_ de zéro 0 0 est défini \_ pour être \_ une.

# Valeur Mnémonique *δ α* La description

0x00 ARRÊTER 0 0 Arrêts exécution.

0x01 AJOUTER 2 1 Une addition opération.

***µs [* 0** ] ≡ ***µs [* 0** ] + ***µs [* 1** ]

0x02 MU 2 1 Multiplication opération.

***µs [* 0** ] ≡ ***µs [* 0** ] × ***µs [* 1** ]

0x03 SOUS 2 1 Soustraction opération.

***µs [* 0** ] ≡ ***µs [* 0** ] − ***µs [* 1** ]

0x04 DIV 2 1 Entier division opération.

0 si ***µs [* 1** ] = 0

f***µ*** [0] ≡

**s**

lµs ***[* 0** ] ÷ ***µs [* 1** ] J autrement



0x05 SDIV 2 1 Signé entier division opération (tronqué).

255 255

  **sgn** ( ***µ* s** [0] ÷ ***µs*** [1]) **l** | ***µs [* 0** ] ÷ ***µs [* 1** ] |J autrement

Où tout valeurs sont traité comme deux complément signé 256 bits entiers. Noter la débordement \_ \_ \_ \_ séma ntique \_ lorsque − 2 255 est nier d.

**s**

***µ*s**[0] ≡

−2 if ***µ*** [0] = −2 ∧ ***µ*** [1] = −1

0 if ***µ*s**[1] = 0

**s**

0x06 MOD 2 1 Modulo reste opération.

***µ*** [0] ≡ 0 si ***µs [* 1** ] = 0

**s**

**s**

f

**s**

***µ*** [0] mod ***µ*** [1] otherwise

0x07 SMOD 2 1 Signé modulo reste opération.

f

***µ*** [0] ≡ 0 si ***µs [* 1** ] = 0

**s**

**sg** ( ***µ*** [0])( | ***µ*** [0] | mode | ***µ*** [1] | ) autrement

**s**

**s**

**s**

Où tout valeurs sont traité comme deux complément signé 256 bits entiers.

0x08 ADDMOD 3 1 Modulo une addition opération.

***µ*** [0] ≡ 0 si ***µs* [** 2] = 0

**s**

**s**

**s**

f

**s**

(***µ*** [0] + ***µ*** [1]) mod ***µ*** [2] otherwise

Tout intermédiaire calculs de cette opération sont ne pas matière pour la 2 256 modulo.

0x09 MULMOD 3 1 Modulo multiplication opération.

***µ*** [0] ≡ 0 si ***µs* [** 2] = 0

f

**s**

( ***µ*** [0] × ***µ*** [1]) mode ***µ*** [2] autrement

**s**

**s**

**s**

Tout intermédiaire calculs de cette opération sont ne pas matière pour la 2 256 modulo.

0x0a EXP 2 1 Exponentiel opération.

***µ*** [0] ≡ ***µ*** [0] ***µs [* 1** ]

**s**

**s**

0x0b SIGNEXTEND 2 1 Étendre longueur de deux complément signé entier.

∀ *je* ∈ [0 *..* 255] : ***µ*** [0]

**s**

≡ fµs ***[* 1** ] t si *je*  *t* où *t* = 256 − 8( ***µs [* 0** ] + 1)

***µ* s** [ *x* ] je donne la *je* th bit (compte depuis zéro) de ***µs*** [ *x* ] **\_**

i

***µ*s**[1]

i

otherwise

ÉTHERUM : UN SÉCURISE DÉCENTRALISÉ GÉNÉRALISÉ TRANSACTION COMPTABLE **BERLIN VERSION** 31

# 10s : Comparaison & Au niveau du bit Logique Opérations

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Value** | **Mnemonic** | *δ* | *α* | **Description** |
| 0x10 | LT | 2 | 1 | Less-than comparison. |

***µ*** [0] ≡ f 1 si ***µs [* 0** ] *<* ***µs [* 1** ]

**s**

0 otherwise

0x11 GT 2 1 Plus grand que comparaison.

f

***µ*** [0] ≡ 1 si ***µs [* 0** ] *>* ***µs [* 1** ]

**s**

0 otherwise

0x12 PS 2 1 Signé moins que comparaison.

f

***µ*** [0] ≡ 1 si ***µs [* 0** ] *<* ***µs [* 1** ]

**s**

0 otherwise

Où tout valeurs sont traité comme deux complément signé 256 bits entiers.

0x13 SGT 2 1 Signé plus grand que comparaison.

***µ*** [0] ≡ 1 si ***µs [* 0** ] *>* ***µs [* 1** ]

f

**s**

0 otherwise

Où tout valeurs sont traité comme deux complément signé 256 bits entiers.

0x14 EQ 2 1 Égalité comparaison.

***µ*** [0] ≡ 1 si ***µs [* 0** ] = ***µs [* 1** ]

f

**s**

0 otherwise

0x15 ISZÉRO 1 1 Simple ne pas opérateur.

***µ*** [0] ≡ 1 si ***µs [* 0** ] = 0

f

**s**

0 otherwise

0x16 ET 2 1 Au niveau du bit ET opération.

∀ *je* ∈ [0 *..* 255] : ***µs [* 0** ] je ≡ ***µs [* 0** ] je ∧ ***µs [* 1** ] je

0x17 OU 2 1 Au niveau du bit OU ALORS opération.

∀ *je* ∈ [0 *..* 255] : ***µs [* 0** ] je ≡ ***µs [* 0** ] je ∨ ***µs [* 1** ] je

0x18 OU exclusif 2 1 Au niveau du bit XOR opération.

∀ *je* ∈ [0 *..* 255] : ***µs [* 0** ] je ≡ ***µs [* 0** ] je ⊕ ***µs [* 1** ] je

f≡

0x19 PAS 1 1 Au niveau du bit NE PAS opération.

∀ *je* ∈ [0 *..* 255] : ***µs [* 0** ] je

1 si ***µs [* 0** ] je = 0

0 autrement

0x1a OCTET 2 1 Récupérer Célibataire octet depuis mot.

f≡

∀ *je* ∈ [0 *..* 255] : ***µs [* 0** ] je

***µ* s** [1] ( *je* − 248+8 ***µ* s** [0]) si *je* ≥ 248 ∧ ***µs [* 0** ] *<* 32

0 sinon

Pour la Nième octet, nous compter depuis la la gauche (c'est à dire N=0 voudrais être la plus important dans gros endien).

|  |  |  |  |
| --- | --- | --- | --- |
| 0x1b | SHL | 2 | 1 La gauche changement opération.  ***µ*** [0] ≡ ( ***µ*** [1] × 2 ***µs [* 0** ] ) mode 2 256  **s s** |
| 0x1c | SHR | 2 | 1 Logique à droite changement opération.  ***µ*** [0] ≡ l ***µ*** [1] ÷ 2 ***µs [* 0** ] J  **s s** |
| 0x1d | DAS | 2 | 1 Arithmétique (signé) à droite changement opération. |

***µ*** [0] ≡ ***lµ*** [1] ÷ 2 ***µs [* 0** ] J

**s**

**s**

Où ***µs [* 0** ] et ***µs [* 1** ] sont traité comme deux complément signé 256 bits entiers, pendant que ***µs [* 0** ] est traité comme non signé.

# 20 s : KECCAK256

|  |  |  |  |
| --- | --- | --- | --- |
| **Valeur** | **Mnémonique** *δ* | *α* | **La description** |
| 0x20 | KECCAK256 2 | 1 | Calculer Keccak-256 hacher.  ***µs [* 0** ] ≡ KEC ( ***µ* m** [ ***µ* s** [0] *. . .* ( ***µs [* 0** ] + ***µs [* 1** ] − 1)])  ***µ*** je ≡ *M* ( ***µ*** je *,* ***µs [* 0** ] *,* ***µs [* 1** ]) |

ÉTHERUM : UN SÉCURISE DÉCENTRALISÉ GÉNÉRALISÉ TRANSACTION COMPTABLE **BERLIN VERSION** 32

∀ *je* ∈ { 0 *. . .* ***µ*** [2] − 1 } : ***µ***

**s**

[ ***µ*** [0] + *je* ] ≡ f *je* **d** [ ***µs [* 1** ] + *je* ] si ***µs [* 1** ] + *je <* ll *je* **d** ll

Le ajouts dans ***µ*** [1] + *je* sont ne pas matière pour la 2 256 modulo.

**s**

**m**

**s**

0 otherwise

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | **30s: Environmental Information** |
| **Value** | **Mnemonic** | *δ* | *α* | **Description** |
| 0x30 | ADDRESS | 0 | 1 | Get address of currently executing account. |
| 0x31 | BALANCE | 1 | 1 | ***µ*s**[0] ≡ *I*a  Get balance of the given account. |
|  |  |  |  | f***σ***[***µ*** [0] mod 2160]b if ***σ***[***µ*** [0] mod 2160] = ∅  ***µ*** [0] ≡ **s s**  **s** 0 otherwise  *A***a** ≡ *A***a** ∪ {***µ*s**[0] mod 2 }  160 |
| 0x32 | ORIGIN | 0 | 1 | Get execution origination address.  ***µ*s**[0] ≡ *I*o  This is the sender of original transaction; it is never an account with  non-empty associated code. |
| 0x33 | CALLER | 0 | 1 | Get caller address.  ***µ*s**[0] ≡ *I*s  This is the address of the account that is directly responsible for  this execution. |
| 0x34 | CALLVALUE | 0 | 1 | Get deposited value by the instruction/transaction responsible for this execution.  ***µ*s**[0] ≡ *I*v |
| 0x35 | CALLDATALOAD | 1 | 1 | Get input data of current environment.  ***µ*s**[0] ≡ *I***d**[***µ*s**[0] *. . .* (***µ*s**[0] + 31)] with *I***d**[*x*] = 0 if *x* ;;: ll*I***d**ll This pertains to the input data passed with the message call instruction or transaction. |
| 0x36 | CALLDATASIZE | 0 | 1 | Get size of input data in current environment.  ***µ*s**[0] ≡ ll*I***d**ll  This pertains to the input data passed with the message call  instruction or transaction. |
| 0x37 | CALLDATACOPY | 3 | 0 | Copy input data in current environment to memory. |

***µ*** je ≡ *M* ( ***µ*** je *,* ***µs [* 0** ] *,* ***µs* [** 2])

Cette se rapporte pour la saisir Les données passé avec la message appel instruction

ou alors transaction.

0x38 CODETAILLE 0 1 Avoir Taille de code course dans courant environnement.

***µs [* 0** ] ≡ ll *je* **b** ll

0x39 CODECOPIE 3 0 Copie code course dans courant environnement pour Mémoire.

∀ *je* ∈ { 0 *. . .* ***µ*** [2] − 1 } : ***µ***

[ ***µ*** [0] + *je* ] ≡ f *je* **b** [ ***µs [* 1** ] + *je* ] si ***µs [* 1** ] + *je <* ll *je* **b** ll

**s m s**

***µ*** je ≡ *M* ( ***µ*** je *,* ***µs [* 0** ] *,* ***µs* [** 2])

ARRÊTER sinon

Le ajouts dans ***µ*** [1] + *je* sont ne pas matière pour la 2 256 modulo.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | **s** |
| 0x3a | GASPRICE | 0 | 1 | Get price of gas in current environment.  ***µ*s**[0] ≡ *I*p  This is gas price specified by the originating transaction. |
| 0x3b | EXTCODESIZE | 1 | 1 | Get size of an account’s code. |

***µs [* 0** ] ≡

160

**s**

fll**b**ll if ***σ***[***µ*** [0] mod 2 ] = ∅

0 sinon

où CEC ( **b** ) ≡ ***σ*** [ ***µ*** [0] mod \_ \_ 2 16 0 ] c

**s**

*Un* **un** ≡ *Un* **un** ∪ { ***µ*** [0] mode 2 160 }

**s**

0x3c EXTCODECOPIE 4 0 Copie un comptes code pour Mémoire.

∀ *je* ∈ { 0 *. . .* ***µ*** [3] − 1 } : ***µ***

**s**

[ ***µ*** [1] + *je* ] ≡ f **b** [ ***µs* [** 2] + *je* ] si ***µs* [** 2] + *je <* ll **b** ll

où CEC ( **b** ) ≡ ***σ*** [ ***µ*** [0] mod \_ \_ 2 16 0 ] c

**s**

**m**

**s**

STOP otherwise

Nous \_ présumer **b** ≡ () si ***σ*** [ ***µ*** [0] mod \_ \_ 2 16 0 ] = ∅ .

**s**

***µ*** je *M* ( ***µ*** je *,* ***µs [* 1** ] *,* ***µs* [** 3])

≡

Le ajouts dans ***µ*** [2] + *je* sont ne pas matière pour la 2 256 modulo.

**s**

*Un* **un** ≡ *Un* **un** ∪ { ***µ*** [0] mode 2 160 }

**s**

ÉTHERUM : UN SÉCURISE DÉCENTRALISÉ GÉNÉRALISÉ TRANSACTION COMPTABLE **BERLIN VERSION** 33

0x3d RETURNDATASIZE 0 1 Avoir Taille de production Les données depuis la précédent appel depuis la courant

environnement.

***µs [* 0** ] ≡ ll ***µ* o** ll

0x3e RETOURDATACOPIE 3 0 Copie production Les données depuis la précédent appel pour Mémoire.

∀ *je* ∈ { 0 *. . .* ***µ*** [2] − 1 } : ***µ***

**s**

[ ***µ*** [0] + *je* ] ≡ f ***µ* o** [ ***µ* s** [1] + *je* ] si ***µs [* 1** ] + *je <* ll ***µ* o** ll

Le ajouts dans ***µ*** [1] + *je* sont ne pas matière pour la 2 256 modulo.

**s**

**m**

**s**

0 otherwise

***µ*** je ≡ *M* ( ***µ*** je *,* ***µs [* 0** ] *,* ***µs* [** 2])

0x3f EXTCODEHAS 1 1 Avoir hacher de un comptes code.

0 si MORT ( ***σ*** *,* ***µ*** [0] mode 2 160 )

***σ***[***µ*** [0] mod 2160]c otherwise

f

***µ*s**[0] ≡

**s**

*Un* **un** ≡ *Un* **un** ∪ { ***µ*** [0] mode 2 160 }

# 40 ans : Bloc Information

**s**

**s**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Value** | **Mnemonic** | *δ* | *α* | **Description** |
| 0x40 | BLOCKHASH | 1 | 1 | Get the hash of one of the 256 most recent complete blocks.  ***µ*s**[0] ≡ *P* (*I*Hp *,* ***µ*s**[0]*,* 0) |

où *P* est la hacher de un bloc de un particulier Numéro, en haut pour un maximum âge. 0 est la gauche sur la pile si la regardé pour bloc Numéro est plus grand que ou alors

égal pour la courant bloc Numéro ou alors Suite que 256 blocs derrière la courant bloc.



*P* ( *h, n, un* ) ≡

1. si *n* = *Salut* \_

  *P* ( *H* p *, n, un* + 1) autrement

0 if *n > H*i ∨ *a* = 256 ∨ *h* = 0

et nous affirmer la entête *H* pouvez être déterminé depuis son hacher *h* sauf si *h* est zéro (comme est la Cas pour la parent hacher de la genèse bloc).

0x41 COINBASE 0 1 Avoir la courant bloc de Bénéficiaire adresse.

***µs [* 0** ] ≡ *je* hc

0x42 Horodatage 0 1 Avoir la courant bloc de horodatage.

***µs [* 0** ] ≡ *Je* Hs

0x43 NUMÉRO 0 1 Avoir la courant bloc de Numéro.

***µs [* 0** ] ≡ *je* salut

0x44 DIFFICULTÉ 0 1 Avoir la courant bloc de difficulté.

***µs [* 0** ] ≡ *je me* suis

0x45 LIMITE DE GAZ 0 1 Avoir la courant bloc de gaz limite.

***µs [* 0** ] ≡ *je* hl

0x46 CHAINID 0 1 Avoir la [chaîne IDENTIFIANT.](#_bookmark1)

***µs [* 0** ] ≡ *β*

0x47 AUTO-ÉQUILIBRE 0 1 Avoir solde de actuellement exécution Compte.

***µ*****s** [0] ≡ ***σ*** [ *je* une ] b

ÉTHERUM : UN SÉCURISE DÉCENTRALISÉ GÉNÉRALISÉ TRANSACTION COMPTABLE **BERLIN VERSION** 34

# 50 ans : Pile, Mémoire, Stockage et Flux Opérations

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Valeur**  0x50 | **Mnémonique**  POPULAIRE | *δ*  1 | *α*  0 | **La description**  Éliminer Objet depuis pile. |
| 0x51 | MLOAD | 1 | 1 | Charge mot depuis Mémoire.  ***µs [* 0** ] ≡ ***µ* m** [ ***µ* s** [0] *. . .* ( ***µs [* 0** ] + 31)]  ***µ*** je ≡ max( ***µ*** je *,* r ( ***µs [* 0** ] + 32) ÷ 32 1 )  Le une addition dans la calcul de ***µ*** est ne pas matière pour la 2 256 modulo.  je |
| 0x52 | MSTORE | 2 | 0 | Sauver mot pour Mémoire.  ***µ* m** [ ***µ* s** [0] *. . .* ( ***µs [* 0** ] + 31)] ≡ ***µs [* 1** ] ***µ*** je ≡ max( ***µ*** je *,* r ( ***µs [* 0** ] + 32) ÷ 32 1 )  Le une addition dans la calcul de ***µ*** est ne pas matière pour la 2 256 modulo.  je |
| 0x53 | MSTORE8 | 2 | 0 | Sauver octet pour Mémoire.  ***µ* m** [ ***µ* s** [0]] ≡ ( ***µs [* 1** ] mode 256)  ***µ*** je ≡ max( ***µ*** je *,* r ( ***µs [* 0** ] + 1) ÷ 32 1 )  Le une addition dans la calcul de ***µ*** est ne pas matière pour la 2 256 modulo.  je |
| 0x54 | SLOAD | 1 | 1 | Charge mot depuis stockage. |

***µs [* 0** ] ≡ ***σ*** [ *je* une ] **s** [ ***µ* s** [0]]

*UN***K** ≡ **AK** *\_*∪ { ( *je* un *,* ***µ*** f **s** [0]) }

*C*

SLOAD

(***µ****, A, I*) ≡

*G*warmaccess if (*I*a*,* ***µ*s**[0]) ∈ *A***K**

*G*coldsload otherwise

0x55 STORE 2 0 Sauver mot pour stockage.

***σ***[ *je* une ] **s** [ ***µ* s** [0]] ≡ ***µs [* 1** ]

**AK** *\_*≡ **AK** *\_*∪ { ( *je* un *,* ***µs [* 0** ]) }

*C* SSTORE ( ***σ*** *,* ***µ*** ) et *Un* r sont spécifié par EIP-2200 comme suit.

Nous rappeler la lecteur que la point de contrôle ("original") Etat [***σ*** 0](#_bookmark41) est la Etat si la courant transaction ont été pour revenir.

Laisser *v* 0 = ***σ*** 0 [ *je* une ] **s** [ ***µ* s** [0]] être la original valeur de la stockage insérer. Laisser *v* = ***σ*** [ *je* une ] **s** [ ***µ* s** [0]] être la courant valeur.

Laisser *v* = ***µs [* 1** ] être la Nouveau valeur.

Puis:

f≡

*C* SSTORE

( ***σ*** *,* ***µ*** *, UN, je* ) 0 si ( *je* une *,* ***µ* s** [0]) ∈ *A* **K** *G* charge froide autrement

   *G* w armac c ess si *v* = *v* ∨ *v* 0 = *v*

+ *G* sset si *v* = *v* ∧ *v* 0 = *v* ∧ *v* 0 = 0



  *G* sreset si *v* = *v* ∧ *v* 0 = *v* ∧ *v* 0 = 0

  *R* clair  si *v*  = *v* ∧ *v* 0 = *v* ∧ *v* = 0

*A*r ≡ *A*r +

*r*dirtyclear + *r*dirtyreset if *v*

*v* ∧ *v*0 = *v*

où

  0 sinon

   − *R* clair si *v* 0 = 0 ∧ *v* = 0

*r* saleclair ≡



*r* salereset ≡

*R* clair  si *v* 0 = 0 ∧ *v* = 0

0 sinon

*G* sset − *G* warmaccess si *v* 0 = *v* ∧ *v* 0 = 0



*G* sréinitialiser − *G* warmaccess si *v* 0 = *v* ∧ *v* 0 = 0



 0 sinon

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0x56 | SAUTER | 1 | 0 Modifier la programme compteur. | |
|  |  |  |  | *J* SAUT ( ***µ*** ) ≡ ***µs [* 0** ]  Cette possède la effet de en écrivant mentionné valeur pour ***µ*** pc . Voir section [9.](#_bookmark52) |
| 0x57 | JUMPI | 2 | 0 | Conditionnellement modifier la programme compteur. |

*J* JUMPI

( ***µ*** ) ***µ* s** [0] si ***µs [* 1** ] = 0

***µ*** pc + 1 autrement

f≡

Cette possède la effet de en écrivant mentionné valeur pour ***µ*** pc . Voir section [9.](#_bookmark52)

0x58 PC 0 1 Avoir la valeur de la programme compteur *avant* pour la incrément correspondant pour cette instruction.

***µs [* 0** ] ≡ ***µ*** pc

ÉTHERUM : UN SÉCURISE DÉCENTRALISÉ GÉNÉRALISÉ TRANSACTION COMPTABLE **BERLIN VERSION** 35

0x59 MSIZE 0 1 Avoir la Taille de actif Mémoire dans octets.

***µs [* 0** ] ≡ 32 ***µ*** je

0x5a GAZ 0 1 Avoir la montant de disponible gaz, y compris la correspondant réduction pour la Coût de cette instruction.

***µs [* 0** ] ≡ ***µg*** \_

0x5b JUMPDEST 0 0 Marquer un valide destination pour saute.

Cette opération possède non effet sur machine Etat pendant exécution.

# années 60 & années 70 : Pousser Opérations Valeur Mnémonique *δ α* La description

0x60 PUSH1 0 1 Lieu 1 octet Objet sur pile.

***µ*****s** [0] ≡ *c* ( ***µ*** p c + f 1)

where *c*(*x*) ≡

*I***b**[*x*] if *x <* ll*I***b**ll

0 otherwise

|  |  |  |
| --- | --- | --- |
|  | | Le octets sont lis dans doubler depuis la programme codes octets déployer.  Le une fonction *c* assure la octets défaut pour zéro si elles ou ils étendre passé la limites. Le octet est aligné à droite (prend la le plus bas important lieu dans gros endien). |
| 0x61 | PUSH2 | 0 1 Lieu 2 octets Objet sur pile. |

. . .

***µs [* 0** ] ≡ ***c*** ( ***µpc*** \_ + 1) *. . .* ( ***µpc*** \_ + 2)

avec ***c*** ( ***x*** ) ≡ ( *c* ( ***x*** 0 ) *, ..., c* ( ***x****X* − 1 )) avec *c* comme défini comme au dessus de.

Le octets sont aligné à droite (prend la le plus bas important lieu dans gros endien).

. .

0x7f PUSH32 0 1 Lieu 32 octets (plein mot) Objet sur pile.

***µs*** [0 ***]* ≡** c( ***µpc*** \_ + 1) *. . .* ( ***µpc*** \_ + 32) où ***c*** est défini comme au dessus de.

Le octets sont aligné à droite (prend la le plus bas important lieu dans gros endien).

# années 80 : Reproduction Opérations

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Valeur** | **Mnémonique** | *δ* | *α* | **La description** |
| 0x80 | DUP1 | 1 | 2 | Dupliquer 1er pile Objet. |
| 0x81 | DUP2 | 2 | 3 | ***µs [* 0** ] ≡ ***µs [* 0** ]  Dupliquer 2e pile Objet. |
|  |  |  |  | ***µs [* 0** ] ≡ ***µs [* 1** ] |

. ....

0x8f DUP16 16 17 Dupliquer 16e pile Objet.

***µs [* 0** ] ≡ ***µs* [** 15]

# Années 90 : Échanger Opérations

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Valeur** | **Mnémonique** | *δ* | *α* | **La description** |
| 0x90 | SWAP1 | 2 | 2 | Échanger 1er et 2e pile éléments.  ***µs [* 0** ] ≡ ***µs [* 1** ]  ***µs [* 1** ] ≡ ***µs [* 0** ] |
| 0x91 | SWAP2 | 3 | 3 | Échanger 1er et 3ème pile éléments. |

. . .

***µs [* 0** ] ≡ ***µs* [** 2]

***µs* [** 2] ≡ ***µs [* 0** ]

. .

0x9f SWAP16 17 17 Échangez la 1ère et la 17ème pile éléments.

***µs [* 0** ] ≡ ***µs* [** 16]

***µs* [** 16] ≡ ***µs [* 0** ]

ÉTHERUM : UN SÉCURISE DÉCENTRALISÉ GÉNÉRALISÉ TRANSACTION COMPTABLE **BERLIN VERSION** 36

# a0s : Enregistrement Opérations

Pour tout enregistrement opérations, la Etat monnaie est pour ajouter un Additionnel Journal entrée sur pour la sous-état Journal séries:

*A* **l** ≡ *A* **l** · ( *je* une *,* **t** *,* ***µ* m** [ ***µ* s** [0] *. . .* ( ***µ* s** [0] + ***µ* s** [1] − 1)]) et pour mettre à jour la Mémoire consommation compteur: ***µ*** je ≡ *M* ( ***µ*** je *,* ***µs [* 0** ] *,* ***µs [* 1** ])

Le entrée sujet séries, **t** , diffère par conséquent:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Valeur** | **Mnémonique** | *δ* | *α* | **La description** |
| 0xa0 | LOG0 | 2 | 0 | Ajouter Journal record avec non les sujets. |
| 0xa1 | LOG1 | 3 | 0 | **t** ≡ ()  Ajouter Journal record avec une sujet. |
|  |  |  |  | **t** ≡ ( ***µs* [** 2]) |

. ....

0xa4 LOG4 6 0 Ajouter Journal record avec quatre les sujets.

**t** ≡ ( ***µs* [** 2] *,* ***µs* [** 3] *,* ***µs* [** 4] *,* ***µs* [** 5])

ÉTHERUM : UN SÉCURISE DÉCENTRALISÉ GÉNÉRALISÉ TRANSACTION COMPTABLE **BERLIN VERSION** 37

# f0s : Système opérations

**Valeur Mnémonique** *δ α* **La description**

0xf0 CRÉER 3 1 Créer un Nouveau Compte avec associé code.

**je** ≡ ***µ* m** [ ***µ* s** [1] *. . .* ( ***µs [* 1** ] + ***µs* [** 2] − 1)]

*[ζ](#_bookmark43)* ≡ ∅

( ***σ****,* ***µg*** *,* \_ *UN, z,* **o** ) ≡

   [Λ (](#_bookmark44) ***σ*** ∗ *, UN, je* un *, je* o *, L* ( ***µ*** g ) *, je* p *,* ***µs [* 0** ] *,* **je** *, je* e + 1 *, , \_ je* w ) si ***µs [* 0** ] ***σ*** [ *je* une ] b

  ***σ*** *,* ***µg*** *,* \_ *UN,* 0 *,* () autrement

∧ *I <* 1024e

***σ*** ∗ ≡ ***σ*** à l'exception ***σ*** ∗ [ *je* une ] n = ***σ*** [ *je* une ] n + 1

***µs [* 0** ] ≡x *\_*

où *X* = 0 si *z* = 0, c'est à dire, la [Contrat création processus manqué,](#_bookmark48) ou alors *je* e = 1024

(la m axi m um appel profondeur J'imite \_ est atteint ) \_ ou alors ***µs [* 0** ] *>* ***σ*** [ *je* une ] b (solde de la l'appelant est aussi bas pour remplir la valeur transférer); et sinon *x* = ADDR ( *je* une *,* ***σ*** [ *je* une ] n *, ζ,* **je** ), la adresse de la nouvellement créé Compte [(86).](#_bookmark45)

***µ*** je ≡ *M* ( ***µ*** je *,* ***µs [* 1** ] *,* ***µs* [** 2])

***µo* \_** ≡ ()

Ainsi la opérande Commande est: valeur, saisir décalage, saisir Taille.

0xf1 APPEL 7 1 Message-appel dans un Compte.

( ***σ****, g, UN, X,* **o** ) ≡

[Θ (](#_bookmark47) ***σ*** *, UN , je* un *, je* o *, t, t, C* CALGAS ( ***σ*** *,* ***µ*** *, A* ) *,*

*je* p *,* ***µs* [** 2] *,* ***µs* [** 2] *,* **je** *, je* e + 1 *, je* w )

∗





**i** ≡ ***µ*m**[***µ*s**[3] *. . .* (***µ*****s**[3] + ***µ*s**[4] − 1)]

si ***µs* [** 2] ***σ*** [ *je* une ] b ∧

*je* e *<* 1024

( ***σ*** *, g, UN,* 0 *,* ()) sinon

*n* ≡ min( { ***µ* s** [6] *,* ll **o** ll} )

***µ* m** [ ***µ* s** [5] *. . .* ( ***µs* [** 5] + *n* − 1)] = **o** [0 *. . .* ( *n* − 1)]

***µo* \_** = **o**

***µg*** \_ ≡ ***µg*** \_ + *g*

***µs [* 0** ] ≡x *\_*

*Un* ∗ ≡ *UN* à l'exception *Un* ∗ **un** ≡ *Un* **un** ∪ { *t* }

*t* ≡ ***µ*** [1] mode 2 160

**s**

***µ*** je ≡ *M* ( *M* ( ***µ*** je *,* ***µs*** [3] *,* **µs [** 4]) *,* ***µs [* 5** ] *,* ***µs* [** 6])

où *X* = 0 si la [code exécution pour cette opération manqué,](#_bookmark50) ou alors si

***µs* [** 2] *>* ***σ*** [ *je* une ] b (ne pas pas assez fonds) ou alors *je* e = 1024 (appel profondeur limite atteint ) ; \_ *X* = 1 sinon.

Ainsi la opérande Commande est: gaz, pour, valeur, dans décalage, dans Taille, dehors décalage, dehors Taille.

*C* CAL L ( ***σ*** *,* ***µ*** *, A* ) ≡ *C* G A f SCA P ( ***σ*** *,* ***µ*** *, A* ) + *C* EXTR A ( ***σ*** *,* ***µ*** *, A* )

f≡

*C*

CALLGAS

(***σ****,* ***µ****, A*) ≡

*C*GASCAP(***σ****,* ***µ****, A*) + *G*callstipend if ***µ*s**[2] =

*C*GASCAP(***σ****,* ***µ****, A*) otherwise

0

*C* GASCAP

( ***σ*** *,* ***µ*** *, A* ) min { [*L*](#_bookmark141) ( ***µ*** g − *C* EXTRA ( ***σ*** *,* ***µ*** *, A* )) *,* ***µs [* 0** ] } si ***µg*** \_ ≥ *C* EXTRA ( ***σ*** *,* ***µ*** *, A* )

***µs [* 0** ] sinon

*C* EXTR A ( ***σ*** *,* ***µf*** *,* \_ *A* ) ≡ [*C* a accès s](#_bookmark140) ( *t, A* ) + *C* XFE R ( ***µ*** ) + *C* NE W ( ***σ*** *,* ***µ*** )

f≡

*C*

XFER

(***µ***) ≡

*G*callvalue if ***µ*s**[2] = 0

0 otherwise

*C* NOUVEAU

( ***σ*** *,* ***µ*** ) *G* nouveau compte si MORT ( ***σ*** *, t* ) ∧ ***µs* [** 2] 0

0 sinon

0xf2 CODE APPEL 7 1 Message-appel dans cette Compte avec un alternative comptes code.

Exactement équivalent pour APPEL à l'exception:



( ***σ****, g, UN, X,* **o** ) ≡

Θ ( ***σ*** *, A* ∗ *, je* un *, je* o *, je* un *, t, C* CALGAS ( ***µ*** ) *,*

*je* p *,* ***µs* [** 2] *,* ***µs* [** 2] *,* **je** *, je* e + 1 *, je* w )



si ***µs* [** 2] ***σ*** [ *je* une ] b ∧

*je* e *<* 1024

( ***σ*** *, g, UN,* 0 *,* ()) sinon

Noter la monnaie dans la Quatrième paramètre pour la appel [Θ](#_bookmark47) depuis la 2e pile valeur ***µs [* 1** ] (comme dans APPEL ) pour la cadeau adresse *je* un . Cette moyens que la destinataire est dans fait la même Compte comme à cadeau, simplement que la code est écrasé.

0xf3 RETOUR 2 0 Arrêt exécution retour production Les données.

*H* RETOUR ( ***µ*** ) ≡ ***µ* m** [ ***µ* s** [0] *. . .* ( ***µs [* 0** ] + ***µs [* 1** ] − 1)]

Cette possède la effet de s'arrêter la exécution à cette indiquer avec production défini.

Voir section [9.](#_bookmark52)

***µ*** je ≡ *M* ( ***µ*** je *,* ***µs [* 0** ] *,* ***µs [* 1** ])

ÉTHERUM : UN SÉCURISE DÉCENTRALISÉ GÉNÉRALISÉ TRANSACTION COMPTABLE **BERLIN VERSION** 38

0xf4 DÉLÉGUÉ APPEL 6 1 Message-appel dans cette Compte avec un alternative comptes code, mais

persistant la courant valeurs pour *expéditeur* et *valeur* .

Par rapport avec APPEL , APPEL DÉLÉGUÉ prend une moins de arguments. Le omis argument est ***µs*** [2] **.** Comme un résultat, ***µs* [** 3], ***µs* [** 4], ***µs* [** 5] et ***µs* [** 6] dans la définition de APPEL devrait respectivement être remplacé avec ***µs* [** 2], ***µs* [** 3], ***µs* [** 4] et ***µs*** [5] **.** Autrement ce est équivalent pour APPEL à l'exception:

   Θ( ***σ*** *, A* ∗ *, je* suis *, je* o *, je* un *, t, C* CALLGA S ( ***µ*** ) *,*

(***σ****, g, A, x,* **o**) ≡

*I ,* 0*, I ,* **i***, I*

+ 1*, I*

)

p

v

e

w

si *je*

*<* 1024

e

( ***σ*** *, g, UN,* 0 *,* ()) sinon



Noter la changements (dans une addition pour que de la Quatrième paramètre) pour la seconde et neuvième paramètres pour la appel [Θ.](#_bookmark47)

Cette moyens que la destinataire est dans fait la même Compte comme à cadeau, simplement que la code est écrasé *et* la le contexte est presque entièrement identique.

|  |  |  |  |
| --- | --- | --- | --- |
| 0xf5 | CRÉER2 | 4 | 1 Créer un Nouveau Compte avec associé code. Exactement équivalent pour CRÉER à l'exception:  Le sel *ζ* ≡ ***µs*** [3] **.** |
| 0xfa | APPEL STATIQUE | 6 | 1 Statique message-appel dans un Compte. Exactement équivalent pour APPEL à l'exception:  Le argument ***µs* [** 2] est remplacé avec 0.  Le Plus profond argument ***µs* [** 3], ***µs* [** 4], ***µs* [** 5] et ***µs* [** 6] sont respectivement remplacé avec ***µs* [** 2], ***µs* [** 3], ***µs* [** 4] et ***µs*** [5] **.**  Le dernière argument de [Θ](#_bookmark47) est ⊥ . |
| 0xfd | REVENIR | 2 | 0 Arrêt exécution revenant Etat changements mais retour Les données et restant gaz. Le effet de cette opération est décrit dans [(143).](#_bookmark55)  Pour la gaz calcul, nous utiliser la Mémoire expansion une fonction,  ***µ*** je ≡ *M* ( ***µ*** je *,* ***µs [* 0** ] *,* ***µs [* 1** ]) |
| 0xfe | INVALIDE | ∅ | ∅ Désigné invalide instruction. |
| 0xff | AUTO-DESTRUCTION | 1 | 0 Arrêt exécution et S'inscrire Compte pour plus tard effacement. |

*Un* **s** ≡ *Un* **s** ∪ { *je* un }

*Un* **un** ≡ *Un* **un** ∪ { *r* }



∅ if ***σ***[*r*] = ∅ ∧ ***σ***[*I*a]b = 0

***σ***[ *r* ] ≡

( ***σ*** [ *r* ] n *,* ***σ*** [ *r* ] b + ***σ*** [ *je* une ] b *,* ***σ*** [ *r* ] **s** *,* ***σ*** [ *r* ] c ) si *r*  = *je* un

 ( ***σ*** [ *r* ] n *,* 0 *,* ***σ*** [ *r* ] **s** *,* ***σ*** [ *r* ] c ) sinon



où *r* = ***µ*** [0] mode 2 160

**s**

***σ***[ *je* une ] b = 0

*C* AUTODESTRUCTION

( ***σ*** *,* ***µ*** ) ≡ *g*

auto-destruction

f

+ 0 si *r* ∈ [*Un* **un**](#_bookmark39)

*G* coldaccountacces autrement

f

+ Compte *G* ne w \_ \_ si MORT ( ***σ*** *, r* ) ∧ ***σ*** [ *je* une ] b = 0

0 sinon

annexe JE. Genèse Bloc

Le genèse bloc est 15 éléments, et est spécifié Donc:

(331) 0 256 *,* KEC RLP () *,* 0 160 *, étatRoot,* 0 *,* 0 *,* 0 2048 *,* 2 34 *,* 0 *,* 0 *,* 3141592 *, temps,* 0 *,* 0 256 *,* KEC (42) *,* () *,* ()

Où 0 256 fait référence au hachage parent, un hachage de 256 bits composé uniquement de zéros ; 0 160 fait référence à l'adresse du bénéficiaire, un 160 bits hachage composé uniquement de zéros ; 0 2048 fait référence à la floraison du journal, 2048 bits de tous les zéros ; 2 34 fait référence à la difficulté; l'essai de transaction racine, racine de trie de réception, gaz utilisé, numéro de bloc et données supplémentaires sont tous deux 0, ce qui équivaut au tableau d'octets vide. Le séquences de tous les deux ommers et transactions sont vider et représentée par (). KEC (42) réf rs pour e e Keccak-256 hacher de

a byte array of length one whose first and only byte is of value 42, used for the nonce. KEC RLP ()

hash of the ommer list in RLP, both empty lists.

value refers to the

La série de preuves de concept comprend une prémine de développement, faisant du hachage de la racine d'état une valeur *stateRoot* . Aussi *time* sera défini sur l'horodatage initial du bloc de genèse. La documentation la plus récente doit être consultée pour ceux valeurs.

ÉTHERUM : UN SÉCURISE DÉCENTRALISÉ GÉNÉRALISÉ TRANSACTION COMPTABLE **BERLIN VERSION** 39

annexe J Éthash

* 1. **Définitions.** Nous employer la Suivant définitions : Nom ValeurDescription

*J* motoctets  4 octets en mot.

*J* datasetinit  2 30  octets dans base de données à genèse. *J* datasetgrowth  2 23 Croissance de l'ensemble de  données par époque. *J* cacheinit  2 24  octets dans cache à la genèse. *J* cachecroissance  2 17  Croissance du cache par époque.

|  |  |  |
| --- | --- | --- |
| *J* époque | 30000 | Blocs par époque. |
| *J* mixbytes | 128 | mélanger longueur dans octets. |
| *J* octets de hachage | 64 | Hacher longueur dans octets. |
| *J* parents | 256 | Numéro de parents de chaque base de données élément. |
| Tours de cache *J* | 3 | Numéro de Les manches dans cache production. |
| *J* accède | 64 | Numéro de accède dans hashimoto boucle. |

* 1. **Taille de base de données et cache.** Le Taille pour d'Ethash cache **c** ∈ B et base de données **ré** ∈ B dépendre sur la époque, qui dans tourner dépend sur la bloc Numéro.

(332) *E* ( *H* ) = *Salut* \_

epoch

epoch

i

*J*

La taille de la croissance du jeu de données par *J* datasetgrowth octets, et la taille du cache par *J* cachegrowth octets, chaque époque. Dans Afin d'éviter une régularité conduisant à un comportement cyclique, la taille doit être un nombre premier. La taille est donc réduite d'un multiple de *J* mixbytes , pour le jeu de données, et *J* hashbytes pour le cache. Soit *d* taille = ll **d** ll soit la taille de l'ensemble de données. Lequel est calculé en utilisant

1. taille *d* = *E* prime ( *J* datasetinit + Croissance du jeu de données *J* · *E* époque − *J* mixbytes *, J* mixbytes )

Le Taille de la cachette, taille *c* , est calculé en utilisant

1. taille *C* = *E* prime ( *J* cacheinit + Croissance du cache *J* · *E* époque − *J octets de* hachage *, J* octets de hachage )

f

1. *E*

prime

( *x, y* ) = *x* si *x/y* ∈ N

*E* premier ( *x* − 2 · *y, y* ) autrement

* 1. **Base de données génération.** Dans Commande pour produire la base de données nous avoir besoin la cache **c** , qui est un déployer de octets. Ce dépend sur la cache Taille taille *C* et la la graine hacher **s** ∈ B 32 .
     1. *Hachage de graines.* Le hachage de graines est différent pour chaque époque. Pour la première époque c'est le hachage Keccak-256 d'une série de 32 octets de des zéros. Pour tous autre époque ce est toujours la Keccak-256 hacher de la précédent la graine hacher:

1. **s** = *C* seedhash ( *H* i )
2. *C* seedhash

i

( *H* ) = f **0** 32  si *Ep* o c h ( *H* je ) \_ \_ = 0

Avec **0** 32 être 32 octets de des zéros.

KEC ( *C* seedhas h ( *H* i − *J* e p o c h )) autrement

* + 1. *Cachette.* Le processus de production de cache consiste à utiliser le hachage de départ pour d'abord remplir séquentiellement des octets de taille *c de* Mémoire, alors effectuer Tours de cache *J* passe de la RandMemoHash algorithme créé par [Lerner [2014](#_bookmark85) ]. Le initial cache **c** , être un déployer de tableaux de Célibataire octets, sera être construit comme suit.

Nous définir la déployer **c** *je* , qui consiste de 64 Célibataire octets, comme la *je* e élément de la initial cache :

f

1. **c** *je*

Donc **c** pouvez être défini comme

= KEC512 ( **s** ) si *je* = 0

KEC512 ( **c** *je* − 1 ) autrement

1. **c** [ *je* ] = **c** *je* ∀ *je < n*

1. *n* = taille *C*

*J* octets de hachage

Le cache est calculé par effectuer Tours de cache *J* Les manches de la RandMemoHash algorithme pour la initial cache **c**:

1. **c** = *E* rondes de cache ( **c** *, J* tours de cache )
2. *E tours de* cache ( **x** *, y* ) =

*E* RMH ( **x** ) si *y* = 1

**x** if *y* = 0

  *E* ca c heround s ( *E* RM H ( **x** ) *, y* − 1) autrement

ÉTHERUM : UN SÉCURISE DÉCENTRALISÉ GÉNÉRALISÉ TRANSACTION COMPTABLE **BERLIN VERSION** 40

Où un Célibataire tour modifie chaque sous-ensemble de la cache comme suit :

(343) *E* RMH ( **x** ) = *E* rmh ( **x** *,* 0) *, E* rmh ( **x** *,* 1) *, , E* rmh ( **x** *, n* − 1)

(344) *E* rmh ( **x** *, je* ) = KEC512 ( **x**[( *je* − 1 + *n* ) mode *n* ] ⊕ **X**[ **x**[ *je* ][0] mode *n* ])

avec **X** = **X** à l'exception **X**[ *j* ] = *E* rmh ( **x** *, j* ) ∀ *j < je*

* + 1. *Plein base de données calcul.* Essentiellement, nous combiner Les données depuis *J* parents pseudo-aléatoire choisi cache nœuds, et hacher que pour calculer la base de données. Le entier base de données est alors généré par un Numéro de éléments, chaque *J* octets de hachage octets dans Taille:

(345) **ré** [ *je* ] = *E* élément de jeu de données ( **c** *, je* ) ∀ *je <*

taille *d*

octets de hachage

*J*

Dans Commande pour calculer la Célibataire Objet nous utiliser un algorithme inspiré par la FNV hacher [(Glenn Chasseur [1991])](#_bookmark97) dans quelques cas comme un non associatif remplaçant pour XOR.

(346) *E* ( **x** *,* **y** ) = ( **x** · (0x01000193 ⊕ **y** )) mode 2 32

FNV

Le Célibataire Objet de la base de données pouvez à présent être calculé comme:

1. *E* ensemble de données m ( **c** *, je* ) = *É* pare n t s ( **c** *, je,* − 1 *,* ∅ )

f

1. *E*

parents

( **c** *, je, p,* **m** ) = *E* parents ( **c** *, je, p* + 1 *, E* mix ( **m** *,* **c** *, je, p* + 1)) si *p < J* parents − 2

*E* mix ( **m** *,* **c** *, je, p* + 1) sinon

f KEC512 ( **c** [ *je* mod \_ \_ *c* taille e ] ⊕ *je* ) si *p* = 0

(349) *E*mix(**m***,* **c***, i, p*) =

**m***,* **c**[*E*

(*i* ⊕ *p,* **m**[*p* mod l*J*

J]) mod *c*

] otherwise

*E*

FNV

FNV

hashbytes

*/J*wordbytes

size

* 1. **Fonction de preuve de travail.** Essentiellement, nous maintenons un "mix" *J* mixbytes octets de large, et récupérer séquentiellement à plusieurs reprises *J* mixbytes octets du jeu de données complet et utilisez la fonction *E* FNV pour le combiner avec le mix. *J* mixbytes octets séquentiels accéder sont utilisé alors que chaque tour de la algorithme toujours extractions un plein page depuis RAM, minimiser Traduction regarder de côté amortir manque qui ASIC voudrais théoriquement être capable pour éviter de.

Si la production de cette algorithme est au dessous de la voulu cibler, alors la nonce est valide. Noter que la En plus application de KEC à la fin assure qu'il existe un nonce intermédiaire qui peut être fourni pour prouver qu'au moins un petit la quantité de travail a été effectuée; cette vérification PoW externe rapide peut être utilisée à des fins anti-DDoS. Il sert également à fournir statistique assurance que la résultat est un impartial, 256 bit Numéro.

Le Fonction PoW Retour un déployer avec la comprimé mélanger comme son première Objet et la Keccak-256 hacher de la enchaînement de la comprimé mélanger avec la la graine hacher comme la seconde Objet:

(350)

PoW ( *H* n *, H* n *,* **ré** ) = { **m** c ( KEC ( RLP ( *L* H ( *H* n ))) *, H* n *,* **d** ) *,* KEC ( **s** h ( KEC ( RLP ( *L* H ( *H* n ))) *, H* n ) + **m** c ( KEC ( RLP ( *L* H ( *H* n ))) *, H* n *,* **d** )) }

Avec *H* n être la hacher de la entête sans pour autant la nonce. Le comprimé mélanger **m** c est obtenu comme suit :

*n* mélanger

1. **m** c ( **h** *,* **n** *,* **ré** ) = *E* compresse ( *E* accède s ( **d** *,* \_**s** h ( **h** *,* **n** ) *,* **s** h ( **h** *,* **n** ) *,* − 1) *,* − 4)

*je* =0

Le la graine hacher être:

1. **s** h ( **h** *,* **n** ) = KEC512 ( **h** + *E* re v e r t ( **n** )) *E* revert ( **n** ) Retour la inversé octets séquence de la nonce **n** :
2. *E* re v er t ( **n** )[ *i* ] = **n** [ ll **n** ll − *je* ]

Nous Remarque que la Opérateur "+" entre deux octet séquences résultats dans la enchaînement de tous les deux séquences.

Le base de données **ré** est obtenu comme décrit dans section [J.3.3.](#_bookmark158) Le Numéro de répliqué séquences dans la mélanger est:

1. *n*

mélanger

= *J* mixbytes

octets de hachage

*J*

Dans Commande pour ajouter Aléatoire base de données nœuds pour la mélanger, la *E* accède une fonction est utilisé:

f

1. *E*

accède

( **ré** *,* **moi** *,* **s** *, je* ) = *E* mixdataset ( **d** *,* **moi** *,* **s** *, je* ) si *je* = *J* accède − 2

*E* accède à ( *E* mixdataset ( **d** *,* **moi** *,* **s** *, je* ) *,* **s** *, je* + 1) autrement

1. *E* mixdataset ( **d** *,* **moi** *,* **s** *, je* ) = *E* FNV ( **m** *, E* nouvelledonnée ( **d** *,* **moi** *,* **s** *, je* )) *E* nouvelles données Retour un déployer avec *n* mélanger éléments:
2. *E* ( **ré** *,* **moi** *,* **s** *, je* )[ *j* ] = **ré** [ *E* ( *je* ⊕ **s** [0] *,* **m** [ *je* mode *J* mixbytes ]) mode *d* taille */J* octets de hachage · *n*

newdata

FNV

*J*

wordbytes

*n*

mix

mix

+ *j* ] ∀ *j < n*

mix

ÉTHERUM : UN SÉCURISE DÉCENTRALISÉ GÉNÉRALISÉ TRANSACTION COMPTABLE **BERLIN VERSION** 41

Le mélanger est comprimé comme suit :

f

1. *E*

compresse

( **moi** *, je* ) = **moi** si *je* ;;: ll **m** ll − 8

*E* compresser ( *E* FNV ( *E* FNV ( *E* FNV ( **m** [ *i* + 4] *,* **m** [ *je* + 5]) *,* **m** [ *je* + 6]) *,* **m** [ *je* + 7]) *, je* + 8) autrement

annexe K Anomalies sur la Principale Réseau

K.1. **Suppression d'un compte malgré la panne d'essence.** Au bloc 2675119, dans la transaction 0xcf416c536ec1a19ed1fb89e 4ec7ffb3cf73aa413b3aa9b77d60e4fd81a4296ba, un compte à l'adresse 0x03 a été appelé et une panne de gaz s'est produite pendant l'appel. Contre l'équation [(207),](#_bookmark120) cela a ajouté 0x03 dans l'ensemble des adresses touchées, et cette transaction est devenue ***σ*** [0 *x* 03] dans ∅ .

annexe L Lister de mathématique symboles

Symbole Latex Commande La description

\bigvee Ceci est la moins plus haut lié, suprême, ou alors rejoindre de tout éléments exploité sur. Ainsi ce est la le plus grand élément de tel éléments [(Davey et Priestley [2002](#_bookmark88) ]).