Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και
Υπολογιστών

Service Predictability in Blockchain Systems

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΓΕΩΡΓΙΟΣ ΤΣΙΡΩΝΗΣ

Επιβλέπων : Αριστείδης Παγουρτζής
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούνιος 2025

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και
Υπολογιστών

Service Predictability in Blockchain Systems

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΓΕΩΡΓΙΟΣ ΤΣΙΡΩΝΗΣ

Επιβλέπων : Αριστείδης Παγουρτζής
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 30η Ιουνίου 2025.

....	....	....
Αριστείδης Παγουρτζής Καθηγητής Ε.Μ.Π.	Δημήτριος Φωτάκης Καθηγητής Ε.Μ.Π.	Νικόλαος Λεονάρδος Επίκουρος Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούνιος 2025

....

Γεώργιος Τσιρώνης

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Dedicated to the Metsovite benefactors

Georgios Averof, Michail & Eleni Tositsa and Nikolaos Stournaris
for giving me and countless others the opportunity to study in this institute, which defined me as a person.

Περίληψη

Τα συστήματα blockchain προσφέρουν μία καινοτόμο αποκεντρωμένη προσέγγιση στις προκλήσεις που προκύπτουν σε συμβατικά μοντέλα με αποτέλεσμα, συγκεντρώνουν αυξανόμενη προσοχή. Ωστόσο, ο διαθέσιμος χώρος για συναλλαγές είναι πεπερασμένος και αποτελεί περιορισμό στη δυνατότητα του συστήματος να εξυπηρετήσει την αυξανόμενη ζήτηση. Έτσι τα συστήματα αυτά καθίστανται αναξιόπιστα και η εξυπηρέτηση που προσφέρουν χαρακτηρίζεται από απρόβλεπτες καθυστερήσεις και χρεώσεις.

Στην εργασία αυτή προτείνεται μία λύση στο πρόβλημα της αβεβαιότητας στην εξυπηρέτηση των συστημάτων blockchain, βασισμένη σε συστήματα σταθερής χρέωσης. Δίνουμε μία περιγραφή του προβλήματος και της μέχρι τώρα σχετικής εργασίας. Παρουσιάζουμε διαφορετικές προσεγγίσεις στο πρόβλημα, όπως οι μηχανισμοί χρέωσης συναλλαγών και άλλες που κάνουν χρήση της τεχνολογίας έξυπνων συμβολαίων για να δημιουργήσουν πρωτόκολλα που θα λειτουργούν πάνω σε ένα υπάρχον σύστημα.

Παρουσιάζουμε μία σύνοψη των λειτουργικοτήτων του Cardano, ενός συστήματος σταθερής χρέωσης, το μοντέλο χρέωσης στο οποίο βασίζουμε τη δουλειά μας. Το Cardano θέτει ως μία από τις βασικές του προτεραιότητες το πρόβλημα της επεκτασιμότητας και εισάγει καινοτόμες μεθόδους για την αντιμετώπισή του.

Στο κύριο μέρος της εργασίας, παρουσιάζουμε το πρωτόκολλό μας για προβλέψιμη εξυπηρέτηση. Στο πρωτόκολλο γίνεται χρήση της τεχνολογίας έξυπνων συμβολαίων για τη δημιουργία ενός παραγώγου συνδεδεμένο με τον χώρο σε μπλοκ. Το παράγωγο αποτελεί μία συμφωνία μεταχύ δύο χρηστών, του Χρήστη και του Ασφαλιστή, την οποία ο τελευταίος προμηθεύει τον πρώτο με εξασφαλισμένο χώρο συναλλαγών σε ένα μελλοντικό μπλοκ.

Υποστηρίζουμε ότι οι εγγυήσεις που προσφέρονται από ένα πρωτόκολλο βασισμένο σε συστήματα σταθερής χρέωσης, δεν είναι επιτεύξιμες σε συστήματα δυναμικής χρέωσης όπως το Ethereum. Αναλύοντας την κοινή ωφέλεια των δύο πλευρών, δείχνουμε πως στα δυναμικά συστήματα δεν είναι δυνατή η εγγύηση εξυπηρέτησης χωρίς ενδεχόμενη ζημιά.

Καταλήγουμε αποδεικνύοντας πως το πρωτόκολλό μας εξασφαλίζει προβλέψιμη εξυπηρέτηση, χρησιμοποιόντας μαθηματική μοντελοποίηση βασισμένη στη θεωρία παιγνίων.

Λέξεις κλειδιά

Αλυσίδες Συναλλαγών, Θεωρία Παιγνίων, Χώρος Συναλλαγών σε Μπλοκ, Συμφόρηση Δικτύου Blockchain, Απρόβλεπτες Καθυστερήσεις, Αστάθεια Τιμών, Παράγωγα Χώρου Συναλλαγών, Σχεδιασμός Μηχανισμών, Έξυπνα Συμβόλαια, Μηχανισμοί Χρέωσης Συναλλαγών

Abstract

Blockchain systems provide a novel decentralized approach to the challenges found in conventional setups. This leads to them receiving increasing attention and traffic and thus, the need for scalability emerges as a prevalent research topic. However, block space being a scarce resource severely limits the chain's ability to accommodate the rising demand. As a result, systems are rendered unreliable as the provided service suffers from unpredictable delays and prices.

This work proposes a solution to the problem of service predictability in blockchain systems, built on top of a fixed-fees model. We give a clear overview of the problem and the related work done so far. We look into different approaches to the problem, solutions like transaction fee mechanisms are based on the system's lower level as they modify its core mechanism, while others utilize smart contract functionality to create a protocol on top of an existing system.

We present in detail Cardano, one of the most prominent fixed-fees systems, the model upon which we base our main results. By introducing many novel functionalities, Cardano tackles the scalability problem, one of its main foundational pillars. We provide interested readers with a compact synopsis of those functionalities, along with sources for further reading.

In the main part of this work, we introduce our predictable service protocol. The protocol makes use of smart contract technology to establish a block space derivative. The derivative is an agreement between two users, a User and an Insurer, where the latter provides the first with a guaranteed block space allocation at a future moment in time.

We argue that the guarantees ensured in a fixed-fee model cannot be achieved in a dynamic-fee model like Ethereum. By analyzing the joint utility of the two parties, we show that in dynamic fees there is no way to securely guarantee the allocation without suffering potential losses.

We conclude the work by proving that our protocol provides predictable service, by creating a mathematical model based on game theory notions.

Key words

Blockchain, cryptocurrency, Scalability, Service Predictability, Block Space, Blockchain Network Congestion, Unpredictable Delays, Price Volatility, Block Space Derivative, Future Allocation, Fixed-Fee Models, Dynamic-Fee Models, Cardano, EUTXO Model, Game Theory, Mechanism Design, Subgame Perfect Equilibrium (SPE), Transaction Prioritization, Blockchain Service Predictability, Cardano Smart Contracts, Transaction Fee Mechanisms (TFM)

Ευχαριστίες

Αυτή η διπλωματική δεν θα είχε γίνει πραγματικότητα χωρίς τη συμπαράσταση και τη βοήθεια πολλών ανθρώπων που πίστεψαν σε εμένα. Ιδιαίτερα θα ήθελα να ευχαριστήσω τον καθηγητή μου Αριστείδη Παγουρτζή και τον Άγγελο Κιαγιά που μου έδωσαν την ευκαιρία να ασχοληθώ με ένα τόσο ενδιαφέρον θέμα. Την Δανάη Μπάλα που με στήριξε κατά την περίοδο της μελέτης αυτού του επιστημονικού πεδίου και με βοήθησε να επιλέξω τελικά θέμα. Τον Γιώργο Παναγιωτάκο που με καθοδήγησε καθ'όλη την περίοδο της έρευνας και της συγγραφής αυτής της διπλωματικής και χωρίς τον οποίο δεν θα είχα καταφέρει να την ολοκληρώσω.

Θα ήθελα να ευχαριστήσω τους ανθρώπους εκείνους που με καθοδήγησαν στη διαδρομή μου μέχρι την εργασία αυτή. Τον καθηγητή μου Δημήτρη Φωτάκη, ο οποίος με ενέπνευσε να ασχοληθώ με τον τομέα της θεωρητικής πληροφορικής, των αλγορίθμων και της θεωρίας παιγνίων. Η συμβολή του υπήρξε καθοριστική για ολόκληρη την ακαδημαϊκή μου πορεία. Τον καλό μου φίλο Βασίλη Μηλιά, χωρίς τον οποίο δεν θα ήμουν σε αυτήν τη σχολή και ο οποίος μου έδωσε την καλύτερη συμβουλή σε μία από τις πιο μπερδεμένες στιγμές της ζωής μου.

Ευχαριστώ επίσης όλους τους ανθρώπους που με συντρόφευσαν όλα αυτά τα χρόνια. Όλους εκείνους που διαβάσαμε μαζί, παρακολουθήσαμε διαλέξεις και δώσαμε μαθήματα παρέα και όσους με άκουσαν να γκρινιάζω για αυτή τη σχολή. Θέλω να ευχαριστήσω τα παιδιά από τη θεατρική που πλαισίωσαν τις σπουδές μου και τα παιδιά από το Παπουλάκειο αναγνωστήριο χωρίς τα οποία δεν θα είχα κουράγιο για τα τελευταία βήματα.

Τέλος θέλω να ευχαριστήσω τους τρεις πιο κοντινούς μου ανθρώπους σε αυτό το ταξίδι και που η βοήθειά τους συγκεντρώνει όλα αυτά για τα οποία έγραψα. Την Αθηνά Τερζόγλου, της οποίας η βοήθεια ήταν ουσιαστική σε εργασίες και μαθήματα και η οποία υπήρξε παράγοντας σταθερότητας σε μια παρέα ιδιόρυθμων ανθρώπων. Τον Γιώργο Μπαμπίλη, ο οποίος ήταν πάντα εκεί να με ξελασπώνει, είτε το πρόβλημα ήταν το πρόγραμμα που δεν έτρεχε, είτε ήταν η πίεση που με ξεπερνούσε. Τον Κώστα Παπαδόπουλο, ο οποίος με στήριξε όσο κανένας τα τελευταία χρόνια. Σε κάθε βήμα αυτής της διαδρομής η βοήθειά του ήταν πανταχού παρούσα και είχε πάντα τον τρόπο να βάζει τάξη στο χάος. Χωρίς αυτόν αυτή η διπλωματική θα συνέχιζε την ήδη τεράστια πορεία της. Από το πρώτο έτος, μέχρι τις τελευταίες λέξεις αυτής της διπλωματικής αυτοί οι άνθρωποι ήταν δίπλα μου. Τους χρωστάω το πτυχίο μου.

Γεώργιος Τσιρώνης,

Αθήνα, 30η Ιουνίου 2025

Περιεχόμενα

Κατάλογος σχημάτων

Κατάλογος πινάκων

List of Algorithms

Κεφάλαιο 0

Εκτεταμένη Ελληνική Περίληψη

0.1 Αλυσίδες Συναλλαγών (Blockchain)

0.1.1 Συστήματα Blockchain

Η τεχνολογία Η τεχνολογία Blockchain απέκτησε δημοτικότητα όταν το 2008 ο έγινε η δημοσίευση του Bitcoin από τον Nakamoto. Ο Nakamoto βασίστηκε σε υπάρχουσα θεωρία κρυπτογραφίας και δημιούργησε ένα ψηφιακό αποκεντρωμένο κρυπτονόμισμα. Αντίθετα με τη συμβατική οικονομία στην οποία χρησιμοποιείται ένα κεντρικό σύστημα αποθήκευσης, το Bitcoin αποθηκεύει τις συναλλαγές σε ένα λογιστικό μητρώο αποτελούμενο από μία σειρά από μπλοκ, το οποίο διατηρείται σε αντίγραφα διαμοιρασμένα σε ένα δίκτυο. Τα μπλοκ σχηματίζουν μία αλυσίδα στην οποία κάθε μπλοκ συνδέεται με κτρυπτογραφικά εργαλεία με το προηγούμενο. Η νέα πληροφορία καταχωρείται στο μητρώο μονάχα μέσω επέκτασης της αλυσίδας και κάθε αλλαγή στην υπάρχουσα αλυσίδα γίνεται άμεσα αντιληπτή από το δίκτυο και απορρίπτεται.

Λογιστικά Μοντέλα Τα συστήματα blockchain χρησιμοποιούν διάφορα λογιστικά μοντέλα για να καταγράφουν το υπόλοιπο των χρηστών στο λογιστικό μητρώο. Παραθέτουμε τρία από αυτά με μία σύντομη περιγραφή.

- **UTxO**: Το λογιστικό μοντέλο που χρησιμοποιείται από το Bitcoin λέγεται UTxO. Το αποδίδουμε στα ελληνικά ως ”Μη Δαπανημένο Αποτέλεσμα Συναλλαγής”. Λειτουργεί με τη χρήση των UTxO ως νομισμάτων τα οποία έχουν έναν ιδιοκτήτη και μία αξία. Οι συναλλαγές καταναλώνουν τα νομίσματα αυτά και δημιουργούν νέα, με διαφορετικούς ιδιοκτήτες και αξίες.
- **Μοντέλο Λογαριασμών**: Το λογιστικό μοντέλο λογαριασμών χρησιμοποιείται από το Ethereum και λειτουργεί διατηρώντας έναν λογαριασμό για τον κάθε χρήστη του blockchain. Στον λογαριασμό διαφαίνεται η περιουσία του χρήστη και με κάθε συναλλαγή οι λογαριασμοί ανανεώνονται.
- **EUTxO**: Το λογιστικό μοντέλο που χρησιμοποιείται από το Cardano λέγεται EUTxO, δηλαδή εκτεταμένο UTxO. Αποτελεί μία επέκταση του μοντέλου UTxO, ώστε να μπορεί να υποστηρίζει έξυπνα συμβόλαια (smart contracts). Χρησιμοποιεί παρόμοια νομίσματα

UTxO με το Bitcoin, όμως τους δίνει τη δυνατότητα να αποθηκεύουν περισσότερες πληροφορίες σχετικά με την κατάσταση του blockchain.

Αλγόριθμοι Συναίνεσης Ο πρώτος αλγόριθμος που χρησιμοποιήθηκε για τη λειτουργία κρυπτονομίσματος ήταν η Απόδειξη Εργασίας (Proof of Work), ο οποίος δίνει δικαίωμα παραγωγής μπλοκ στον χρήστη που δαπανάει την περισσότερη ενέργεια για την επίλυση ενός προβλήματος. Ακολούθησε η Απόδειξη Μεριδίου (Proof of Stake), η οποία βασίζει την επιλογή παραγωγού μπλοκ σε μία τυχαία διαδικασία με βάση το οικονομικό μερίδιο που έχουν δηλώσει οι χρήστες για να συμμετέχουν στη διαδικασία.

Το Τρίλλημα των Blockchain Τα blockchain βασίζονται σε τρεις αξίες, τις οποίες σκοπεύουν να συνδυάσουν: την αποκέντρωση, την ασφάλεια και την επεκτασιμότητα. Ωστόσο, είναι κοινώς αποδεκτό πως όταν δίνεται βάση σε δύο από αυτές, θυσιάζεται η τρίτη και κάθε σύστημα πρέπει να επιλέξει που θα δώσει τη μέγιστη βαρύτητα.

0.1.2 Μηχανισμοί Χρέωσης Συναλλαγών (TFM)

Η κίνηση που μπορεί να εξυπηρετηθεί από τα συστήματα blockchain είναι περιορισμένη και καθώς περισσότεροι χρήστες προσέρχονται χρειάζεται ένας τρόπος περιορισμού τις εισόδου των συναλλαγών σε μπλοκ. Αυτόν και άλλους στόχους εξυπηρετούν οι μηχανισμοί χρέωσης συναλλαγών μέσω των οποίων οι χρήστες υποχρεούνται να πληρώσουν για να εισάγουν μια συναλλαγή σε μπλοκ.

Εξετάζουμε τρεις βασικούς μηχανισμούς

- **Πλειστηριασμός Πρώτης Τιμής (First Price Auction)**: Σε αυτόν τον μηχανισμό οι χρήστες κάνουν προσφορές για τις συναλλαγές τους. Οι χρήστες με τις μεγαλύτερες προσφορές εισάγουν τις συναλλαγές στο μπλοκ, αφού πρώτα πληρώσουν τιμή ίση με την προσφορά τους στον παραγωγό του μπλοκ.
- **Δυναμικές Χρεώσεις**: Στον μηχανισμό των Δυναμικών Χρεώσεων, οι χρήστες πρέπει να πληρώσουν ένα συγκεκριμένο ποσό για να βάλουν συναλλαγές σε μπλοκ. Το ποσό αυτό δεν το παίρνει ο παραγωγός του μπλοκ και αντ'αυτού καίγεται. Επιπλέον, σε κάθε μπλοκ η χρέωση ανανεώνεται ακολουθώντας την κίνηση, ώστε να κρατάει τον αριθμό των εισαχθέντων συναλλαγών σταθερό.
- **Σταθερές Χρεώσεις**: Στον μηχανισμό αυτό η χρέωση των συναλλαγών γίνεται με πάντα με τον ίδιο τρόπο. Υπολογίζεται με βάση το μέγεθος της συναλλαγής και πόρους που καταναλώνει ο τυχόν κώδικας που χρησιμοποιεί η συναλλαγή με την μορφή έξυπνου συμβολαίου.

0.1.3 Ανάλυση TFM

Εκτεταμένη έρευνα έχει γίνει για την αξιολόγηση των μηχανισμών χρεώσεων με βάση τη θεωρία παιγνίων και την προσπάθεια εύρεσης ενός ιδανικού μηχανισμού ο οποίο είναι συμβατός με τα κίνητρα των χρηστών. Αναφέρουμε τρεις δημοσιεύσεις των Roughgarden, Chung και Shi, οι οποίοι αξιολογούν υπάρχοντες μηχανισμούς και αποδεικνύουν ότι ένας τέτοιος μηχανισμός δεν μπορεί να υπάρξει.

- **Η δημοσίευση του Roughgarden**: Ο Roughgarden δημιουργεί ένα μοντέλο αξιολόγησης των μηχανισμών χρεώσεων και αξιολογεί με βάση αυτόν τους πιο συνήθεις μηχανισμούς.
- **Η δημοσίευση των Chung και Shi**: Χρησιμοποιούν ένα μοντέλο παρεμφερές με αυτό του Roughgarden και αποδεικνύουν ότι δεν υπάρχει μηχανισμός που να ικανοποιεί τις απαιτήσεις που έχουν θέσει ως επιθυμητές στο μοντέλο. Ωστόσο, κατασκευάζουν ένα μηχανισμό που προσφέρει ικανοποιητικά αποτελέσματα θυσιάζοντας την αποδοτικότητα.
- **Η δημοσίευση των Roughgarden, Chung και Shi**: Αποδεικνύουν ότι δεν μπορεί να υπάρξει μηχανισμός που να ικανοποιεί τις συνθήκες συμβατότητας για τα κίνητρα των χρηστών, ούτε για το αρχικό μοντέλο του Roughgarden.

0.1.4 Παράγωγα σε Περιβάλλοντα Blockchain

Οι μηχανισμοί χρέωσης χρησιμοποιούνται, μεταξύ άλλων στόχων, και για τη δημιουργία ενός πιο προβλέψιμου περιβάλλοντος εξυπηρέτησης στα συστήματα blockchain. Μια άλλη μέθοδος είναι παράγωγα συνδεδεμένα με τον διαθέσιμο χώρο συναλλαγών σε μπλοκ.

- Το Pitch Lake είναι ένα συμβόλαιο που αποδίδει αξία στον αγοραστή του αν η χρέωση για την εισαγωγή σε μπλοκ ανέβει πάνω από μία συγκεκριμένη τιμή. Αυτό το συμβόλαιο μπορεί να χρησιμοποιηθεί από χρήστες που στοχεύουν στη μελλοντική εισαγωγή μίας συναλλαγής τους ώστε να τους προστατέψει από απρόβλεπτες ανόδους στην τιμή της χρέωσης.
- Το Blockspace Tokenization, θεσπίζει ένα πρωτόκολλο αγοράς δελτίων τα οποία μπορούν να χρησιμοποιηθούν στη συνέχεια για την πληρωμή της εισαγωγής σε μπλοκ, αντί της κανονικής χρέωσηςη. Παρότι δεν είναι παράγωγο, παρουσιάζει κάποια κοινά χαρακτηριστικά στη χρήση του καθώς επιτρέπει στους χρήστες την προαγορά χώρου συναλλαγών.
- Το LedgerHedger είναι ένα συμβόλαιο μεταξύ δύο χρηστών, ο ένας εκ των οποίων θέλει να εισαγάγει μία συναλλαγή σε μπλοκ σε κάποια μελλοντική στιγμή και ο άλλος μπορεί να διαθέσει χώρο σε μπλοκ την ίδια μελλοντική στιγμή. Μέσω ενός έξυπνου συμβολαίου ο πρώτος χρήστης προαγοράζει τον χώρο από τον δεύτερο.

0.2 Cardano

Στην παρούσα εργασία ασχολούμαστε με συστήματα blockchain με μηχανισμό σταθερής χρέωσης. Το σύστημα στο οποίο επικεντρωνόμαστε ειδικότερα είναι το Cardano, ένα σύστημα τρίτης γενιάς που χρησιμοποιεί αλγόριθμο Proof of Stake και χρησιμοποιεί ως νόμισμα το Ada την υποδιαίρεσή του Lovelace.

0.2.1 Blockchain Τρίτης Γενιάς

Η πρώτη γενιά blockchain εισήγαγε την έννοια των κρυπτονομισμάτων με το Bitcoin να είναι το πρώτο και πιο χαρακτηριστικό παράδειγμα. Στη συνέχεια, συστήματα δεύτερης γενιάς, όπως το Ethereum, επέτρεψαν την ύπαρξη έξυπνων συμβολαίων και τη χρήση του blockchain για εκτέλεση κώδικα. Τέλος, τα συστήματα τρίτης γενιάς, όπως το Cardano, προσπαθούν να φέρουν λύσεις σε τρεις βασικές προκλήσεις στον χώρο των blockchain.

- **Επεκτασιμότητα (Scalability)**: Εκφράζει τη δυνατότητα του συστήματος να εξυπηρετεί έναν αυξανόμενο αριθμό χρηστών.
- **Διασυνδεσιμότητα (Interoperability)**: Η δυνατότητα διασυνδεσιμότητας του συστήματος με άλλα συστήματα blockchain και με συμβατικά οικονομικά συστήματα.
- **Βιωσιμότητα(Sustainability)**: Η δυνατότητα του συστήματος να συντηρείται και να εξελίσσεται με την πάροδο του χρόνου.

0.2.2 Εμπιστοσύνη στο Cardano

Μία από τις βασικές προτεραιότητες του Cardano είναι η αξιοπιστία του συστήματος. Αυτό επιτυγχάνεται αρχικά με την επιστημονική τεκμηρίωση των δομικών του στοιχείων μέσω αξιολογημένων δημοσιεύσεων. Έπειτα, για τα έξυπνα συμβόλαια του Cardano χρησιμοποιείται η Plutus, μία συναρτησιακή γλώσσα προγραμματισμού που περιορίζει τα σφάλματα κατά την εκτέλεση του κώδικα και σε συνδυασμό με το EUTxO μοντέλο και το μηχανισμό σταθερών χρεώσεων δημιουργεί ένα περιβάλλον συνέπειας στη λειτουργία του συστήματος.

0.2.3 Πλάνο Ανάπτυξης του Cardano

Ακόμα μία καινοτομία του Cardano είναι οι σταδιακή ανάπτυξή του σε 5 φάσεις. Κάθε νέα φάση είσηγαγε καινοτομίες μονάχα αφού σταθεροποιούταν η προηγούμενη. Οι φάσεις ήταν οι ακόλουθες:

- **Byron**: Δημιούργησε το δίκτυο του blockchain και εκκίνησε τον αλγόριθμο συναίνεσης Ouroboros.
- **Shelley**: Επέτρεψε την αποκέντρωση του δικτύου χρησιμοποιώντας τον αλγόριθμο Ouroboros Praos δίνοντας τη δυνατότητα παραγωγής μπλοκ στους συμμετέχοντες (stakeholders).
- **Goguen**: Εισήγαγε τη λειτουργικότητα των έξυπνων συμβολαίων.
- **Basho**: Εισήγαγε λετουργικότητες στοχευμένες στην εξυπηρέτηση μεγαλύτερου αριθμού χρηστών.
- **Voltaire**: Μετέφερε την πλήρη κυβερνησιμότητα του συστήματος στους συμμετέχοντες.

0.2.4 Αλγόριθμος Ουροβόρος

Το Ouroboros είναι ο πρώτος PoS αλγόριθμος επιστημονικά αποδεδειγμένα ασφαλής. Είναι ο αλγόριθμος που χρησιμοποιεί το Cardano, πιο συγκεκριμένα χρησιμοποιείται το μοντέλο Ouroboros Praos. Στον αλγόριθμο αυτό ο χρόνος χωρίζεται σε εποχές (epochs) και θέσεις (slots). Στην αρχή κάθε εποχής οι θέσεις ανατίθεται στους συμμετέχοντες μέσω μίας τυχαίας διαδικασίας με βάση το οικονομικό μερίδιο που διαθέτουν. Στη συνέχεια οι συμμετέχοντες συλλέγουν κατατεθημένες συναλλαγές και τις εισάγουν σε ένα νέο μπλοκ που δημιουργούν στη θέση που τους ανατέθηκε. Έπειτα το μπλοκ συνδέεται με την υπάρχουσα αλυσίδα επεκτίνοντάς τη και το δίκτυο ενημερώνεται για την αλλαγή.

0.2.5 Αμοιβές στο Cardano

Οι δημιουργοί των μπλοκ λαμβάνουν μία αμοιβή για τη δουλειά τους. Στο Cardano αυτή η αμοιβή κατανέμεται με το τέλος της εποχής στους συμμετέχοντες ανάλογα με τα μπλοκ που παρήγαγαν και πηγάζει από δύο προελεύσεις. Ένα αρχικό απόθεμα το οποίο απελευθερώνει σταδιακά κεφάλαιο στο δίκτυο και τις χρεώσεις των συναλλαγών που εισήχθησαν σε μπλοκ κατά τη διάρκεια της εποχής.

0.2.6 Μοντέλο EUTxO

Το λογιστικό μοντέλο του Cardano λέγεται EUTxO και είναι μία επέκταση του UTxO που χρησιμοποιεί το Bitcoin, ούτως ώστε να μπορεί να υποστηρίξει την ύπαρξη έξυπνων συμβολαίων. Για το σκοπό αυτό τα UTxO που είναι στην ουσία το νόμισμα του μοντέλου εφοδιάζονται με πιο μία δομή δεδομένων που ονομάζεται datum. Ενώ στο UTxO αυτή η δομή δεδομένων αποθηκεύει μονάχα τον δικαιούχο του νομίσματος, στο EUTxO επεκτίνεται ώστε να μπορεί να αποθηκεύσει προγραμματιστική λογική, υπεύθυνη για τη δημιουργία πιο σύνθετων συναλλαγώ όπως θα δούμε στη συνέχεια.

0.2.7 Συναλλαγές στο Cardano

Το μοντέλο EUTxO φέρνει δύο καινοτομίες στις συναλλαγές: α) Το script context, στα ελληνικά πλαίσιο συναλλαγής, είναι δεδομένα σχετικά με τη συναλλαγή τα οποία δίνουν τη δυνατότητα εκτέλεσης σύνθετης λογικής από έξυπνα συμβόλαια και β) Ορίζει ένα χρονικό διάστημα στο οποίο η συναλλαγή δύναται να εισαχθεί σε μπλοκ, όταν γίνεται προσπάθεια εισαγωγής μίας συναλλαγής εκτός του χρονικού της πλαισίου, η προσπάθεια ακυρώνεται. Οι συναλλαγές σε αυτό το μοντέλο αποτελούνται από τα ακόλουθα χαρακτηριστικά

- **Ταυτότητα Συναλλαγής**: Ένα διακριτικό για τη συναλλαγή
- **Εισροές (Inputs)**: Τα νομίσματα (UTxO) που καταναλώνει μία συναλλαγή.
- **Προϊόντα (Outputs)**: Τα νομίσματα που δημιουργούνται από τη συναλλαγή.
- **Χρέωση**: Η χρέωση που πληρώνεται για την εισαγωγή μία συναλλαγής σε μπλοκ.

- **Κομμάτια Κώδικα (Scripts)**: Μία συναλλαγή μπορεί να καταναλώνει ή να δημιουργεί ειδικά νομίσματα που φέρουν κώδικα ως αποδεικτικό ιδιοκτησίας.
- **Πλαίσιο Συναλλαγής (Script Context)**: Δεδομένα σχετικά με τη συναλλαγή στα οποία έχει πρόσβαση ο κώδικας που μπορεί να χρησιμοποιείται από τη συναλλαγή.
- **Δεδομένα Εξαργύρωσης (Redeemer)**: Δεδομένα παρεχόμενα από τον δημιουργό της συναλλαγής που χρησιμοποιούνται από τον κώδικα ώστε να ξεκλειδώσουν κεφάλαια.

0.2.8 Έξυπνα Συμβόλαια

Τα συστήματα blockchain μπορούν να χρησιμοποιηθούν, πέρα από απλές συναλλαγές μεταξύ χρηστών και για την εκτέλεση κώδικα. Αυτό υλοποιείται μέσω των έξυπνων συμβολαίων (smart contracts). Σε αυτήν την εργασία επικεντρωνόμαστε στα έξυπνα συμβόλαια όπως αυτά υλοποιούνται στο μοντέλο EUTxO του Cardano.

Ένα έξυπνο συμβόλαιο είναι μία διεύθυνση στην οποία αποθηκεύονται κεφάλαια κλειδωμένα από μία παραμετροποιήσιμη λογική. Νομίσματα (UTxO) δύνανται να χρησιμοποιούν την διεύθυνση αυτή αντί της διεύθυνσης ενός χρήστη. Επιπλέον κάθε νόμισμα που αναφέρει τη διεύθυνση χρησιμοποιεί δεδομένα αποθηκευμένα στο datum για να παραμετροποιήσει τη λογική του συμβολαίου ώστε να ταιριάζει στις ανάγκες του συγκεκριμένου χρήστη που το δημιουργεί.

Ένας χρήστης που θέλει να ξεκλειδώσει τα αποθηκευμένα κεφάλαια από τη διεύθυνση, πρέπει να χρησιμοποιήσει το νόμισμα σε μία συναλλαγή στην οποία θα παρέχει και τα δεδομένα εξαργύρωσης (redeemer) τα οποία θα χρησιμοποιηθούν από τον κώδικά που περιγράφεται στη διεύθυνση για να ξεκλειδωθούν τα κεφάλαια.

Στη διαδικασία ξεκλειδώματος χρησιμοποιείται επίσης το πλαίσιο συναλλαγής (script context) το οποίο περιέχει δεδομένα σχετιά με τη συναλλαγή στην οποία γίνεται η απόπειρα ξεκλειδώματος.

0.2.9 Χρεώσεις

Το Cardano χρησιμοποιεί μοντέλο σταθερής χρέωσης για τις συναλλαγές του. Η χρέωση κάθε συναλλαγής υπολογίζεται πάντα με τον ίδιο τρόπο και βασίζεται στο μέγεθος της συναλλαγής και στους πόρους που καταναλώνει αν χρησιμοποιεί έξυπνα συμβόλαια. Οι πόροι αυτοί χωρίζονται σε δύο κατηγορίες: α) την επεξεργαστική ισχύ που είναι απαραίτητη για την εκτέλεση του κώδικα που υλοποιεί τη λογική του συμβολαίου και β) τον αποθηκευτικό χώρο που χρησιμοποιείται για την εκτέλεση αυτή.

0.3 Θεωρία Παιγνίων

Θεωρία παιγνίων ονομάζεται ο κλάδος των μαθηματικών που μελετάει τις αλληλεπιδράσεις μεταξύ πλευρών - παικτών, οι οποίες έχουν στόχο τη μεγιστοποίηση της ωφέλειάς τους.

Οι αλληλεπιδράσεις μοντελοποιούνται ως παίγνια με αυστηρούς μαθηματικούς ορισμούς. Για την απόδοση όρων στα ελληνικά συμβουλευόμαστε το [73], όπου αυτό είναι δυνατό.

0.3.1 Συνάρτηση Ωφέλειας

Κάθε παίκτης έχει διαφορετικές επιθυμίες για την κατάληξη του παιγνίου και κάθε κατάληξη του δίνει διαφορετική ικανοποίηση. Ονομάζουμε την ποσοτικοποίηση την ικανοποίησης αυτής ωφέλεια του παίκτη και συνάρτηση ωφέλειας, τη συνάρτηση που αντιστοιχίζει καταλήξεις ενός παιχνιδιού στην ωφέλεια του παίκτη.

0.3.2 Παίγνια Κανονικής Μορφής

Η βασικότερη μέθοδος αναπαράστασης ενός παιγνίου είναι μέσω της κανονικής μορφής. Τα παίγνια κανονικής μορφής αναπαρίστανται σε έναν πίνακα N διαστάσεων, όπου N είναι ο αριθμός των παικτών. Κάθε διάσταση αναπαριστά τις διαθέσιμες κινήσεις κάθε παίκτη και τα κελιά περιέχουν την ωφέλεια που λαμβάνουν οι παίκτες αν ακολουθηθούν κινήσεις που αντιστοιχούν στο συγκεκριμένο κελί.

0.3.3 Στρατηγικές και Ισορροπίες

Σε κάθε παίγνιο οι παίκτες έχουν ένα σύνολο διαθέσιμων κινήσεων. Ονομάζουμε την επιλεγμένη κίνηση ενός παίκτη στρατηγική του και το σύνολο των επιλεγμένων κινήσεων στρατηγικό προφίλ. Αν ο παίκτης αποφασίσει να παίξει μία συγκεκριμένη κίνηση τότε λέμε ότι έχει μία καθαρή στρατηγική, ωστόσο, μπορεί να αποφασίσει να παίξει μία κίνηση τυχαία, αναθέτοντας σε κάθε κίνηση μία πιθανότητα να παιχτεί. Το σύνολο των πιθανοτήτων που έχουν ανατεθεί σε κάθε δυνατή κίνηση αποτελεί μία μικτή στρατηγική του παίκτη.

Στη μελέτη των παιγνίων μας νοιάζουν ιδιαίτερα τα στρατηγικά προφίλ που φέρνουν το παίγνιο σε ισορροπία. Η βασικότερη έννοια ισορροπίας της θεωρίας παιγνίων είναι η Ισορροπία Nash. Ένα στρατηγικό προφίλ ενός παιγνίου αποτελεί ισορροπία Nash αν κανένας από τους παίκτες δεν μπορεί να βελτιώσει την ωφέλειά του αλλάζοντας μονομερώς τη στρατηγική του, δηλαδή δεδομένου ότι οι υπόλοιποι παίκτες θα κρατήσουν τις στρατηγικές τους σταθερές.

0.3.4 Παίγνια Τέλειας Πληροφόρησης σε Αναλυτική Μορφή

Τα παίγνια κανονικής μορφής εκφράζουν τις κινήσεις των παικτών χωρίς να λαμβάνουν υπόψιν τον χρονικό παράγοντα, θεωρώντας πως όλες συμβαίνουν ταυτόχρονα. Για αλληλεπιδράσεις στις οποίες οι συμμετέχοντες δρουν διαδοχικά, χρησιμοποιούμε τα παίγνια αναλυτικής μορφής. Τα παίγνια αυτά αναπαρίστανται με ένα δέντρο στο οποίο οι καταστάσεις του παιγνίου παρατίθενται ως κόμβοι και οι διαθέσιμες κινήσεις των παικτών ως ακμές του δέντρου. Κάθε κόμβος αντιστοιχίζεται στον παίκτη ο οποίος πρέπει να επιλέξει κίνηση στην συγκεκριμένη κατάσταση.

Τα παίγνια τέλειας πληροφόρησης με τα οποία θα ασχοληθούμε στη συνέχεια, είναι το υποσύνολο των παιγνίων σε αναλυτική μορφή στα οποία οι παίκτες γνωρίζουν με ακρίβεια όλες τις κινήσεις που έχουν προηγηθεί.

Για την ανάλυση αυτών των παιγνίων η ισορροπία Nash δεν επαρκεί. Οπότε, χρησιμοποιείται μία διαφορετική, αυστηρότερη έννοια ισορροπίας, η Subgame Perfect Equilibrium (SPE), που αποδίδουμε στα ελληνικά ως Τέλεια Ισορροπία Υποπαιγνίου. Αναπαριστώντας το παίγνιο με ένα δέντρο, ονομάζουμε υποπαίγνιο το παίγνιο που αναπαρίσταται από ένα υποδέντρο του αρχικού δέντρου, έχει δηλαδή ως ρίζα έναν κόμβο του αρχικού δέντρου και αποτελείται από όλους τους απογόνους αυτού. Τότε ένα στρατηγικό προφίλ αποτελεί SPE όταν το προφίλ αυτό αποτελεί ισορροπία Nash για κάθε υποπαίγνιο του αρχικού παιγνίου.

Για να βρούμε την SPE χρησιμοποιούμε τον αλγόριθμο οπισθογενούς επαγωγής (backward induction). Ο αλγόριθμος λειτουργεί με διάσχιση του δέντρου κατά βάθος. Ξεκινώντας από τα φύλλα, σε κάθε κόμβο επιλέγεται η ακμή η οποία οδηγεί τον αντίστοιχο παίκτη σε μεγαλύτερη ωφέλεια, στη συνέχεια ο κόμβος ενημερώνεται με την ωφέλεια αυτή.

0.3.5 Προσθέτοντας Τυχαίες Κινήσεις

Πολλές φορές μία αλληλεπίδραση δεν εξαρτάται μόνο από τις κινήσεις των παικτών, αλλά και από εξωτερικές συνθήκες. Οι αλληλεπιδράσεις αυτές μοντελοποιούνται με παίγνια που περιέχουν τυχαίες κινήσεις. Θεωρούμε έναν παίκτη με μηδενική συνάρτηση ωφέλειας ο οποίος αναπαριστά τις εξωτερικές συνθήκες. Η στρατηγική του είναι μικτή και αναθέτει σε κάθε πιθανή κίνηση την αντίστοιχη πιθανότητα για τα αποτελέσματα των εξωτερικών συνθηκών.

Για την ανάλυση αυτών των παιγνίων μετατρέπουμε ελαφρώς τον αλγόριθμο οπισθογενούς επαγωγής και στους κόμβους στους οποίους επιλέγεται η τυχαία κίνηση αντιστοιχίζεται η αναμενόμενη ωφέλεια με βάση τις πιθανότητες της κάθε κίνησης.

0.4 Παράγωγα στο χώρο συναλλαγών του μπλοκ

0.4.1 Μοντέλο

Σε αυτήν την εργασία συγκρίνουμε τα συστήματα δυναμικής και σταθερής χρέωσης σχετικά με τις εγγυήσεις που δύνανται να παρέχουν στους χρήστες, όσον αφορά την εξυπηρέτηση. Οι χρήστες υποβάλλουν συναλλαγές για εισαγωγή σε μπλοκ και οι παραγωγοί τις συγκεντρώνουν και δημιουργούν τα μπλοκ. Οι χρήστες μπορούν επίσης να καταθέσουν συναλλαγές που περιλαμβάνουν έξυπνα συμβόλαια, όπως αυτά εξηγήθηκαν νωρίτερα.

Για την εισαγωγή των συναλλαγών τους σε μπλοκ, οι χρήστες πληρώνουν χρεώσεις που στα συστήματα δυναμικής χρέωσης υπολογίζονται με μεταβαλλόμενο τρόπο σε κάθε μπλοκ, ενώ με σταθερό τρόπο σε εκείνα σταθερής χρέωσης. Όταν η κίνηση είναι αυξημένη, τα συστήματα δυναμικής χρέωσης ανεβάζουν τις τιμές, ενώ σε αυτά με σταθερή χρέωση η κίνηση μεταφράζεται σε καθυστερήσεις στην εισαγωγή των συναλλαγών.

Θεωρούμε έναν χρήστη, ο οποίος θέλει να υποβάλλει μία συναλλαγή κάποια στιγμή στο μέλλον. Ωστόσο, αυτή η συναλλαγή του αποφέρει ωφέλεια μονάχα αν εισαχθεί σε μπλοκ εντός ενός χρονικού διαστήματος, ενώ με το πέρας του διαστήματος δέχεται μία πιθανώς αρνητική ωφέλεια, είτε η συναλλαγή εισαχθεί σε μπλοκ, είτε όχι.

Καθώς η συναλλαγή θα υποβληθεί μελλοντικά, δεν υπάρχει τρόπος να προβλεφθεί η κίνηση που θα παρουσιάζει εκείνη τη στιγμή το δίκτυο, οπότε κάνουμε τις ακόλουθες δύο υποθέσεις. α) Στα συστήματα σταθερής χρέωσης, δεν υπάρχει ασφαλής τρόπος πρόβλεψης της καθυστέρησης στην εισαγωγή μιας συναλλαγής σε εκείνη την μελλοντική στιγμή. β) Στα συστήματα δυναμικής χρέωσης η τιμή μπορεί να φτάσει μέχρι ένα εξαιρετικά υψηλό όριο και δεν υπάρχει τρόπος πρόβλεψης της τιμής πέρα από αυτό το όριο. Ως αποτέλεσμα, και στις δύο περιπτώσεις, ο χρήστης δεν μπορεί να γνωρίζει εκ των προτέρων αν θα μπορέσει να λάβει την επιθυμητή ωφέλεια ή θα καταλήξη με πιθανώς αρνητική.

Μελετάμε πιθανές συμφωνίες του χρήστη με έναν ασφαλιστή, δυνητικά παραγωγό μπλοκ, ο οποίος θα εξασφαλίσει την εμπρόθεσμη εισαγωγή της συναλλαγής. Ορίζουμε την έννοια της ϵ-προβλέψιμης εξυπηρέτησης ως μία συμφωνία που: α) είναι αμοιβαία προσοδοφόρα και β) εγγυάται ένα κάτω όριο στην αναμενόμενη ωφέλεια του χρήστη.

0.4.2 Σύγκριση Μοντέλων Δυναμικών και Σταθερών Χρεώσεων

Το [69] περιγράφει ένα πρωτόκολλο αντίστοιχο με αυτό που περιγράφουμε για ένα σύστημα δυναμικής χρέωσης, ωστόσο δεν καταφέρνει να πετύχει ϵ-προβλέψιμη εξυπηρέτηση για ϵ καλύτερο από ένα όριο. Οι παραγωγοί μπλοκ δεν μπορούν να προφυλάξουν μία συναλλαγή από την απρόβλεπτη άνοδο της τιμής χρέωσης πέρα από το όριο.

Πράγματι, αναλύοντας την κοινή ωφέλεια χρήστη και ασφαλιστή σε ένα τέτοιο πρωτόκολλο, βλέπουμε ότι δεν μπορεί να είναι εγγυημένα θετική. Ως επακόλουθο, οποιαδήποτε προσπάθεια είτε δεν θα είναι αμοιβαία προσοδοφόρα, είτε δεν θα προσφέρει ένα εγγυημένο θετικό κάτω όριο.

0.4.3 Πρωτόκολλο

Παρουσιάζουμε το συμβόλαιό μας για ϵ-προβλέψιμη εξυπηρέτηση σε συστήματα σταθερής χρέωσης. Το πρωτόκολλο αποτελείται από δύο φάσεις, στην πρώτη ο χρήστης και ο ασφαλιστής δημιουργούν ένα έξυπνο συμβόλαιο που υλοποιεί το πρωτόκολλο και στη δεύτερη, που εκτυλίσσεται κατά το μελλοντικό χρονικό διάστημα, η λογική του συμβολαίου εκτελείται και η συναλλαγή εισάγεται στο μπλοκ.

Η πρώτη φάση εκκινείται από τον χρήστη ο οποίος δημιουργεί το συμβόλαιο, καταθέτει τα απαραίτητα κεφάλαια και την πληρωμή του ασφαλιστή και του το στέλνει. Στη συνέχεια, ο ασφαλιστής το συμπληρώνει καταθέτοντας ένα ενέχυρο και το συμβόλαιο εισάγεται στο blockchain.

Η δεύτερη φάση περιγράφει τρία πιθανά σενάρια:

- Οι δύο πλευρές ακολουθούν το πρωτόκολλο με επιτυχία, η συναλλαγή εισάγεται εγκαίρως στο μπλοκ και ο ασφαλιστής λαμβάνει την αμοιβή του.

- Ο χρήστης δεν στέλνει ποτέ τη συναλλαγή στον ασφαλιστή και ο ασφαλιστής σπαταλάει τον ασφαλισμένο χώρο του μπλοκ λαμβάνοντας μέρος της αμοιβής
- Ο ασφαλιστής εγκαταλείπει το συμβόλαιο και αφήνει το επιθυμητό χρονικό διάστημα να περάσει. Σε αυτήν την περίπτωση ο χρήστης μπορεί να λάβει πίσω τα κεφάλαια που αποθήκευσε στο συμβόλαιο.

0.5 Παιγνιοθεωρητική ανάλυση του πρωτοκόλλου

0.5.1 Παίγνιο στην απλή περίπτωση

Το παίγνιο που περιγράφει το πρωτόκολλό μας είναι αναλυτικής μορφής, τέλειας πληροφόρησης, με τυχαίες κινήσεις. Ο χρήστης και ο ασφαλιστής επιλέγουν κινήσεις εναλλάξ, γνωρίζοντας όλες τις κινήσεις που επιλέχθηκαν μέχρι εκείνη τη στιγμή. Ανάμεσα στις δύο φάσεις του πρωτοκόλλου προκύπτει η καθυστέρηση του δικτύου, αυτή η εξωτερική συνθήκη επηρεάζει το παίγνιο και παρουσιάζεται ως τυχαία κίνηση. Υποθέτουμε αρχικά ότι ο ασφαλιστής παράγει με βεβαιότητα μπλοκ στο μελλοντικό επιθυμητό χρονικό διάστημα.

Θέλουμε να δείξουμε ότι το πρωτόκολλό μας πετυχαίνει ϵ-προβλέψιμη εξυπηρέτηση για κάποιο ϵ. Για να το πετύχουμε χρησιμοποιούμε τον αλγόριθμο οπισθογενούς επαγωγής και αποδεικνύουμε ότι το στρατηγικό προφίλ, που αποτελείται από τις κινήσεις όπως περιγράφονται στο πρωτόκολλο, αποτελεί SPE. Χωρίζουμε το παίγνιο σε μικρότερα υποπαίγνια βρίσκοντας σταδιακά την SPE και συνθέτοντάς τα στη συνέχεια μέχρι να καταλήξουμε στην SPE ολόκληρου του παιγνίου.

0.5.2 Εκτεταμένο Παίγνιο

Στη συνέχεια αναλύουμε ένα γενικευμένο παίγνιο που μοντελοποιεί το μοντέλο μας όταν ο ασφαλιστής δεν παράγει απαραίτητα μπλοκ στο επιθυμητό διάστημα. Αντ'αυτού, είναι παραγωγός μπλοκ με πιθανότητα που εξαρτάται από το οικονομικό του μερίδιο και η οποία είναι εκ των προτέρων γνωστή σε όλους. Τελικά, ο ασφαλιστής μαθαίνει αν παράγει μπλοκ ανάμεσα στις δύο φάσεις, ωστόσο, η πληροφορία δεν φτάνει στον χρήστη. Η μοντελοποίηση αυτής της συνθήκης γίνεται με μία ακόμα τυχαία κίνηση.

Αποδεικνύουμε εκ νέου ότι το πρωτόκολλο πετυχαίνει ϵ-προβλέψιμη και σε αυτήν την περίπτωση, αυτή τη φορά δεδομένης μίας συνθήκης για την πιθανότητα με την οποία παράγει μπλοκ ο ασφαλιστής. Χρησιμοποιούμε ξανά τον αλγόριθμο οπισθογενούς επαγωγής, αυτή τη φορά στο νέο παιχνίδι.

0.5.3 Χωρίς Συμβόλαιο

Τέλος, υπολογίζουμε την αναμενόμενη ωφέλεια όταν δεν δημιουργείται συμβόλαιο και ο χρήστης ακολουθεί τη συνηθισμένη διαδικασία υποβολής της συναλλαγής του, με ρίσκο να προκύψει μεγάλη καθυστέρηση και η συναλλαγή να μην μπει εγκαίρως. Συγκρίνουμε την ωφέλεια αυτή με την εγγυημένη του συμβολαίου και καταλήγουμε κάτω υπό ποιες συνθήκες ο χρήστης έχει συμφέρον να δημιουργήσει το συμβόλαιο και ο ασφαλιστής να το ολοκληρώσει.

0.6 Συμπεράσματα και Μελλοντική Εργασία

Σε αυτήν την εργασία μελετήσαμε το πρόβλημα της αβεβαιότητας της εξυπηρέτησης σε περιβάλλοντα blockchain. Είδαμε μηχανισμούς χρέωσης που έχουν ως στόχο τον έλεγχο των συναλλαγών που εισάγονται σε μπλοκ και την αξιολόγησή τους από καταξιωμένους αναλυτές. Παρουσιάσαμε εργασίες στον τομέα των παραγώγων συνδεδεμένων με τον χώρο σε μπλοκ και εισήγαμε έναν ορισμό προβλεψιμότητας που αξιολογεί τέτοια πρωτόκολλα. Δείξαμε ότι τα συστήματα δυναμικής χρέωσης δεν υποστηρίζουν πρωτόκολλα που ικανοποιούν τον ορισμό αυτό. Τέλος, περιγράψαμε ένα πρωτόκολλο για συστήματα σταθερής χρέωσης και δείξαμε ότι πληροί τον ορισμό αξιολόγησης που δώσαμε.

Μελλοντικές επεκτάσεις της παρούσας εγασίας, είναι η υλοποίηση του πρωτοκόλλου μας για ένα υπαρκτό σύστημα όπως το Cardano, η επέκτάση του ώστε να περιλαμβάνει την εξασφάλιση περισσοτέρων συναλλαγών και η ανάλυση του πρωτοκόλλου σε ένα σύστημα με πολλούς συμμετέχοντες και αντικρουόμενα συμβόλαια.

Chapter 1

Introduction

1.1 Background

Over the past years, blockchain systems have been growing in popularity. Their decentralized and secure nature provides novel solutions to a wide range of challenges across many different fields. More and more decentralized apps are being built upon the technology, creating an increasing demand. However, blockchains still suffer from serious scalability issues.

One of the main scalability challenges comes with the limited supply of block space, conflicting with the high demand. As a result, blockchains suffer from periods of congestion where the service provided is not guaranteed and users have to deal with elevated prices or long delays. The congested periods occur unexpectedly, making the system unpredictable and thus unreliable.

Various alternatives have been proposed to provide a solution to this service predictability, aiming for a more user-friendly ecosystem. Nevertheless, the problem is still open and new proposals are introduced, each with a different focus.

1.2 Motivation

Although there has been extensive work on addressing service unpredictability using Transaction Fee Mechanisms (TFMs), the use of block space derivatives is almost non-existent. Block space derivatives are agreements between users of the system using the space allocation of a block as their underlying value. While TFMs target protocol-level functions of the system to provide a solution to the problem, derivatives are built on top of an existing protocol without altering the underlying mechanisms. In this work, we introduce a novel protocol describing a derivative between a user and a block producer securing a future transaction.

Furthermore, the limited research regarding block space derivatives is mostly focused on dynamic fee mechanisms, works like [50, 1, 69] are built on top Ethereum-like protocols. In [37], a token model is introduced that makes possible the presale of block space; however, even though the protocol could be implemented to work in parallel to an existing fixed-fee system, it requires major modifications to the underlying system. Our protocol is designed to be implemented on top of a fixed-fee model as a smart contract and requires no changes to the system.

Moreover, to our knowledge, there exists no work comparing dynamic and fixed-fees models regarding the guarantees achieved when a block space derivative protocol is built on top of them. We argue that protocols built on fixed-fees models can achieve stronger assurances compared to dynamic fees and use joint utility analysis to back our claim.

1.3 Literature Review

Over the years, extensive work has been done to improve the predictability and quality of service of blockchain systems. Different TFMs have been introduced, as well as improvements and modifications to existing ones. A notable debate happened around FPA mechanism, when the implementation of larger blocks was proposed in Bitcoin. The proposal aimed to raise blockspace demand, limiting the price surge of the fees. The dispute escalated, leading to a major split, resulting in a new chain known as Bitcoin Cash [8]. Houy[31] and Rizun[53] formally model a fee equilibrium derived by the market's supply and demand. Lavi et al.[41] and Yao[71], review a mechanism aimed at raising the miners' revenue as their reward per block declines. In the mechanism, the fee is equal to the lowest bid of an included transaction; miners then raise this fee by publishing underfull blocks only containing high bids. Focusing on incentive compatibility and preventing miner manipulation, Basu et al.[5] propose a mechanism that charges lower the transactions included in underful blocks and distributes fees to the miners of future blocks. Kiayias et al.[36] introduce a novel tiered mechanism intended for accommodating transactions of different urgency. In the work, transactions are charged differently depending on the service and their required service priority. Ethereum's EIP-1559 dynamic mechanism has also been rigorously analyzed in terms of stability and incentive compatibility, including the works of Liu et al.[43], Reijsbergen et al.[52], Leonardos et al.[42] and Zhang and Zhang[72], while different dynamic fee methods are reviewed in[22] of Crapis, Moallemi, and Wang, [28] of Ferreira et al., and Ndiaye's[49]. Roughgarden [54] creates a game-theoretic TFM incentive review model. Chung and Shi [20] build upon this model providing a notion of imposibility and a different TFM to circumvent this impossibility. Gafni and Yaish[29] and Tang and Yao[68] continue the work of Chung and Shi[20]. In [19], Roughgarden joins forces with Chung and Shi and they provide a new impossibility notion. Complementing these approaches, AQQUA [51] explores the design of decentralized payment systems that achieve both user privacy and auditability, enabling selective disclosure and policy compliance without sacrificing anonymity or incentive compatibility.

Another line of work is pursuing service predictability in blockchain systems using gas tokens that the users can obtain in advance. Those works are often based on estimations of future network congestion, an issue tackled by Lotem et al.[44]. In [9, 47, 1], existing Ethereum functionalities are extended to achieve this. However, the mechanism was deemed unusable with Ethereum more recent updates [27, 15, 7]. Similar mechanisms are [37] by Kiayias et al., where they introduce a new token not based on ethereum's system, gas token uGAS [25] and derivative protocols [69, 50] that aim to adapt traditional derivative methods [33] to the blockchain environment.

1.4 Thesis Outline

1.4.1 Chapter 1: Introduction

In the first chapter, we give a brief introduction to the service unpredictability problem. We present the motivation that led to this work and provide an overview of the related work done so far regarding this blockchain issue. The chapter is concluded with an outline1.4 of our work.

1.4.2 Chapter 2: Blockchain Systems

In the second chapter, the core blockchain principles are explained. We present the concept of accounting models, the method used by blockchain systems to keep the ledger state, and we refer to the UTxO and account model, used by the two most popular systems, Bitcoin and Ethereum, as well as the EUTxO, used by Cardano, the system we focus on in this work.

Next, we explain the concept of a consensus algorithm, the method used by a decentralized network of nodes to agree upon a single state. We briefly review the difference between Proof of Work, the first blockchain consensus algorithm used by Bitcoin, and Proof of Stake, a more recent alternative to Proof of Work, used by Ethereum and Cardano.

Furthermore, we describe the blockchain trilemma, the three pillars blockchain systems aim to combine, Decentralization, Security and Scalability. However, the general consensus is that no two of the three pillars can be successfully accommodated without significant compromise on the third.

Our work mainly delves into the third pillar, scalability. More specifically, we focus on the limited block space available for transactions and the ways this space is allocated to the - often excessive amount of - user transactions. The first line of work we analyze are transaction fee mechanisms (TFMs), the algorithms used to charge users submitting transactions as a way to bridge supply and demand.

The most popular TFMs are the First Price Auction, used by Bitcoin, the dynamic-fees, used by Ethereum (EIP-1559), and fixed-fees, used by Cardano. Each with different advantages and disadvantages, they have been rigorously analyzed in various works. Most notably, we refer to the works of Roughgarden, Chung and Shi [54, 20, 19] that use game-theoretic notions to analyze the incentive compatibility of TFMs.

The chapter is concluded with a review of three different solutions to the limited block space issue [37, 50, 69] by introducing derivatives on the blockchain ecosystem.

1.4.3 Chapter 3: Cardano

The third chapter revolves around Cardano, the system that will be the focus of our work. We provide a synopsis of the system's core components and explain its foundational goals and operational principles.

Cardano is built around three pillars: Scalability, Interoperability and Sustainability. Scalability expresses the ability of the system to handle a growing number of users. Interoperability reflects the protocol's vision of achieving compatibility with different systems. Sustainability addresses the governance of the system as time passes.

Another core objective of Cardano is the ideal of reliability. Cardano's theoretical foundations are peer-reviewed works and its script language follows the functional programming paradigm. Using this approach, Cardano achieves stability guarantees for its system that make it a desirable platform for users.

The development of the system was completed in 5 different stages, introducing its novelties gradually and making sure each one is stable before moving to the next. This way the process introduced in turn the network, decentralized staking, smart contracts, scaling solutions and last decentralized governance.

We proceed by reviewing the system's underlying mechanisms. Its consensus algorithm is the Ouroboros protocol, the first Proof of Stake protocol to be proven secure. It functions by splitting time into epochs and then slots and holding a lottery to assign slots to stakeholders who will be responsible for creating a block.

Last, we explain the methods used by Cardano to give out rewards, keep the ledger state and handle transactions and smart contracts. In doing so, we aim to provide an interested reader with an understandable guide for the system's main components.

1.4.4 Chapter 4: Game Theory

In the fourth chapter, we provide the reader with the game theory knowledge necessary to follow the subsequent analysis of our protocol. We start with explaining the basic notions of a game, as a formal model of real-world scenarios, utility, as the metric for a player's satisfaction, strategies, as the courses of action chosen by the players, and equilibrium, as the state of a game in which the players cannot achieve better utility by changing their course of action.

Furthermore, we explain two game forms: normal forms games, the basic game form in which the players act simultaneously, and the extensive form games, where players take turns selecting their actions one at a time. We show that the two forms require different equilibria notions and present the Subgame Perfect Equilibrium we use in the main results of this work.

Last, we describe the concept of a chance move. An action selected randomly, representing the natural conditions of an extensive form game, conditions independent of the rational players of the game.

1.4.5 Chapter 5: Block Space Derivatives

In the fifth chapter, our main work is introduced. The chapter is split into three sections; we first describe the setting in which our analysis is made. Both a dynamic and a fixed model are defined, their differences are highlighted and the required assumptions are made. We then introduce our problem, presenting the two parties who participate in the protocol along with their respective objectives and utility functions.

In the second section, we argue that a predictable service protocol with the desirable characteristics, such as the one we introduce under a fixed-fees model, cannot be achieved in a dynamic-fees one. We back our claim by analyzing the joint utilities of the two parties and comparing the guarantees that can be attained under each model.

In the final section, our protocol is introduced. A smart contract between a User requiring the inclusion of a transaction at a future moment and an Insurer who will produce a block at this future moment and is willing to ensure the transaction's inclusion in return for some payment. We present the creation process of the smart contract implementing the protocol and the intended actions of the players. We further detail the functions available for the parties to interact with the smart contract.

1.4.6 Chapter 6: Simple Game

In the sixth chapter, we formally analyze our protocol using game theory. We consider that the Insurer knows at the time the contract is created whether she will be producing a block at the desired future moment or not.

We represent our protocol as an extensive form game and prove that, once a contract is created, there is a subgame perfect equilibrium when the two parties follow the protocol with no deviations; that is, no party can improve its utility by deviating. Furthermore, we identify the assumption under which the two parties maximize their expected utility by creating a contract instead of following the usual transaction inclusion process.

1.4.7 Chapter 7: Extended Game

In the seventh chapter, we consider that the Insurer does not know at the time of the contract creation whether she will be producing a block at the desired time or not. Instead, both parties only have the probability under which the Insurer will produce the block and the actual information will be revealed to the Insurer at some point between the contract creation and the transaction inclusion time.

As a result, our analysis game has to be modified and the results are weakened, as we work using expected utilities. However, we show that our main result remains the same under a suitable assumption. We prove that following the protocol in the new setting is once again a Subgame Perfect Equilibrium, provided the assumption holds. Once more, we additionally identify the assumptions under which both parties maximize their utilities by creating the contract.

1.4.8 Chapter 8: Conclusion

In the final chapter, we conclude our work and provide a summary of our results. Moreover, we point out open topics requiring further research and possible extensions of our work.

Chapter 2

Blockchain

2.1 Blockchain Systems

2.1.1 The technology

Blockchain technology rose in popularity when in 2008 Nakamoto published the Bitcoin white paper[48]. Bitcoin used existing principles to introduce a novel decentralized cryptocurrency. Where traditional currencies depend on a central authority to keep track of balances for users, Bitcoin distributes this responsibility to every node in a network. Balances are kept on a ledger in the form of a series of immutable blocks of information that create an append-only chain, called a blockchain. Blocks contain all the transactions submitted to the system since a starting point stored in the genesis block. Each block points to the previous one via cryptographic means, making it impossible for past blocks to be tampered with without a trace. This chain of blocks leads from the starting point to the current state. Every node in the network keeps a copy of the blockchain, making the system harder to control. New transactions are included regularly in newly created blocks appended to the chain. Each block created is accepted or denied by the network through a consensus algorithm.

2.1.2 Accounting Models

Different blockchain systems have different accounting models to maintain the ledger state. Each with its advantages and disadvantages, those models are used to keep track of transactions and fund exchanges, and the balance distribution can be confirmed by all users at any time.

UTxOs

Bitcoin introduced Unspent Transaction Output (UTxO), an accounting model that resembles the use of cash in the traditional economy. A UTxO is essentially a coin representing a value of the native blockchain token and is owned by a user. UTxO can hold any amount of value and can be spent as an input in a transaction. As the name suggests, UTxOs are transaction outputs; an unspent transaction output can be consumed by a transaction as input, at which point it is deemed spent and no longer usable. The transaction then produces new unspent outputs, UTxOs, which can be spent for new transactions, continuing the process. A UTxO, much like traditional cash coins, cannot be partially spent. Instead, a UTxO holder intending to spend part of the UTxO spends it as a whole, receiving back a new UTxO as change for the exceeding value. The balance of a user is calculated as the sum of the value of all the UTxOs controlled by them.

While the UTxO model is simple and very convenient in the decentralized setting of a blockchain system, it comes with certain shortcomings. A transaction involves the unlocking of output funds from a user using a mechanism called a validator. The validator is the locking mechanism that validates whether a user controls the UTxO or not. In the UTxO model, the validator usually just verifies the public key hash of the user attempting to use the UTxO. However, many blockchain systems aim for more advanced locking methods that make use of complex logic based on information regarding the state of the ledger. The UTxO model cannot accommodate such logic and thus, new accounting models are created to solve this insufficiency.

Account Model

To be able to facilitate more complex transactions, which can run code using its system, Ethereum uses the Account Model to manage the state of its ledger. Where UTxO resembles cash, the Account Model resembles the account system of traditional finance. In this model, each user has an account with a balance. A global state is kept by all nodes in the blockchain, keeping track of all account balances. After every transaction, the accounts affected update their balance and the global state is synchronized in every node across the network.

The Account Model is more complex and harder to maintain than the UTxO; however, it provides much greater freedom in transaction creation. In the Account Model, a transaction can access the complete information of the ledger state and not just the value of its inputs. This information enables the development and execution of code; in other words, what is known as smart contracts.

EUTxO

Cardano aims to find the sweet spot between the two concepts, followed by the two most popular blockchain systems, by introducing Extended UTxO (EUTxO). EUTxO keeps the idea of UTxO but adds some features to it so that it can extend its functionality while keeping its simplicity. This way, a transaction in the EUTxO model can utilize information about the ledger state to execute code, without having to deal with the complexity and challenges of a global state. In the EUTxO model, the validator has extended functionality and can use complex code to validate whether a user controls the UTxO or not.

2.1.3 Consensus Algorithms

Proof of Work (PoW) was introduced by Nakamoto for Bitcoin and it was the first consensus algorithm used for a digital cryptocurrency. It is based on computing power as the decisive resource for the selection of a new block producer. Every node participating in the block creation process is called to solve a cryptographic puzzle; the first to accomplish the task can create a block that will be accepted by everyone in the network. This puzzle-solving process is called mining, taking its name from Bitcoin's parallelism to digital gold. Mining depends on the processing power of a node; the higher it is, the faster the process. Blocks are created at regular intervals and the complexity of the puzzle is adjusted to the power of the contestants to keep this interval roughly stable. This consensus algorithm has been criticized since Bitcoin's creation, mainly due to the vast amounts of energy wasted in this process. As a result, although many cryptocurrencies keep using it, Bitcoin being the most well-known, others switched to other algorithms. Ethereum, the second cryptocurrency in popularity, following Bitcoin, made the change to Proof of Stake (PoS)[14, 26], the second most well-known consensus algorithm. PoS uses staked assets, instead of processing power, as the decisive resource for the block producer. Nodes participating in this block-creating process, called staking, have to lock their funds as collateral. Then, through a randomized process, a participant is selected to produce the next block; typically, the chances of a node being selected depend on the stake locked. Though most blockchain systems use PoW or PoS, there are several other consensus algorithms (Proof Of Burn, Proof Of Space, etc) and others are being researched.

2.1.4 The Blockchain Trilemma

Blockchain systems are built on three basic pillars. Decentralization, Security and Scalability. However, these pillars conflict with each other and no two can be accommodated without the sacrifice of the third. This concept, suggested by Ethereum founder Vitalik Buterin, is commonly known as the blockchain trilemma and is one of the main challenges of the blockchain area of research.

Decentralization

Decentralization is the defining characteristic of blockchain systems and digital cryptocurrencies. Distributing the duty of keeping the ledger to all nodes of the network, instead of having a central authority responsible, is what makes this technology stand out from other currencies. Decentralization shields the system from malicious or faulty authorities who want to take advantage of the network. This way, no agent holds the power to dictate the state of the ledger, controlling the network, and any attempt to modify the records would need the majority of the network to agree on the modified state; this is known as the 51% attack.

Security

Security is an essential characteristic of any system where users trust their funds. Any failure on the security part would induce economic losses for the users and a loss of faith in the system.

Scalability

If blockchain systems aim for widespread acceptance and compete with centralized ones, scalability is of utmost importance. Blockchains need to support a much greater number of users than they currently do, without performance issues. As of now, most blockchain systems can only accommodate a few transactions per second, 12-15 for Ethereum and 6-8 for Bitcoin, while systems like Visa handle more than 50,000. The reason is that blocks have limited capacity and the frequency of their creation is tied to the functionality of each protocol.

When attempting to accommodate one of the pillars, another is taken back. For example, the greater the decentralization, the harder it is for nodes to come to consensus, which lowers the frequency of block creation, hence, Scalability. Trying to keep the frequency high with many nodes would favor decentralization and scalability in expense of a weaker consensus protocol and security.

2.2 Transaction Fee Mechanisms

As we saw, blocks have a limited capacity and often the network suffers from congestion and many transactions are waiting to be included in the ledger. Deciding the priority with which these transactions will be included in a block is an open subject of discussion. One simple way of dealing with this issue is a first-in-first-out (FIFO) queue. However, one might argue that some transactions might provide greater utility than others and thus should be prioritized as they maximize the social welfare of the system.

Another topic for discussion is how we should incentivize the block producers to follow the respective block creation protocol, including the right transactions in the blocks. In many systems, block producers get a reward for the blocks they publish; however, that does not protect from deviations like publishing empty blocks. In other cases, some systems, like bitcoin, have a limited amount of funds to be given as rewards and further in the future, the block producers will have to rely on other sources for their payment.

One more issue we have to take into consideration is DOS (denial of service) attacks, where a user floods the network with a multitude of transactions, making it unavailable for new transactions to be included.

A possible solution for all these issues is a Transaction Fee Mechanism (TFM). A TFM is a protocol for issuing fees that users have to pay for a transaction to be included in the ledger. Multiple such protocols aim at different goals and address different issues. TFM design is the area of research that studies TFMs through Game Theory lens and tries to create the protocol in such a way that users of the system acting for their self-interest will end up following the desired behaviour.

2.2.1 First Price Auction

The simplest TFM, introduced by Bitcoin[48], is the users deciding what they wish to pay for their transaction's inclusion and paying this amount to the block producer to incentivize them to include their transaction in the ledger. This TFM is following a basic auction principle, known in game theory as the First Price Auction (FPA) with multiple winners. In FPA, each participant bids the largest amount they are willing to pay and the largest bids win the auction and pay their bids. In the blockchain context, they get their transaction included and pay a fee equal to their bid to the miner. Block producers have total command over what they include in the blocks they create and they are incentivized to include their highest bidders.

FPAs are the most natural TFMs, but they come with several downfalls. To begin with, those protocols give block producers excessive control. As all fees go directly to the producer of each block, this producer can include their own transactions for free. This means that some transactions favored by the producer can skip the line without actually having to pay the necessary fees. Furthermore, block producers can include fake transactions with no real value, just to manipulate the protocol, generating artificial congestion.

Another important issue with FPA protocols is that the fee prices show extreme volatility, which makes the network highly unpredictable. In times of congestion, when demand for block space surges, the prices skyrocket. We should note that in periods of congestion, transaction inclusion suffers from delays based on the fee attached to them in comparison to the rest of the transactions. As a result, when transactions with elevated fee prices are introduced suddenly, the bids that followed the price levels of the past remain out of the ledger for extended periods. It follows that users do not have a secure way of bidding that will give them the best utility, as they have to risk making a higher bid than necessary or suffer unwanted delays. Making a bid that seems enough at the moment, just before a spike in prices, may lead the transaction to a perpetual waiting list with no upper limit on the delay. On the other hand, bidding their highest fee they're willing to pay would often lead users to overpay for their inclusion. Although digital wallets offer an estimation of fee prices with the corresponding delay, there is no way of predicting when the next price surge will happen and thus those estimations offer no guarantee.

2.2.2 Fixed-Fees Mechanism

One different approach in the TFM field is a fixed-fees mechanism. These protocols, followed by several blockchain systems like Cardano, are based on the FIFO principle. Users submitting transactions pay a constant price and get included in the ledger according to the time of issuance. The fee is calculated as the result of a function that takes into account the processing power used for its inclusion. This means that not all transactions pay the same fee, but rather the same price per resource used. The fee is not directed directly to the producer of the block; it ends up in a pool and is then distributed to all block producers according to the blocks they produced. This way, producers get paid for their work, while at the same time do not have a free pass on the transactions they include in their blocks. Although fixed-fees protocols may be fair compared to other ones, they come with a serious disadvantage. There is no way for a user to have faster service than the one offered at the specific time by the network. The delays are inevitable and no one can buy their way to the front of the line. This has advantages like the eventual inclusion of all transactions and stable prices, but comes with a price of leaving out transactions of greater value. For example, users like DeFi (Decentralized Finance) apps, which require immediate inclusion, even with a high price, are driven out of such protocols. In addition, fixed-fees protocols are more vulnerable to DOS attacks.

2.2.3 Dynamic-Fees and EIP-1559

Dynamic-fees systems update the fee price in every block to control the demand and balance it with supply. Improvement Proposal 1559 (EIP-1559) is a change to a dynamic-fees model made by Ethereum in 2021[13, 27]. It is meant to solve the problems of FPA that was being used before 2021. EIP-1559 brought some major changes to the Ethereum environment. Those changes come down to three basic pillars.

Doubling the maximum block space

The first significant innovation of EIP-1559 was doubling the capacity of the blocks and introducing the desirable capacity, from 12.5 to 25 million gas. [1] However, the desirable capacity remained at 12.5 million, meaning at 50% of the maximum. This means the system aims to fill the blocks only up to half the maximum capacity.

Burning a base fee

The second and most defining feature of EIP-1559 is the existence of a base fee. This fee is both necessary and sufficient for a transaction to be included, meaning transactions with a fee smaller than the base will not enter and no higher fee than the base one is ever paid for inclusion. The fee is burned instead of going to the producer. That means that producers can no longer include transactions of their own freely in the ledger. They need to burn the fee the same way as anyone else. On another matter, the value of the fee is no longer directed to the producers but rather to all stakeholders, as burning an amount of tokens makes the system somewhat deflationary, raising the value of the remaining tokens.

The base fee is adjustable and calculated deterministically based on the state of the previous block, so users know the fee beforehand. As we said, the desirable capacity of the blocks is 50%. If the load on a block exceeds the desirable capacity, the price is raised for the next block, while if the load is lower than the desirable capacity, the price for the next block falls. This way, the traffic in a block remains on average at the desirable load, keeping the system working without the problems of a generally larger capacity. Furthermore, the spikes in traffic are absorbed by the larger maximum capacity and extreme delays are rare, while the system returns to stability after a few blocks. The fee of a transaction is calculated as the product of the transaction gas and the gas price, which is given by the following expression

$$p' = p \cdot (1 + \frac{1}{8}\frac{load - targetLoad}{targetLoad}),$$

where p' is the gas price of the next block, p is the gas price for the previous block, load is the load of the previous block and targetLoad is the desirable capacity.

[1] In Ethereum, block space is counted in gas. Gas represents the computational power needed by a transaction to be included in a block. As we will see later, transactions in Ethereum and other blockchains can perform logic using the ledger as a cloud computer. Thus, the capacity of a block is measured in gas rather than information storage.

Tip for the block producer

The third feature of EIP-1559 is establishing a tip to be given to the block producer. A user submitting a transaction should provide a tip together with the payment. This tip goes directly to the block producer to incentivize them to include the transaction as a priority. This is of great importance to users like DeFi apps, as we saw earlier, or even users who care about not only being included but also in what spot of the block they are being included. However, this tip is relatively small compared to the base fee, so no extreme differences in service occur. Another reason the tip is necessary is that including transactions on a block is not entirely free for a producer. There exists a marginal cost depending on the size of the block published, meaning the load it contains.

EIP-1559 is carefully designed to tackle several FPA issues like unpredictability in fees and high authority of block producers, without having the issues of a Fixed Fee protocol, like arbitrary delays.

2.2.4 Second Price (Vickrey) Auctions

A reader familiar with Game Theory principles might be thinking that some of the issues arising with the use of FPAs are often solved by the use of Second Price, or Vickrey, Auctions. In those auctions, participants bid the maximum amount they are willing to pay and the N winners are charged by the N+1 bid. Unlike FPA, a Vickrey auction is a truthful mechanism that charges the users no more than the necessary. Unfortunately, this mechanism cannot be directly used in a blockchain context. Let us consider the Vickrey equivalent of a TFM. Users would bid their valuations of transactions and the N highest bidders would be included. However, those N winners can't be charged by the N+1 bid, as this bid was not included and fees can only be derived from on-chain verifiable data. There could be variations of the principal design for blockchains, but the pure mechanism is not transferable to this setting.

2.3 TFM Analysis

Numerous papers have analyzed the incentives created for blockchain system users by using different TFMs. This analysis uses game theory tools to prove the stability of each protocol and its resilience to potential deviations. In the following, we briefly cover three main results of this area of research.

2.3.1 Roughgarden's Transaction Fee Mechanism Design

In [54], Tim Roughgarden begins his work by describing an analysis model for transaction inclusion in a blockchain. He then proceeds to break down TFMs into three rules.

- The allocation rule describes how the block producer chooses which transactions will be included in the new block.

- The payment rule calculates the amount of tokens paid by the creator of a transaction to the producer of a new block.
- The burning rule calculates the amount of tokens burnt by the creator of a transaction.

Having modeled TFMs in a formal set of rules, Roughgarden establishes three desirable criteria of incentive compatibility that a TFM must satisfy.

- DSIC (Dominant-Strategy Incentive Compatible) is a term generally used in game theory. A mechanism is DSIC when all players have a dominant strategy that always provides them with the highest utility, no matter the strategies of the other players.
- MMIC (Myopic-Miner Incentive Compatible)[2] is a term introduced by Roughgarden specific for the blockchain analysis model used in this paper. A TFM is MMIC when block producers receive the highest utility by following the protocol rules without introducing fake transactions or deviating in other ways.
- OCA-Proofness (Off-Chain Agreement[3]) is the second term introduced by Roughgarden in the paper. A TFM is OCA-Proof when there exists an individual rational strategy, in which all users and the block producer follow the protocol, granting all of them a greater joint utility than any OCA.

His work is concluded by using the model established to analyze the following TFMs

- FPA The common first-price auction, used by Bitcoin and other blockchains.
- SPA A second-price-auction-like TFM, defined by Roughgarden in the paper.
- β-burn FPA A first-price auction TFM where part β of the payment is burnt.
- 1559 The EIP-1559 protocol used by Ethereum.
- β-burn 1559 The EIP-1559 protocol but where only a part β of the payment is burnt.
- tipless A version of EIP-1559, proposed by Roughgarden, where the tip is fixed to a constant value.

2.3.2 Chung and Shi, Foundations of TFM Design

Following a similar line of work, Hao Chung and Elaine Shi introduced a slightly different model in [20]. They keep the three incentive criteria model.

- UIC (User Incentive Compatibility) is identical to DSIC, used by Roughgarden.

[2] Myopic Miner is a term used in blockchain analysis meaning a block producer trying to maximize her utility for the current block, without strategizing for long term gains of a future blockchain state.

[3] An off-chain agreement is the collusion of the users and the agent out of the blockchain to bypass the protocol rules

- MIC (Miner Incentive Compatibility) is identical to MMIC, used by Roughgarden.
- c-SCP (c Side Contract Proofness) is a new off-chain proofness notion introduced in the paper and is used in the place of OCA, used by Roughgarden.

A TFM is c-SCP when no coalition of the block-producer and up to c users can give them a joint utility greater than following the protocol without deviations. The higher the c, the more the options for collusion are covered and thus the stricter the notion.

Chung and Shi use Myerson's lemma to prove the following lemma and then their theorem.

Lemma: Any TFM that is UIC and 1-SCP must have 0 block producer revenue[4], whether it has finite or infinite block size.

Theorem: Assume a finite block size. Any non-trivial TFM cannot be both UIC and 1-SCP.[5]

Chung and Shi proceed to introduce a new protocol based on two principles.

- It is generally considered that a user can make a fake bid and get away with it as long as the transaction is not confirmed. A malicious user may bid higher for a transaction provided that she knows it will not be confirmed, and thus she will not be charged for the fake bid. However, while it is true that non-confirmed transactions pay nothing, it might be the case that the transaction will be later confirmed and the user will have to pay for it. The protocol is based on this idea, introducing the concept of $\gamma \in [0, 1]$. γ denotes the fraction of the full price that the user will be charged at some point for the fake bid. The worst case comes with $\gamma = 1$, when the user has to pay the full bid and the user pays nothing when $\gamma = 0$.
- Allowing the inclusion of transactions that are not confirmed in a block is the subject of debate, as blockspace is considered a valuable asset. However, this practice can have interesting benefits. Chung and Shi permit the inclusion of transactions that are not confirmed and serve as the price setting for the fees paid by the rest, as well as the revenue received by the miner.

[4] Here revenue refers to the block producer's revenue from fees and not the reward she gets for creating the block, which is separate.

[5] As we said 1-SCP is the looser notion, so if the theorem holds for 1-SCP it holds for every n-SCP

The burning second price auction

The protocol is based on a specific pair (γ, c), where $\gamma \in (0, 1]$ and $c \in \mathbb{N}$ is the maximum coalition size (of c-SCP). Let B be the blockspace, then we pick k, k' such that

$$k + k' = B \tag{2.1}$$

$$0 \leq k' \leq \frac{\gamma \cdot k}{c} \tag{2.2}$$

[6]

Assume $(b_1 \geq \cdots \geq b_B)$ are the B highest bids that get included in the block, presented in descending order. A set $S \subseteq \{b_1, \ldots, b_k\}$ of transactions is randomly selected and confirmed, the rest of the transactions in $\{b_1, \cdots, b_k\} \setminus S$ are included but not confirmed, the same as the ones in $\{b_{k+1}, \ldots, b_{k+k'}\}$. The confirmed transactions are paying a fee equal to b_{k+1}, the rest pay nothing. Finally, the miner receives revenue equal to $\gamma \cdot (b_{k+1} + \cdots + b_{k+k'})$.

The protocol is proven to be UIC, MIC and c-SCP for any $c \geq q$ and $\gamma \in (0, 1]$.

Finally, Chung and Shi prove the necessity of randomness in a protocol that aims to be UIC and c-SCP, for $c \geq 2$, as no deterministic TFM can be UIC and 2-SCP, even with $\gamma = 1$.

2.3.3 Roughgarden, Chung and Shi's work

In [19], Tim Roughgarden teams with Hao Chung and Elaine Shi to investigate further impossibilities regarding collusion resilience in TFMs. Although Chung and Shi had already proven the impossibility of a dream TFM using the c-SCP collusion notion, the original question of Roughgarden using the OCA-proofness notion was still open.

The two notions have a core difference. Where c-SCP forbids up to c users to collude with the miner, potentially at the expense of the rest of the users, OCA-proofness demands that no coalition can exist between all the users and the miner to steal from the protocol.

In their work, they introduce a new, stepping-stone notion, global-SCP. In global-SCP, there is no strategy followed by all users and the insurer that provides a greater joint-utility than the honest one. The only difference between global-SCP and OCA-proofness is that the latter accepts non-truth-telling strategies as long as they are individually rational, the miner respects the inclusion rule provided by the protocol, each user bids independently without knowing anyone else's bid, and there are no fake bids injected.

Their work concludes in two results.

- Assuming a finite block size, no TFM is satisfying UIC, MIC and global-SCP at the same time. This result holds for randomized and non-direct revelation mechanisms.

[6] [Tsironis: How do we know k' ain't gonna be just one transaction?]

- Assuming a finite block size, no TFM is satisfying UIC, MIC and OCA proofness at the same time. This result holds for randomized mechanisms and even when the globally optimal strategy is not individually rational, but not for non-direct revelation mechanisms.

2.4 Blockchain Derivatives

TFMs tackle, among others, the issue of service predictability in a blockchain system. A different approach comes in the form of blockchain derivatives. Similar to derivatives in traditional economic theory, blockchain derivatives are essentially an agreement about a future exchange of an underlying value, in our case, block space.

2.4.1 Blockchain Space Tokenization

In [37], a new mechanism is introduced, called Blockchain Space Tokenization. Although the BST mechanism is not a contract between two parties, it displays some derivative characteristics. It is essentially an agreement between users and the protocol; the latter provides tokens for a fee that can be later used instead of payment for transaction inclusion.

BST Mechanism

A blockchain system can allocate part of the available block space to facilitate this functionality that works in parallel to the traditional transaction inclusion. Tokens of different values are issued by the protocol and sold in an auction. Users owning a token can use it to include their transactions. Users' transactions get included based on a priority value depending on the value of the tokens used, the age of the tokens used and the size of the transaction. Each transaction may use more than one token and each token can be used multiple times.

- The age of each token is calculated by the number of blocks published since the token was last used.
- Each token has a value, transactions can use multiple tokens, adding their values to get higher priority.
- The priority of a transaction is inversely proportional to the transaction's size. The larger the transaction, the smaller its priority.

Furthermore, transaction priority has an upper limit, to defend against attacks flooding the system with old tokens of small value. This limit is based on the total token value and the size of the transaction. The resulting expression giving the priority is the following.

$$priority(tx) = \min\left\{\frac{\sum_{tok\in tx} val(tok)\cdot age(tok)}{size(tx)}, \frac{4}{(1-\mu)\cdot k}\cdot\left(1+\frac{val(tx)}{size(tx)}\right)\right\}$$

Where $tok \in tx$ are the tokens used by transaction tx, $val(tx)$ is the sum of token value used in tx, k is the block space and μ is a constant explained later.

Mechanism behavior in worst case scenario

In the paper, an analysis model is used to assess the mechanism. This model assumes the existence of an adversary that has power over the protocol and is trying to maximize the delay of a transaction. More specifically, the adversary can do the following

- Send transactions, honest or false, to the block producers.
- Issue her own blocks, sharing them to the network.
- Order the creation of a block from a specific block producer.

However, the network retains a proportion of honest blocks no less than $1 - \mu, \mu \in [0, 1)$.

Having established this adversary model, an upper bound on the worst-case delay is proven. That is the maximum number of blocks an honest transaction has to wait from the moment it is submitted to the moment it gets included.

The work concludes by making modifications to the mechanism in order to make it off-chain proof. After the modifications, the new mechanism retains its service reliability with upper-bounded worst-case delay, while at the same time is off-chain proof. That is, the parties participating cannot raise their utilities by colluding to bypass the protocol.

2.4.2 Pitch Lake

Pitch Lake [50], developed by Oiler Network, is a call option contract using Ethereum's base fee as its underlying value. A DeFi vault funds the minting of the options, which are then sold through a batch auction to users. After their expiration, the options can be executed by the owners. If the base fee on settlement is above the strike value, the owner is funded accordingly.

A user trying to defend against base fee volatility may use this functionality to protect a future transaction from a potential surge in the network. If a surge occurs when the user submits their transaction, they get compensated for the elevated prices by the call option, which pays them back accordingly.

The option uses a time-weighted average (TWAP) instead of the actual base fee. This TWAP is used to calculate the strike, as well as settle the option. The strike value is the result of the TWAP multiplied by $(k + 1)$. There are 3 options minted, one Out of The Money $(k < 0)$, one In The Money $(k > 0)$ and one At The Money $(k = 0)$. Option owners are paid up to $cl \cdot K$, where cl is the collateral deposited in the contract upon creation. The process works in the following steps

- The base fee TWAP BF_{T_1,T_2} of interval $T_1 - T_2$ is derived and the strike value $K = BF_{T_1,T_2} \cdot (k + 1)$ is calculated.
- The batch auction is held starting at a reserve price. The reserve price is calculated using Ethereum's average staking rate and the base fee standard deviation, derived from historical data.

- The options are settled after interval $T_3 - T_4$, using TWAP BF_{T_3,T_4} as the settlement value.

The vault managing the options functions repetitively with the liquidity gathered from one contract cycle, funding the next. Intervals $T_1 - T_2$ and $T_3 - T_4$ have the same duration τ. Thus, BF_{T_3,T_4} used for settlement on cycle i, is also used for strike calculation on cycle $i + 1$.

2.4.3 LedgerHedger

In [69], Tsabary et al. introduce a derivative protocol, called LedgerHedger. LedgerHedger is essentially a futures contract for the purchase of space allocation on the Ethereum network.

As we've seen before, the Ethereum block space market is subject to great price volatility. LedgerHedger aims to tackle that by introducing an agreement between a seller, owning space allocation on the Ethereum network, and a buyer interested in buying this space allocation for a predetermined price at some point in the future. The role of the seller fits better to a block producer but can be any user as well.

The protocol makes use of smart contract technology to implement the protocol logic in the Ethereum environment. The smart contract executes the protocol automatically, minimizing the room for deviation of the two parties.

The protocol is executed in two phases, the initiation phase and the execution phase. In the Initiation phase, the buyer initiates the contract by setting its parameters and depositing the payment, which will be received by the seller for the successful completion of the contract. Following the creation of the contract, the seller accepts it by depositing collateral, which will be returned when the contract is completed. The collateral protects the buyer from a malicious seller who will abandon the contract.

The execution phase consists of three different outcomes. The successful one is followed when the buyer provides their transaction, the seller includes it in the ledger and receives the funds from the contract. The second outcome finds the contract without a seller to accept it; in this case, the buyer retrieves the funds deposited and the contract is called off. Finally, there is a possibility that a malicious or faulty buyer does not provide a transaction as expected. Then the seller can waste their allocation, filling it with trash and receiving part of the funds from the contract. This way, they are compensated for their participation, but they are also disincentivized to do so if they have received the transaction.

The two phases consist of a total of 5 functions. Two functions make up the initiation phase and three the execution phase. Those functions are the way for the two parties to interact with the contract. Each of the functions is a transaction submitted to the ledger.

The initiation phase consists of the following two functions:

- **Initiate** The Initiation phase begins with the Initiate function submitted by the buyer. Using this function, the buyer sets the parameters of the contract, namely the beginning and ending of the phases, the size of the allocation they need and the collateral required by the seller to accept the contract. In addition, the buyer deposits the payment that the seller will receive if the contract is concluded successfully.

- **Accept** Before the end of the initiation phase, a seller can choose to accept the contract. To do that, they submit the Accept function, depositing the collateral required by the contract.

The execution phase concludes in three different ways, depending on which of the three different concluding functions is used.

- **Recoup** If the Initiation phase concludes without a seller accepting the contract, the buyer can submit the Recoup function to withdraw the funds deposited to the contract.
- **Apply** The seller submits the Apply function to complete the protocol by using their space allocation to include the buyer's transaction in the ledger.
- **Exhaust** If the buyer does not provide a transaction during the execution phase, the seller can submit the Exhaust function to receive part of the payment deposited in the contract.

The work continues by analyzing the conditions affecting the incentives of the two parties participating in the protocol.

- **Protocol Parameters**: The costs of the protocol functions and the earnings caused by the interactions with the protocol.
- **Gas Price Distribution**: The random distribution describing the block-space price fluctuations.
- **Utility Functions**: The function that maps the earnings of two parties to the utility that describes their happiness levels.

The analysis shows how the payment and the collateral are picked depending on the factors above to maximize the incentives of the parties to follow the protocol.

It concludes by proving, using Game Theory, that as long as the protocol is initiated by a buyer and accepted by a seller, its successful conclusion is a Subgame Perfect Equilibrium 4.4.2. That is, both parties are incentivized to follow the protocol with no deviations.

Chapter 3

Cardano

This work will focus on the Cardano Ecosystem. Cardano is a third-generation Blockchain network founded by Charles Hoskinson. Its native token is called ADA, with its sub-unit called Lovelace. The underlying consensus protocol is a Proof-of-Stake algorithm called Ouroboros.

3.1 A Third-Generation Blockchain

Blockchain systems can be categorized into generations. The first-generation blockchain protocols is the one initiated with the release of Bitcoin. First-generation protocols introduced the fundamentals of blockchain technology, initiating the work on decentralized cryptocurrencies. The second generation, with Ethereum being the most prominent system, extended the blockchain functionality so that it accommodates the decentralized execution of code in the form of smart contracts.

The third-generation protocols aim to tackle three main issues the blockchain technology faces since its early days: Scalability, Interoperability and Sustainability. Other prominent third-generation protocols are Algorand [18], Polkadot [12] and Avalanche [57].

3.1.1 Scalability

Scalability refers to a network's ability to accommodate a growing number of users. There are three main challenges when more and more users use a blockchain system:

- **TPS**: Transactions per second is the number of transactions included in the ledger every second. TPS is one of the main challenges of blockchain protocols, as they cannot yet be compared with traditional financial transactions in this regard.

  Cardano introduces Ouroboros, the first Proof of Stake system to be proven secure. PoS systems are a lot cheaper than their predecessors, Proof of Work, requiring way fewer resources. As a result, Ouroboros stakeholders can maintain a possibly greater number of chains and thus accommodate a larger TPS.

- **Network**: Apart from the validation and inclusion requirements of a larger TPS, a growing number of users generates more traffic to the network, which requires greater bandwidth. As the traffic grows larger and larger, a homogeneous peer-to-peer network where every node forwards all messages to every other node is no longer viable.

Cardano tackles this issue by introducing RINA (Recursive Inter Network Architecture)[24]. RINA aims to create a heterogeneous network with the same guarantees so it can handle the increasing traffic while at the same time remaining secure.

- **Data Scaling**: The last challenge associated with an increasing number of users is the data storage of an ever-growing ledger. With the inclusion of more and more transactions, the memory requirements of the nodes storing the copies of the ledger become problematic. Furthermore, a new node joining the protocol will have a much harder time bootstrapping, as there is much more to catch up on.

  Cardano researches this issue, exploring solutions that include

  – Pruning redundant data from the blockchain

  – Compression methods

  – Partitioning, where parts of the ledger are kept by different nodes

3.1.2 Interoperability

As more and more blockchain systems emerge, it becomes clear that there will never be a single prominent protocol. Communication between protocols has been essentially non-existent, with the weight falling on exchange platforms. However, blockchain platforms do not generally share the decentralized vision of blockchain systems and thus no matter how well those systems implement decentralization, it collapses as these platforms become essential and gain power. Furthermore, entities intending to do business using cryptocurrencies most of the time need to partner with traditional finance systems as well. Thus emerges a need for interoperability between protocols as well as traditional finance without the need for a trusted third party.

Cardano aims for interoperability with other blockchain protocols through the use of sidechains [30, 40] A sidechain is essentially a highly compressed view of the chain of a different protocol. By keeping sidechains of other protocols, Cardano can communicate and interact with them.

Nevertheless, the major difficulty comes with the connection to the traditional finance systems. This difficulty comes down to three main issues.

- **Metadata**: Metadata is the information surrounding a transaction. When the transaction is the payment of x amount of tokens, the metadata is the context of what the payment was for, information about the sender and recipient and more.

  The traditional finance heavily relies on that data to estimate the risk of a transaction, while most of the blockchains completely disregard it as it tends to be private information. Privacy is one of the main pillars of cryptocurrencies and so the existence of such data posted on an immutable and completely transparent ledger raises concerns.

Cardano explores ways of including the bare minimum of such data, essential for interactions with traditional systems, while at the same time, keeping the private information encrypted or out of the chain.

- **Attribution**: Attribution is a kind of Metadata whose importance requires special treatment. Among all context information of a transaction, the most important is probably the identity of the parties involved. While in traditional banking, there is available information about the physical identity of an entity participating in a transaction, in the digital system, we rely on codes that are not sufficient for providing the information needed.

  The issue is the cryptographic complexity required to securely identify an entity, a complexity that most websites cannot handle. Fortunately, cryptocurrencies have an abundance of cryptographic tools available for these kinds of purposes.

  Cardano intends to explore ways to utilize these tools so that it can permit users to add attribution to their transaction should they wish to.

- **Compliance**: Finally, we have compliance, which is the way of traditional banking to ensure a transaction is not part of money laundering, terrorist financing and/or any other criminal activity. Naturally, banks have strict regulations against this kind of transaction, which makes the interaction with cryptocurrencies extremely difficult, as the latter do not pay respect to these notions.

  Cardano is confident that by utilizing ways to permit the voluntary existence of Metadata and Attribution, it will be able to provide the necessary guarantees required for compliance.

3.1.3 Sustainability

Sustainability is the last of the pillars Cardano is built upon and it means the ability of a protocol to maintain itself as time passes. It can be broken down into two aspects: a) Maintenance cost and b) Design Decisions.

Maintenance Cost By maintenance cost, we refer to the funding mechanisms supporting the protocol. Blockchain protocols are not companies having expenses and revenues, but rather decentralized open-source networks. An entity that would act as a sponsor for the system would inevitably be a threat to its decentralization. By controlling the direction of the system's development, it would draw excessive power. An initial pool of funds would also not be sufficient in the long run, as it would eventually run out, regardless of its size.

Cardano solves the problem with the creation of a treasury. The treasury is a pool of funds sustained by the protocol and used to provide liquidity for the development purposes of the system. The treasury is partly funded by inflation, that is, a certain amount of coins released to the network every epoch[1]. Part of the fees paid by the users for their transactions' inclusion

[1] Part of the newly released coins are also directed to the producers of the new blocks

is also directed towards the treasury. Cardano then holds a democratic voting system that determines where those funds will be directed. Each user can submit a development proposal and if it gathers sufficient votes, the proposal will be funded.

Design Decisions By design decisions, we refer to the direction followed by the protocol. Blockchain protocols are not static; they grow and adapt to new technologies and goals. This can be done either by soft forks or hard forks, which introduce novelties to the protocol. However, there is no obvious ”correct” version of the protocol after the fork. This leads the two versions to take different directions till they become two completely different protocols, e.g. Bitcoin - Bitcoin Cash and Ethereum - Ethereum Classic.

Cardano introduces a democratic mechanism of proposing innovations and changes to the system, so they can then be implemented, provided that they get the necessary endorsement from the community. The proposals submitted to the mechanism are called CIPs (Cardano Improvement Proposals) and have the objective of reaching a consensus between the users of the protocol regarding the direction of future development.

3.2 Trust in Cardano

One of Cardano’s primary objectives is to establish a reliable and trustworthy ecosystem. This is achieved in two ways: a) Peer-reviewed academic research on its core components and b) Using high-assurance code for its smart contracts.

Peer-Reviewed Research Cardano’s innovations are backed using rigorous academic analysis regarding their stability and security. Papers like ”Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol”[39] and ”Reward Sharing Schemes for Stake Pools”[11] have been published in peer-reviewed journals and conference proceedings. Such a foundation makes Cardano stand out among blockchain platforms as a well-founded protocol grounded in formal cryptographic proofs.

High-Assurance Code Transactions in Cardano are meant to be deterministically successful at the time of submission to the network. Various design principles, such as deterministic fee calculation and high-assurance code, achieve this. The smart contract language of Cardano, called Plutus, is a Haskell-based functional language. The constraints and immutable data structures inherent in functional languages differentiate them from other programming paradigms, such as imperative or object-oriented, and contribute to the development of more error-resilient software. Unintended side effects from code execution are markedly limited. The Cardano EUTxO model further enhances predictability by permitting offline smart contract execution before on-chain submission. As a direct outcome, the successful execution of every transaction appended to the ledger is assured.

3.3 Cardano Roadmap

Unlike most blockchain protocols, Cardano's was developed following a detailed roadmap. The roadmap consisted of five development phases[6, 63, 34], each representing a stepping stone toward the full implementation of the system. This step-by-step introduction of the various components allowed the gradual adaptation of the system to the sophisticated structure of Cardano.

- **Phase 1: Byron (Foundation):** The first development phase, named after the poet Lord Byron, laid the foundations of the Cardano system. The blockchain network was created, and the first native ADA coins were released, named Ada in honor of Ada. Lovelace, considered the mother of computer programming. The consensus protocol was Ouroboros Classic and later Ouroboros BFT (Byzantine Fault Tolerance), an intermediary step towards Ouroboros Praos. The network was maintained by IOHK, the research and development company behind Cardano and it was not yet decentralized. Byron was launched in September of 2017 and included the homonymous Byron era.
- **Phase 2: Shelley (Decentralization):** The second phase[61], named after the poet Percy Shelley, decentralized the network, adopting the consensus model of Ouroboros Praos. Maintenance responsibilities shifted from the protocol founders to the community as the staking and delegation functionalities were introduced. Shelley was launched in July of 2020 and included the homonymous Shelley era.
- **Phase 3: Goguen (Smart Contracts):** The third phase, named after the computer scientist Joseph Goguen, introduced smart contract functionality. This development phase consists of three different eras.
  - Allegra Era[59]: Named after Byron's daughter, Allegra Byron and launched in December of 2020, it introduced simple locking of tokens using smart contract logic.
  - Mary Era[65]: Named after the novelist Mary Shelley and launched in March of 2021, it introduced the creation of custom native tokens and NFTs.
  - Alonzo Era[60]: Named after the mathematician Alonzo Church and launched in September of 2021, it introduced the EUTxO accounting model and Cardano's smart contract functional language Plutus. Allowed for the creation of more complex smart contracts with explicit state management.
- **Phase 4: Basho (Scaling):** The fourth phase, named after haiku poet Matsuo Bashō, focused on accommodating a larger user base. It introduced Hydra, a layer-2 solution meant to relieve congestion on the main chain. Part of this Phase is the Babbage era, named after the inventor Charles Babbage, launched in September 2022 with the Vasil hard fork[66]. It brought in numerous changes aimed at transaction enhancement with Plutus V2, inline datums and more.
- **Phase 5: Voltaire (Governance):** The fifth and final phase of the development, named after the Enlightenment philosopher, aims towards the full decentralization of the protocol. It places the full control of the network on its stakeholders, who have the power

to propose and vote on future changes and can shift the development to new directions. It was launched in two steps, the Chang hard fork in September and the Plomin hard fork in December of 2024[32], initiating the Conway era, named after the mathematician John Horton Conway.

3.4 Ouroboros

As mentioned, Cardano's consensus algorithm is Proof of Stake, where stakeholders are chosen to produce blocks depending on the funds they have locked to participate in the process. The specific model run by Cardano is called Ouroboros. Ouroboros was the first PoS protocol formally proven to be secure in [39]. Following the initial Ouroboros model, there have been multiple updates in the Cardano ecosystem, both predecessors of the current Ouroboros Praos[23] and versions to be implemented in the future. A brief overview is shown below[67].

1. **Ouroboros Classic**: The first implementation of the Ouroboros protocol used in the first stage of Cardano development.
2. **Ouroboros BFT**:[38] Ouroboros Byzantine Fault Tolerance was the intermediary step between the initial implementation and the current protocol. It laid the groundwork for the decentralization of the network and stakeholder participation.
3. **Ouroboros Praos**: The current Cardano consensus protocol. It adapted the Ouroboros protocol to a semi-synchronous model.
4. **Ouroboros Genesis**:[2] It will enhance Ouroboros Praos, facilitating the bootstrapping of new nodes.
5. **Ouroboros Crypsinous**:[35] It will advance the privacy-preserving properties of Ouroboros Genesis.
6. **Ouroboros Chronos**:[3] It will introduce a novel time synchronization mechanism and create a cryptographically secure source of time.
7. **Ouroboros Leios**:[21] The most recently published Ouroboros paper, aims to improve the scalability of the protocol, facilitating greater traffic.
8. **Ouroboros Peras**:[4] Ouroboros Peras is an extension of Ouroboros Praos designed to introduce a novel finality notion based on a voting system held by the stakeholders.

3.4.1 Ouroboros Praos

The protocol divides time into epochs and, furthermore, into slots. In each slot, a stakeholder is randomly selected with a probability relative to their stake to be the slot leader. The slot leader collects valid transactions and includes them in a newly created block, which then proceeds to attach to the chain. More specifically, the protocol follows this procedure:

The protocol is initialized with the creation of a genesis block containing information on the stakeholders participating, along with their respective initial stake. Furthermore, the genesis block contains a nonce. This nonce is the seed that will be used in the random function determining the upcoming slot leaders.

The random function used for determining the slot leaders is implemented using a cryptographic tool called verifiable random function(VRF) [2]. VRFs are essentially unique functions that stakeholders use to produce a pseudorandom value using a seed. Furthermore, a VRF also provides a proof π, verifying that the value was indeed produced by the specific VRF used by the specific stakeholder.

To calculate their VRF value for a specific slot, stakeholders need a seed. This seed needs to be the same for all stakeholders and unique for each slot. It is calculated by concatenating a nonce assigned to every epoch and the number of the current slot. So, the seed of the i-th slot of an epoch with a nonce h, is the value $h||i$.

Having the slot seed, the stakeholders calculate the respective VRF value. The leaders of this specific slot are the stakeholders whose VRF value is less than a threshold, related to their respective relative stake, that is, the proportion of the total stake they control. As described, the rest of the stakeholders can verify if the VRF value provided by an alleged slot leader is indeed correctly calculated and less than the required threshold, as the stakes are publicly known. The probabilities of each stakeholder being the slot leader are independent of each other. Every slot can have multiple leaders eligible to create a block or none whatsoever.

The nonce assigned to an epoch is calculated by concatenating the VRF values of the first blocks of the previous epoch and then feeding them into a hash function. The resulting value is the nonce shared with the stakeholders at the beginning of the current epoch. In the course of the execution of the protocol, multiple versions of the blockchain arise, originating from a common start. This can happen due to the creation of multiple valid blocks in a slot, possible network latencies resulting in different views of the chain by different stakeholders, or by malicious parties. Stakeholders receive these different versions of the chain and have to pick which one to extend when creating a new block. The selection of the right chain is deterministic and is designated by the protocol. Stakeholders have one version of the chain they previously deemed correct; as time passes, they receive updates with new versions of the chain, collecting them in a set. The next correct chain is the one containing the most blocks, while at the same time differs at most to a specified degree from the previous correct chain. Ties are broken by favoring the previous correct one if it is part of the tie and then the first one received.
The whole process is executed in the following steps:

- **Initialization**: Genesis block is created, containing the nonce of the first epoch together with the stake distribution of all stakeholders participating in the protocol.
- **Slot Leadership**: In every slot i of each epoch, the stakeholders check whether they are eligible to create a transaction. They get the concatenation (h||i), where h is the nonce of

[2] You can find more documentation about VRFs in the Cardano Docs [62]

the current epoch, and use it as the seed to their unique VRF. The results are their VRF value and the respective proof π. If the VRF value is less than a threshold derived by their relative stake, they are the slot leader, that is, they are eligible to create a block in the specific slot.

- **Chain Selection**: Each time a block is created, the stakeholders receive the new version of the chain. A stakeholder receives multiple versions of the chain, possibly contradicting each other. After checking the validity of those chains, she picks the one with the largest number of blocks, making sure that the selected chain does not differ more than a certain threshold from the last accepted chain.

  After checking the validity of those chains [3], she picks the one with the largest number of blocks, making sure that the selected chain does not differ more than a certain threshold from the last accepted chain.

- **Epoch Nonce Generation**: Once an epoch is over, the VRF values of its first blocks are fed to a hash function, resulting in the nonce of the next epoch.

3.5 Cardano Rewards

Like many other blockchain systems, Cardano uses inflation to gradually release funds to the network. An initial reserve containing the total supply of coins was created at the introduction of the protocol. Every epoch, a part ρ of the reserve is released. As a result, the reserve depletes not only gradually, but also at a decreasing rate, so that the reserve will last for a prolonged time.

Funding Process The released funds are added to a pot that has a dual purpose: a) to provide rewards to block creators, incentivizing them to maintain the chain, b) to fund the treasury that provides liquidity for the development of the protocol [16, 46, 11]. Part T of the pot is directed to the treasury, while the rest is given as a reward for the block creation process.

Stake Pools Cardano's stakeholders can be organized in stake pools, acting as a single entity, with its stake being the sum of the individual members' stakes. This allows minor stakeholders to join forces, creating a pool with considerable power. The pool is operated by the SPO (Stake Pool Operator) who is responsible for maintaining the pool, executing the consensus and block creation process, and distributing the rewards to the members of the pool.

Handling Rewards Every epoch, the SPO collects the reward earned by the pool for the blocks created. A part of the funds received is kept by the SPO to cover: a) the cost of operating the pool and b) a margin that incentivizes stakeholders to take on the responsibility of operating the pool. The rest of the funds are distributed to the members of the pool proportionally to their stake.

[3] exact definition of a valid chain can be found on Definitions 2, 4 of [23]

3.6 EUTxO

Cardano's accounting model, called EUTxO[17, 55], is an extension of the one used in Bitcoin. The new features added to Bitcoin's UTxO model provide Cardano with a greater range of functionalities. More specifically, the Cardano can be used for the execution of code much more complex compared to the code supported by Bitcoin.

The first extension to the UTxO model is in the UTxO tokens. In Bitcoin's model, a UTxO consists of an identifier, an address and a value. EUTxO model keeps those attributes and adds to them a fourth, the datum. The datum is an additional piece of data attached to the UTxO in a transaction and can keep information about the ledger state, which can then be used for the execution of code.

- **Value**: It specifies the value contained in the UTxO
- **Identifier**: The identifier of a UTxO is a number unique to the specific UTxO by which users can refer to it. It consists of the hash of the transaction that produced it, followed by the output index that separates it from the rest of the outputs of the transaction.
- **Address**: The address is the public key hash of the user controlling the UTxO or the hash of a script containing a piece of code that needs to be satisfied for the funds to be unlocked to a recipient.
- **Datum**: Piece of data carrying information about the larger state.

There are two different kinds of data: a) the datum hash and b) the inline datum.

- In datum hash, only the hash of the datum is attached to the UTxO. A user attempting to unlock the funds stored in the UTxO has to provide a datum matching the datum hash to unlock the funds.
- In inline datum, the whole datum is attached to the UTxO. This way the user doesn't have to provide the datum as it is visible on the UTxO

EUTxO model brings two more changes in the UTxO model. These changes are focused on the transaction instead of the UTxO token and we will discuss them below.

3.7 Transactions in Cardano

In Cardano, as in every blockchain, the users interact with the ledger by submitting transactions to the network. The block producers gather these transactions, creating blocks and include them in the blockchain. Cardano uses the EUTxO model, which is similar to Bitcoin's UTxO model with some extensions. In this model, the transaction takes UTxO as inputs and produces new ones as outputs.

The EUTxO model introduces two novelties to transaction creation: a) The validator has access to a set of data, called Script Context, which carries information about the specific transaction. b) The transaction is valid for a time period, called the validity interval, and can be included only inside this time period, while it is invalid before or after.

We list the following essential characteristics of the transaction process in the EUTxO model.

- **Transaction ID**: Each transaction has a unique identifier that users use to refer to it.
- **Inputs**: A transaction consumes UTxOs (unspent outputs) of previous transactions. Inputs are funding the transaction.
- **Outputs**: The transaction creates new UTxOs, transferring the funds to the new owners, or locking them in a smart contract.
- **Fee**: A user submitting a transaction has to pay a fee for the transaction to be included in the blockchain. Part of this fee is transferred to the block producers creating the blocks and the rest is used to fund the blockchain development.
- **Scripts**: Outputs of a transaction can contain a validator using a script. The script is a piece of code used to validate the unlocking of the output funds by a user.
- **Script Context**: A piece of data carrying information about the transaction. Script Context is utilized by the validator as input.
- **Redeemer**: A piece of data provided as credentials by a user attempting to unlock funds from a smart contract.

3.8 Smart Contracts

Modern blockchain systems are used for a wider variety of purposes than just cryptocurrency, as they were originally deployed in Bitcoin. Decentralized apps can use blockchain technology to execute logic in the form of pieces of code called Smart Contracts. Smart Contracts in cryptocurrency can be viewed as the equivalent of contracts in traditional finance. The Smart Contract logic is imbued into the transaction and gets executed when the transaction is confirmed and included on the ledger.

In this work, we focus on smart contracts as they are implemented in the EUTxO accounting model. As we saw earlier, UTxOs use a locking mechanism, called a validator. This validator carries the logic of the smart contract. A validator can be viewed as the blueprint that defines the logic of the smart contract. Users can use this blueprint to create their own contracts specifying their details.

In a traditional finance analogy, the validator could be a general contract of a house sale, created by a lawyer, with blank details. The parties that would make the sale would fill in their details and the specific house properties to create a specific instance of a sale contract. This would be the analogue of a smart contract.

The validator is stored at an address, which works as a vault. Users can interact with this vault in two different ways, either locking funds inside or retrieving funds stored in the vault. Those interactions are made using a transaction. To lock funds, a user, whom we shall call the sender, creates a transaction that outputs a UTxO with address the one of the validator. To retrieve the funds, a user, whom we shall call the retriever, creates a transaction consuming a UTxO with address the one of the validator.

As we described, the same validator-vault can hold multiple contract-instances. The lock is configurable and is instantiated every time a sender locks funds inside. Each UTxO stored in the vault represents a different contract and a retriever trying to consume it has to provide credentials for its specific lock.

The validator is a function that gets three inputs and outputs a boolean. The inputs are the datum, the redeemer and the script context.

- The datum contains the details specifying a contract instance derived from the validator blueprint. It can carry specific signer information, validity periods or any other specification. The datum is provided by the sender in the contract creation transaction and is held by the smart contract UTxO.
- The redeemer is a piece of data that has the role of credentials, provided by the retriever when attempting to consume the contract UTxO.
- The script context contains information about the transaction consuming the contract UTxO, such as the validity interval of the transaction, the input UTxOs, the output UTxO and more.

The validator is called when the retriever is attempting to consume the contract UTxO. It uses the three input parameters and outputs True if the unlocking is successful and False if the unlocking fails. If it is successful, the retriever gets the funds, while if not, the transaction fails.

3.9 Fees

Cardano is a fixed-fee blockchain system. The fees needed to be paid for a transaction to be included in the ledger are deterministically determined based on the size of the transaction and the calculation is the same for every transaction and does not change over time.

The fees for every transaction are gathered in a pot and then they are directed to block producers and the Cardano treasury. The block producers receive part of the fees indirectly from the pot as a reward for the blocks they create. The rest of the fees are collected in the Cardano treasury and then used for the development and maintenance of the network.

Transactions pay fees relative to the resources they require for their inclusion, so fees are based on the following factors[64, 45, 10]:

- **Size**: Transactions are essentially data uploaded and stored in the blockchain. The size of these data is the basic factor for the fee calculation of a transaction and is determined by its inputs and outputs

- **CPU Steps**(for smart contracts): A transaction involving a smart contract requires the execution of code for its inclusion in the ledger. CPU steps computational effort consumed by the transaction.
- **Memory Units**(for smart contracts): The code executed for the transaction also requires memory usage to hold data. Memory Units express this usage, so they are accounted for in the fee calculation.

Simple transactions pay fees equal to $fee = a \cdot tx(size) + b$, while those involving scripts require an additional cost equal to $script\_fees = cpu\_steps \cdot cpu\_price + memory\_units \cdot memory\_price$.

3.10 Nested Transactions

Our later results involve an agreement between two users. The agreement is included in the ledger in the form of a smart contract jointly created by the two parties. One possible implementation is based on a novel functionality of Cardano called Nested Transactions. Nested Transactions functionality, introduced in [70], is currently under review and not yet deployed on the Cardano Network.

This functionality allows the creation of, possibly unbalanced, sub-transactions completed by a top-level transaction, forming a valid batch that can be included on the ledger. A sub-transaction can lack various necessary components (e.g., fees, collateral, etc.) covered by the top-level transaction. Moreover, a transaction can specify as input by reference, that is, referring to inputs that are actually provided by another transaction in the same batch.

In addition, Nested Transactions provide greater freedom in script usage. A sub-transaction may include a script defining conditions that need to be followed by the rest of the sub-transactions in the batch in order for it to be valid. Furthermore, sub-transactions in a batch have access to the script data of each other. These novelties enable greater collaboration between users.

Chapter 4

Game Theory

Game Theory is the field that studies relations between independent agents (or players) who aim to maximize their utility in any given situation. By using models, real-world scenarios are expressed as games using strict mathematical models. In this chapter, we will follow the definitions of [56].

4.1 Utility Function

As mentioned before, the notion with which desirable outcomes of players are measured is utility, a real number value of their satisfaction. We view any real-world scenario as a game with a specific set of states. Players take actions that affect the outcome of the scenario and bring the game to a resulting state. Each player has a specific utility for any of these states. We call a utility function the mapping from the states to the utility of the players.

4.1.1 Prisoner's Dilemma

Probably the most famous example of a scenario analyzed with game theory is the prisoner's dilemma. In this game, two criminals are captured and asked to confess, turning in their partner. If they do so, their sentence will be lighter. Each of the two prisoners has two options: either to cooperate with their partner or to defect. Thus, the situation can result in four different outcomes, as explained below.

- If none of them defects, the police only have enough evidence for minor crimes and they each get a three-year sentence.
- If either of them cooperates while the other defects, the betrayer only gets a one-year sentence, while the other is considered the mastermind and gets a seven-year sentence.
- If they both defect, they are both considered equally responsible for the worst crimes and get five-year sentences.

The game is represented in Figure 4.1. We consider a utility function where each year of sentence counts as minus 1 utility.

Prisoner 1 \ Prisoner 2	Coop	Defect
Coop	**-3,-3**	**-1,-7**
Defect	**-7,-1**	**-5,-5**

Table 4.1: Prisoner's Dilemma

The above game can have multiple variations with different approaches for its analysis. For example, if we consider the decision of the first player known when the second one chooses, the problem might seem trivial. However, the game becomes complicated when both players are in isolation and neither of them knows what the other will decide. Other variations might suggest that the players have some time to discuss before making their decision, or they have committed multiple crimes in the past and have been presented with this dilemma several times before.

4.2 Normal Form

Game theory models different scenarios with different forms of games. The representation of Figure 4.1 shows a Normal Form game. Normal Form is the simplest way to view a game, where states are presented as blocks of an $n \times n$ table, with each dimension presenting the available actions of each player.

The formal definition of Normal Form games is the following

Definition 4.2.1. *(Normal-Form game) A (finite, n-person) normal-form game is a tuple* (N, A, u)*, where:*

- N *is a finite set of n players, indexed by i;*
- $A = A_1 \times \cdots \times A_n$*, where* A_i *is a finite set of actions available to player I.*
  Each vector a = $(a_1, \ldots, a_n) \in A$ *is called an action profile;*
- $u = (u_1, \ldots, u_n)$ *where* $u_i : A \mapsto \mathbb{R}$ *is a real-valued utility (or payoff) function for player I.*

4.3 Strategies and Equilibria

As we've seen, each player has an available set of actions they can play. However, each player also has a strategy they will eventually follow. Strategies and actions are not the same thing.

The strategy of a player is the decision they make out of all their possible actions. A strategy might be a single action the player chooses to play. This is called a pure strategy and it is the kind of strategy that we'll work with in our analysis. A set containing a pure strategy for each player of the game is called a pure-strategy profile.

However, a player may randomize over several actions instead of picking one deterministically. A strategy might include multiple actions $a_i \in A$, each selected with a probability p_i, where $\sum p_i = 1$. These are called mixed strategies. A mixed-strategy profile consists of one mixed strategy for every player of the game.

In a game with two or more players, picking a strategy can be very complex as the outcome will be decided by the combination of the strategies of every player. However, for a given set of chosen adversary strategies, the selection of a player becomes a matter of selecting the best outcome.

Considering a player i, we denote by $s_{-i} = (s_1, \dots, s_{i-1}, s_{i+1}, \dots, s_n)$ a strategy profile without the strategy of player i, so $s = (s_i, s_{-i})$. The definition of i's best response to s_{-i} is the following

Definition 4.3.1. *(Best Response) Player i's best response to the strategy profile s_{-i} is a mixed strategy $s_i^* \in S_i$ such that $u_i(s_i^*, s_{-i}) \geq u_i(s_i, s_{-i})$ for all strategies $s_i \in S_i$, where S_i is the set of all possible mixed strategies of player i.*

Having defined Best Response, we can now present and formally define the most central notion of game theory, Nash Equilibrium.

Game theory analysis has always been trying to find the outcomes of different games that display some kind of balance on the players' desires. The players are satisfied with the utilities of those outcomes compared to the alternatives. The different kinds of stability these outcomes display are captured by different equilibria notions, with the most basic being the Nash Equilibrium. An outcome of a game is a Nash Equilibrium when no player can achieve a greater utility by choosing another strategy, provided that the rest of the players keep the same strategy.

Formally we have:

Definition 4.3.2. *A strategy profile $s = (s_1, \dots, s_n)$ is a Nash equilibrium if, for all agents i, s_i is a best response to s_{-i}.*

4.4 Perfect Information Extensive-Form Games

For the game-theoretic analysis of our protocol, we use the Extensive-form with chance moves. Extensive-form captures the temporal nature of the game played, where the players take turns making actions rather than selecting their actions to be played concurrently. Each action taken leads to a different state of the game till the end of the game when pay-offs are calculated. More specifically, our game follows the perfect-information extensive-form model, meaning players know all the previous actions played by opponents and so they know the specific state they're in.

Definition 4.4.1. *(Perfect-information game) A (finite) perfect-information game (in extensive form) is a tuple G = (N, A, H, Z, χ, ρ, σ, u), where:*

- *N is a set of n players;*
- *A is a (single) set of actions;*
- *H is a set of nonterminal choice nodes;*
- *Z is a set of terminal nodes, disjoint from H ;*
- $\chi : H \mapsto 2^A$ *is the action function, which assigns to each choice node a set of possible actions;*
- $\rho : H \mapsto N$ *is the player function, which assigns to each nonterminal node a player* $i \in N$ *who chooses an action at that node;*
- $\sigma : H \times A \mapsto H \cup Z$ *is the successor function, which maps a choice node and an action to a new choice node or terminal node such that for all* $h_1, h_2 \in H$ *and* $a_1, a_2 \in A$, *if* $\sigma(h_1, a_1) = \sigma(h_2, a_2)$ *then* $h_1 = h_2$ *and* $a_1 = a_2$ *;*
- $u = (u_1, \ldots, u_n)$, *where* $u_i : Z \mapsto \mathbb{R}$ *is a real-valued utility function for player i on the terminal nodes Z.*

The best representation of these games is in the form of a tree, where the states of the game are shown as nodes and the players' actions are edges that lead to new states. Each state represented by a node is uniquely identified as the sequence of choices leading from the root-node to it. Furthermore, each non-terminal node is the root of a sub-tree that can be viewed as a different game. Each such game is called a Subgame of the original game.

4.4.1 Family Example

Maria and Jack are mother and son. Maria needs to go to the grocery store and can either take Jack with her or leave him alone in the house. If he stays at home, Jack can either watch TV or read a book, but he'd rather go with his mom. If he goes, he can either behave or go wild. He finds it really difficult to behave, though, and he'd rather be himself. Maria wants to go with her son, but if he doesn't behave, she knows she will have a difficult time, so she warns him that if he doesn't, they will return home at once and he will be punished, even though she is terribly sorry when that happens. The game tree is shown in Figure 4.1

Extensive form games can be converted to normal form ones and Nash equilibria are well defined for those games as well. In the example above, both players have to make two choices and so they have the following strategies.

- Maria's choices are: take or leave Jack and punish or do not punish Jack
  (we will mark them as TJ or LJ and P or NP for convenience)
  So her strategies are (TJ,P),(LJ,P),(TJ,NP),(LJ,NP)

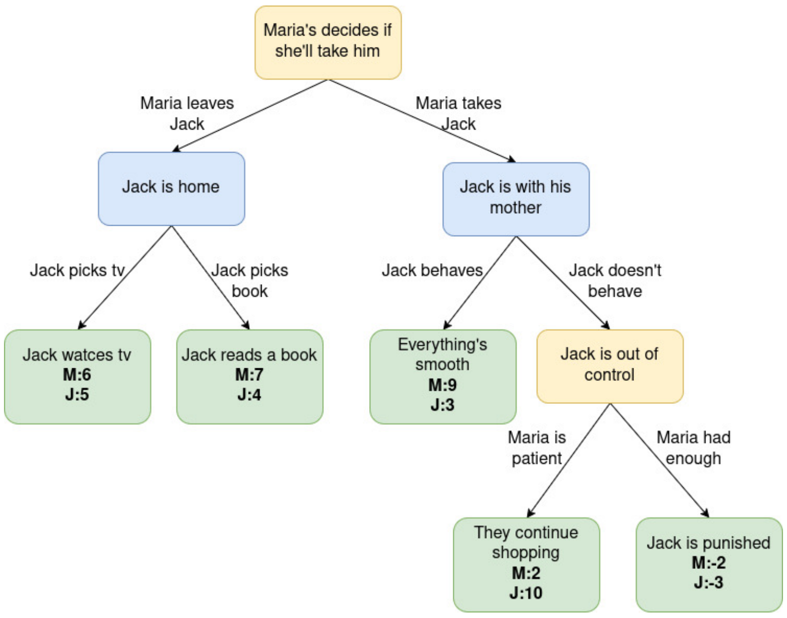


Figure 4.1: Family Example

- Jack's choices are: watch TV or read a book and behave or do not behave
  (we will mark them as TV or RB and B or NB)
  So his strategies are (TV,B),(RB,B),(TV,NB),(RB,NB)

Note that for a complete strategy, we have to include combinations of choices that are impossible to happen in the same scenario. For example, if Maria leaves Jack, there is no way she will punish him; however, strategy (LJ, P) has to be included.
Using these strategies, we create the corresponding normal form game, shown in Figure 4.2.

Examining the normal-form presentation of the game, we can finally deduce its Nash equilibria. However, having done this, we make an observation. Nash equilibria of extensive form games are a relatively weak notion. In the above example, outcome (TJ,P , TV,B) is a Nash equilibrium. Jack is discouraged by Maria's warning, as he knows that if he misbehaves, Maria's strategy is to punish him. However, the sequential nature of the game introduces a problem as Jack can consider Maria's warning a bluff and misbehave anyway. If that happens and the game reaches the final subtree, Maria's choice to punish Jack is no longer a best response as it leads to the worst outcome for both of them.

4.4.2 Subgame Perfect Equilibrium

To tackle this problem, a different equilibrium notion is used, called subgame-perfect equilibrium. To formally define it, we first define the Subgame notion.

Definition 4.4.2. *(Subgame) Given a perfect-information extensive-form game G, the Subgame of G rooted at node h is the restriction of G to the descendants of h. The set of Subgames of G consists*

Jack / Maria	TV,B	RB,B	TV,NB	RB,NB
TJ,NP	**9,3**	**9,3**	**2,10**	**2,10**
LJ,NP	**6,5**	**7,4**	**6,5**	**7,4**
TJ,P	**9,3**	**9,3**	**-2,-3**	**-2,-3**
LJ,P	**6,5**	**7,4**	**6,5**	**7,4**

Table 4.2: Normal-Form of Family Example

of all the Subgames of G rooted at some node in G.

Finally, we have the Subgame Perfect Equilibrium

Definition 4.4.3. *(Subgame Perfect Equilibrium) The Subgame Perfect Equilibria (SPE) of a game G are all strategy profiles s such that for any subgame G' of G, the restriction of s to G' is a Nash equilibrium of G'.*

Subgame Perfect Equilibrium (SPE) is not only a stricter notion, representing better real-world scenarios, but is significantly easier to compute as well. Backward Induction is the core principle used to find an SPE. The idea is to find the Nash equilibria for the bottom subtrees and follow the way up to the root of the game tree. This procedure follows the steps of a depth-first traverse and thus takes linear time in the size of the game tree.

Algorithm 1: Backward Induction

```
Input: $node\ h$
Output: $u(h)$                                          ▷ The utility vertex of players
Function Backward Induction($h$):
    if $h \in Z$ then
        return $u(h)$                                    ▷ h is a terminal node
    end
    $best\_util \longleftarrow -\infty$
    foreach $\alpha \in \chi(h)$ do
        $util\_at\_child \longleftarrow$ Backward Induction($\sigma(h, \alpha)$)
        if $util\_at\_child_{\rho(h)} > best\_util_{\rho(h)}$ then
            $best\_util \longleftarrow util\_at\_child$
        end
    end
    return $best\_util$
End Function
```

In Figure 4.2, we see how backward induction works in the previous example.

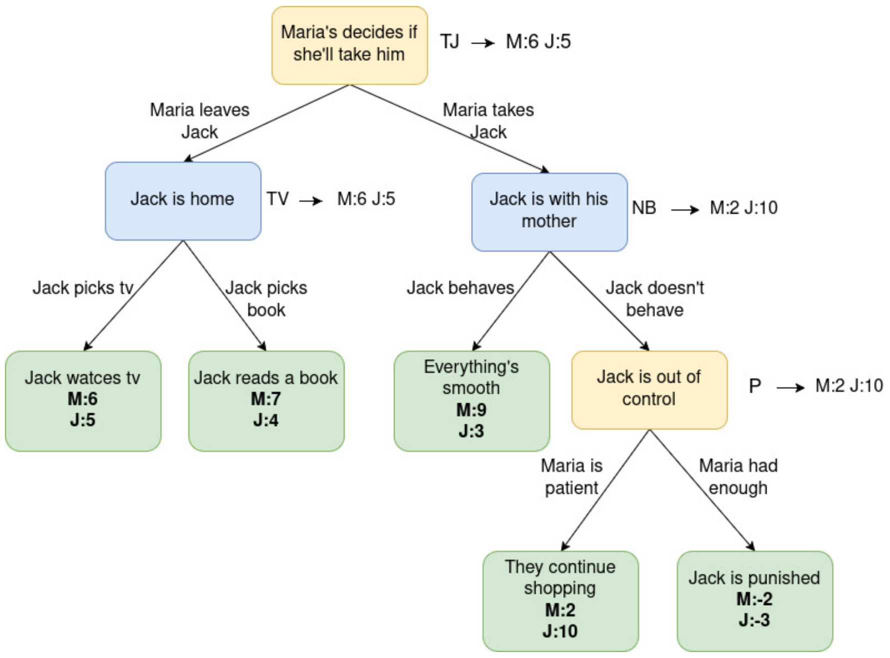


Figure 4.2: Backwards Induction in Family Example

Beginning the process from the first bottom subtree, "Jack is home", where Jack has to pick between the actions TV and RB. This trivial game has the obvious Nash equilibrium of Jack picking TV as it brings him the larger utility, 5 instead of 4. Following similar logic, the subtree "Jack is out of control" is a Nash equilibrium when Maria picks action NP instead of P. We can continue with the subtree "Jack is with his mother", where Jack picks one of the actions B or NB, where B finishes the game and NB leads to Maria's choice. Note that we already know Maria's choice if Jack plays NB, so given Maria's NP action Jack's best response is NB that gives him utility 10 instead of 3 and the Subgame has the equilibrium (NB,NP). We conclude by considering the whole game, now knowing Jack's strategy is (TV, NB). It is easy to see that Maria's best pick in the choice LJ or TJ is LJ, given Jack's strategy, as it gives her utility 6 instead of 2. Finally, the Family game example has the Subgame Perfect Equilibrium(LJ,NP , TV,NB).

4.5 Adding chance moves.

Real-world scenarios often depend on external conditions, independent of the players' actions. We model this by viewing external conditions as a third player having zero utility for all outcomes. This player chooses actions randomly with probabilities derived from the probabilities of the actual external conditions.

We modify the Perfect-information game definition to include chance moves.

Definition 4.5.1. *(Perfect-information game with chance moves) A (finite) perfect-information game with chance moves (in extensive form) is a tuple G = (N, A, H, C, Z, χ, ρ, σ, p, u), where:*

- *N is a set of n players;*
- *A is a (single) set of actions;*
- *H is a set of nonterminal choice nodes, non-chance choice nodes;*
- *C is a set of chance choice nodes, disjoint from H;*
- *Z is a set of terminal nodes, disjoint from H and C;*
- *$\chi : H \cup C \mapsto 2^A$ is the action function, which assigns to each choice node a set of possible actions;*
- *$\rho : H \mapsto N$ is the player function, which assigns to each nonterminal, non-chance node a player $i \in N$ who chooses an action at that node;*
- *$\sigma : (H \cup C) \times A \mapsto H \cup C \cup Z$ is the successor function, which maps a choice node and an action to a new choice node or terminal node such that for all $h_1, h_2 \in H$ and $a_1, a_2 \in A$, if $\sigma(h_1, a_1) = \sigma(h_2, a_2)$ then $h_1 = h_2$ and $a_1 = a_2$;*

- $p : C \times A \mapsto \mathbb{R}$ *is the probability function, which maps a chance choice node and an action to the probability of this action being chosen in the node. For all* $c \in C$ *and* $a \in A,\ p(c, a) \in [0, 1]$

- $u = (u_1, \ldots, u_n)$*, where* $u_i : Z \mapsto \mathbb{R}$ *is a real-valued utility function for player i on the terminal nodes Z.*

To determine the equilibrium of the game, taking into account this chance move, we use the Expectimax algorithm. The Expectimax algorithm is similar to Backward Induction, with the only difference in the states of chance moves. In those moves, we use the Expected Value of the utilities of the child states and consider those as the utilities of players in that state. To calculate the Expected Value, we use the weighted sum of the utilities of the children, with weights being the probabilities of those states being chosen by the chance move.

Algorithm 2: Expectimax

```
Input: $node\ h$
Output: $u(h)$                                          ▷ The utility vertex of players
Function Expectimax($h$):
    if $h \in Z$ then
        return $u(h)$                                   ▷ h is a terminal node
    end
    if $h \in C$ then
        return $\sum_{\alpha \in \chi(h)}$ Expectimax($\sigma(h, \alpha)$) $\cdot\, p(h, \alpha)$     ▷ h is a chance node
    end
    $best\_util \longleftarrow -\infty$
    foreach $\alpha \in \chi(h)$ do
        $util\_at\_child \longleftarrow$ Expectimax($\sigma(h, \alpha)$)
        if $util\_at\_child_{\rho(h)} > best\_util_{\rho(h)}$ then
            $best\_util \longleftarrow util\_at\_child$
        end
    end
    return $best\_util$
End Function
```

4.6 Imperfect Information Games

Our main result in this work is perfectly modeled by an extensive-form game of perfect information. However, our work continues to include a more complex extension, with the analysis of a generalized game. This game includes states where the player choosing the action has no information on the action previously chosen by their adversary. As a result, they know they are located in a set of states without the ability to distinguish between the states of the set and thus not knowing the exact state they play. The set of those states is called an information set. All states in an information set have the same player choosing an action and the same set of actions available for that player.

A game containing thi kind of states is called imperfect-information game and is formally defined as follows

Definition 4.6.1. *(Imperfect-information game) An imperfect-information game (in extensive form) is a tuple $(N, A, H, Z, \chi, \rho, \sigma, u, I)$, where:*

- *$(N, A, H, Z, \chi, \rho, \sigma, u)$ is a perfect-form game; and*
- *$I = (I_1, \ldots, I_n)$, where $I_i = (I_{i,1}, \ldots, I_{i,k_i})$ is a set of equivalence classes on (i.e., a partition of) $h \in H : \rho(h) = i$ with the property that $\chi(h) = \chi(h')$ and $\rho(h) = \rho(h')$ whenever there exists a j for which $h \in I_{i,j}$ and $h' \in I_{i,j}$.*

Having defined the imperfect-information extensive-form games, we need to consider the equilibrium notion fit for analyzing such a game. Although the SPE notion used for extensive-form games is the best fit for our analysis, there have to be modifications in its definition to suit imperfect-information games. More specifically we need to redefine the subgame notion used in the equilibrium. The generalized subgame notion, taken by [58], is the following

Definition 4.6.2. *(Subgame on Imperfect-information games) A Subgame of an extensive game is a subtree of the game tree that includes all information sets containing a node of the subtree.*

Given the new, general definition of the Subgame notion, the Subgame Perfect Equilibrium notion of a game remains the same. That is, a strategy profile is a Subgame Perfect Equilibrium of a game, when every Subgame of the game is a Nash Equilibrium.

Chapter 5

Predictable Service Protocols

5.1 Model

Dynamic/Fixed Fees Ledgers. Our work compares dynamic to fixed fees in a PoS blockchain environment. Users create transactions and submit them to the network to be included in the public ledger. Block producers gather the transactions submitted by the users and batch them into blocks they append to an ever-growing chain.

The block production process is performed by a set of stakeholders. The stakeholders lock in an amount of funds, called stake, to be eligible for participation. Stakeholders maintain a mempool in which they receive transactions submitted to the network for inclusion in the ledger. When they create a block, they gather transactions from their mempool and batch them in a block they append to the ever-growing chain.[1] Our work also assumes smart contract functionality. Scripts can be executed in the blockchain environment, facilitating protocol creation and binding agreements between users.

Users are required to pay fees for the transactions they want to be included in the ledger. In fixed fees systems, the fee is deterministically calculated based on the size of the transaction and the resources it may need for its logic to be executed. In contrast, in dynamic fees, the fee is constantly adjusted as time passes, so its future price may be hard to predict.

During periods of congestion, the rate at which transactions reach a mempool is greater than the one at which they are included in the ledger. Thus, transactions form queues in the mempools and compete with each other for the scarce block space. Dynamic fees attempt to solve this problem by raising the fees needed for transaction inclusion, bringing the demand down to reasonable levels, so that the set of eligible transactions fits in the block. In this way, congestion translates to elevated prices where users can get their transactions included on time but at a much greater cost. On the other hand, fixed fees keep the cost at the same level even when the demand increases excessively. As a consequence, some users suffer great delays in transaction inclusion, as there is no way that the system can serve them.

[1] The transactions are typically prioritized in a FIFO manner by the stakeholder; however, this is not binding and block producers have complete freedom of the block space allocation in their slot. Therefore, they choose the transactions they prefer to be included in the block they create, provided that the necessary fee is paid. Not even block producers can omit paying the fee for the inclusion of a transaction.

The predictable service problem. Our work revolves around a user who wants to include some transaction tx in the ledger at some point in the future. We consider the moment tx is ready to be submitted as fixed and denote it by t_{ready}. She receives value v_{in} from the inclusion of tx, provided this will happen within a specific interval. The interval begins at the moment t_{ready} and ends at the moment denoted by $deadline$. When this moment passes, the user receives value v_{out} whether tx is included in the ledger or not, where $v_{out} \leq 0 < v_{in}$. Although the user knows well ahead of time that she will need blockspace allocation at the specific time, tx will only be ready for submission shortly before the desired time and thus it is susceptible to the delays or the high prices of the network, putting the User at risk.

The utility received by the user from tx when it gets submitted the usual way is the following

$$U = \begin{cases} v_{in} - F & \text{, tx included} \\ v_{not} & \text{, tx not included} \end{cases}$$

where F is the fee required to be paid for the tx inclusion.

As described earlier, when fixed fees systems are congested, transactions may suffer extended delays from the moment they are submitted till they get included in the ledger. Therefore, there exists a threshold on delay value, denoted by d_{small}= $deadline$- t_{ready}, over which a transaction submitted by the user without an agreement with a block producer does not get included in time, thus providing value v_{out}. Therefore, the delay can be either small $(d \leq d_{small})$ or large (d > d_{small}), depending on whether it is less or greater than the threshold, respectively. In other words, when delay is in the d_{small} range, the user can get tx included in time even with no contract. We denote with p_{small} the probability of the delay being small. Thus, a user trying to lower bound the expected utility needs to lower bound p_{small}.

In this work, we assume that future traffic after some time horizon is computationally unpredictable and its estimation is *unreliable.* Consequently, the estimation of both future delays in fixed fees and fee prices in dynamic fees is also unreliable. We thus make the following assumptions.

Assumption 1 (unpredictable delay). *In fixed fees systems, no user can bound* p_{small}.

As a result of Assumption 1, the only secure lower bound on utility is the non-positive value v_{out} derived assuming p_{small}=0.

On the other hand, when dynamic fees systems are congested, the fee price is elevated. Therefore, tx is included if, for the desired interval, the fee prices are below the utility v_{in} received by the user when tx is included. This time, a user trying to lower bound the expected utility needs to lower bound the fee price. Similarly to Assumption 1, the following assumption is made.

Assumption 2. *[unpredictable price] In dynamic fees systems, parties can only bound the fee price inside an interval (α, β). The price is sure to stay inside the interval; however, it can take any value in that range.* [2]

[2] In systems like Ethereum, the price adapts at an exponential rate from block to block. Given that in a few days'

As a result, the only secure lower bound on utility is the non-positive value v_{out}, assuming $v_{in} - F \leq v_{out} \leq 0$.

To tackle this uncertainty on the future utility, we study possible agreements between our user, hereinafter referred to as the User, and a stakeholder, hereinafter referred to as the Insurer. The Insurer is a stakeholder participating in the block production process and possibly a block producer in the User's desired time. This work analyzes two possible situations, one where the Insurer is guaranteed to be a block producer and one where the Insurer is the block producer under a probability p_{prod}. When the Insurer produces a block, she has absolute control over the transactions included in this block.

The two parties may communicate in any of the following two ways: (i) on-chain, by interacting through the blockchain by published transactions, and (ii) off-chain, by communicating through a regular pairwise channel. [3]

We formalize the problem of the User obtaining a guaranteed lower bound on expected utility through such agreements with the Insurer as follows:

Definition 5.1.1. (ϵ-predictable service*) A protocol* Π *between the user and the insurer offers* ϵ-predictable service *if:*

- Π *is an SPE*[4]*;*
- *the expected utility of a user who executes* Π *is* $(1 - \epsilon)v_{in}$.

The goal of this work is to investigate the existence of predictable service protocols in the setting described above.

The relevant notation introduced in this section is summarized in Table 5.1.

5.2 Predictable Service in Dynamic Fees Ledgers

The idea of using derivatives to secure future transactions is not new. In [69], a protocol similar to ours is introduced for a dynamic fees system. In this section, we show that in dynamic fees systems, it is impossible to solve the predictability problem defined earlier.

In dynamic fees, the insured value required for a predictable service is the price of the fee that shows unpredictable volatility. However, no party has any control over this volatility and any attempt to create such a protocol would just transfer the unpredictability issue from the User to the Insurer. When we consider the joint utility instead of the individual ones, we see that no protocol could provide any improvement.

Theorem 5.2.1. *In dynamic fees ledgers, no protocol guarantees* ϵ*-predictable service, for* $\epsilon < \frac{\beta}{v_{in}}$.

time thousands of blocks are created, the price is only theoretically bounded as it can get extraordinarily high.

[3] In practice, this channel could be established through social media, websites, etc..

[4] In simple words, no party can maximize its respective utility by deviating from the behaviour detected by the protocol. 4.4.2

t_{ready}	the moment tx is ready for submission and the start of the desired inclusion interval
$deadline$	the deadline after which the User receives no value by the inclusion of tx
v_{in}	the value received by the User if tx is included in the ledger before $deadline$
v_{out}	the value received by the User if $deadline$passes and tx is not included
p_{prod}	the probability of the Insurer being a block producer
d_{small}	the threshold on the delay over which a transaction submitted does not get included before $deadline$; is equal to $deadline$-t_{ready}
p_{small}	the probability of the delay being less than d_{small}

Table 5.1: Model parameters.

Proof. We begin by analyzing the joint utility of the User and the Insurer under an arbitrary protocol, trying to achieve predictable service. We shall ignore all funds transferred between the two parties, inside the system, and focus on external sources of values and costs of the system. There are two inevitable factors defining the joint utility, the same for any potential protocol.

- The transaction insured brings to the User, and thus the system, a value depending on whether it is included in time or not. Let it be v_{in} for the included transaction and v_{not} for the not included transaction.[5]
- The fee F, for the insured transaction to be included at the desired moment.

The resulting upper bound on the joint utility is given by the following expression stripped of specifications that may come with the corresponding details of any particular protocol is the following:

$$U_{joint} \leq v_{in} - F, \text{ if the transaction is included} \tag{5.1}$$
$$U_{joint} \leq v_{out}, \text{ if the transaction is not included} \tag{5.2}$$

According to the Assumption 2 made in the previous section, the only secure bound on fee price is β. Given that $U_{joint} \leq \max\{v_{in} - F, v_{out}\}$, it follows that the only secure upper bound for U_{joint} is $\max\{v_{in} - \beta, v_{out}\}$. For such a protocol to guarantee ϵ-predictable service, we need: a) the Insurer to have a non-negative utility[6] and b) the User to receive utility equal to

[5] It is possible that the User commits, upon initiating the protocol, that the transaction will be included. Therefore, v_{not} can have a negative value.

[6] If the Insurer has negative utility the protocol would not be an SPE and the Insurer would rather not follow the protocol.

$(1-\epsilon)v_{in}$. Given the worst case of $U_{joint} = v_{in} - \beta$, the User ends up with maximum utility equal to $v_{in} - \beta = (1 - \frac{\beta}{v_{in}})v_{in}$, as the Insurer has minimum utility of zero. Hence, the theorem holds.

$\square$

A reader might suggest that there are other sources of income in the joint utility. Indeed, we have the reward and the tip for the block-producer. However, the reward is always the same for the block-producer that gets it even if the transaction is not included, whether there is a contract or not. Thus, we consider it irrelevant to the exchange and we do not take it into account. The tips, on the other hand, are not irrelevant and someone might argue that the block-producer may leave the insured transaction out in favor of another transaction with a much greater tip, resulting in a different utility expression to the one above. Nevertheless, the tip is meant to be much smaller than the fee to actually make a difference. Furthermore, it might or might not be the case that this scenario happens, so the argument still stands and the utility might be negative.

In conclusion, such contracts in dynamic fees systems impose an inevitable risk. As described, β can get excessively large and thus, the lower bound $\frac{\beta}{v_{in}}$ for ϵ can be greater than 1, leading User to a negative utility.

LedgerHedger. The protocol of [69] assumes that the fee price follows a truncated normal distribution. This means the fee prices are bounded and have zero probability of exceeding a certain threshold. In the protocol, the Insurer is suggested to lock a collateral equal to or greater than this threshold to prevent her from deviations. While such a threshold theoretically exists, given that prices change at an exponential rate and Assumption 2, it is safe to assume that the collateral required to exceed the price surges would discourage any stakeholder from taking on the role of the Insurer.

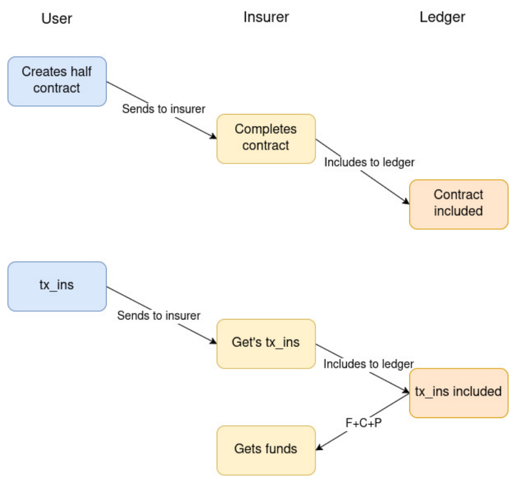
Next, we turn our attention to predictable service in fixed fees systems.

5.3 Predictable Service in Fixed Fees Ledgers

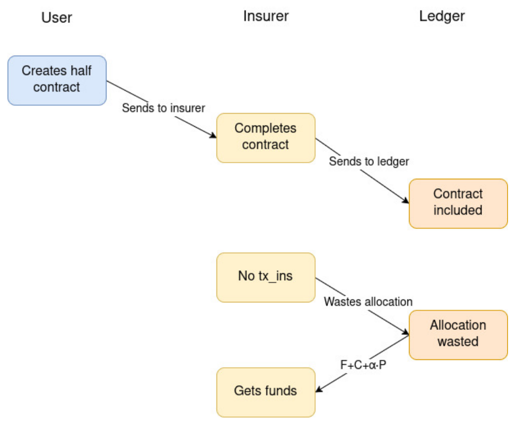
In this Chapter, we describe in detail our protocol. The protocol is initiated by a User having a future transaction tx_{ins}, she wants to be included at some point in the future and an Insurer, who in this case is assumed to be a block producer and thus possibly controls a space allocation at the desirable interval.

The protocol consists of two phases: (i) the creation of the contract between the user and the block-producer, and (ii) the execution of the contract. In the first phase, the user sends off-chain [7] to the block-producer a half-contract indicating its interest in including a transaction at a future interval and for a specific price. If the block-producer finds the contract appealing,

[7] We shall think of the first phase happening in a marketplace where Users can post their half-contracts to find an interested Insurer to complete and deploy them. The protocol can work in many other ways of off-chain communication but this is the one we'll use to describe it.



(a) Protocol followed



(b) Allocation Wasted

she completes it (by adding a collateral deposit) and posts it on-chain. In the second phase, the user sends to the block-producer the target transaction off-chain, and the block-producer includes in time to the blockchain to receive in turn the payment agreed by the contract. Next, we describe the protocol in detail.

Phase 1. The first phase begins with the User initiating the communication, by uploading off-chain a half contract. In the half contract, the User locks a sufficient amount of funds to cover for the fee of the future conclusion of the contract, denoted by F, as well as the price of the contract, denoted by P, i.e., the payment the Insurer is going to receive should the contract be fulfilled and tx_{ins} be included. Furthermore, the User specifies: (i) the collateral required by the Insurer to complete the contract, denoted by C, (ii) two intervals corresponding to Phase 1 and 2 of the contract intervals, (iii) the amount of blockchain resources needed by tx_{ins}, and (iv) the factor $\alpha \in (0,1)$ that indicates the reduced payment received by the Insurer should the User deviates. The User is responsible for covering the fee of the whole contract, denoted by F_C.

Next, after the Insurer receives the half contract, she decides whether to accept it by completing the deposit required, signing it, and submitting to the blockchain, or whether to reject it. If the interval, set by the User for the first phase, concludes with no insurer completing the contract, the half contract can no longer be completed. This completes the first phase of the protocol.

Phase 2. The interval set by the User in the contract defines when the second phase begins and ends. It is the interval in which tx_{ins} should be included for the User to take her full valuation.

The moment the User learns tx_{ins}, she send it off-chain to the Insurer. The Insurer then can proceed including tx_{ins} before Phase 2 ends. If this process is carried out successfully by both parties, the funds stored in the contract, i.e., F, C, P, become available to the Insurer. Note, that the fee paid by the Insurer for the inclusion of tx_{ins} is covered by F.

If the User does not provide a candidate transaction for inclusion, or provides a transaction that exceeds the resources specified, the Insurer has the option to waste the allocated resources by filling it with trash and receive some of the funds locked. In more detail, the Insurer receives from the contract a F, C, and $\alpha \cdot P$, where α is less than 1 and is a parameter of the initial contract. This functionality is provided to protect the Insurer from a faulty or malicious User who will ignore the contract, but also the partial payment secures that no malicious Insurer will choose this process instead of the desirable one.

If the Insurer ignores the contract and does not include tx_{ins} in the specified interval, then the User can retrieve her funds. More specifically, in this case, the User submits a transaction unlocking part of the funds stored in the contract, getting back F,P and F_C, and part of the collateral equal to the fee paid for the contract F_C. Thus, the User will have the complete *cost* returned[8]. This functionality is provided to protect the User from a faulty or malicious Insurer who will ignore the contract. Note that this may happen whether the User actually sent tx_{ins} or not, as there is no way the contract can verify what happened in the off-chain channel.

We require the collateral to be greater or equal to the fee of the future conclusion and the fee paid by the User for the contract, $C \geq F + F_C$. C will be returned to the Insurer if she follows the protocol to the end with no deviations. However, if the Insurer abandons the contract part of the collateral will be used to compensate the User for their spending. The Insurer then proceeds to submit the completed contract for inclusion in the ledger.

Details of fee calculation. The protocol involves several interactions with the ledger. These interactions happen in the form of a transaction, requiring fees to be included. The fees are calculated based on two factors: a) the size of the transaction depending on the inputs consumed and outputs produced by the transaction and b) the script they execute in a smart contract. As a result the fee for the possible interactions is calculated as follows

- **tx_{ins} Submission**: When tx_{ins} is submitted for inclusion as a simple transaction requires a specific fee we shall denote with F_{tx}
- **Contract Creation**: The creation of the smart contract used for the protocol is based on the inputs of the funds locked in it as well as the code defining its functionality. We denote this fee with F_C.
- **Contract Fulfillment**: The successful conclusion of the protocol happens when the Insurer fulfills the contract by including tx_{ins} in the ledger. The fee for this transaction is calculated as follows: a) The size of this transaction is based on the size of the contract it uses, its input accounting for the fee and its outputs, meaning the funds $F + C + P$ received by the Insurer. b)The script cost of the transaction results from the execution of the smart contract. c) Furthermore, the fee for this transaction includes the fee F_{tx} of tx_{ins} itself.

[8] **[Tsironis: There can be a greater compensation equal to $\beta \cdot C$ but it is not necessary for our analysis]**

symbol	meaning
P	the price payed by the User and stored in the contract to be received by the Insurer, provided tx_{ins} will be included in the ledger
F_{tx}	the fee required for the inclusion of tx_{ins}
F	the fee needed for the contract to be concluded
F_C	the fee needed for the contract to be included (covered by the User)
C	the collateral the Insurer has to lock in the contract to accept it. It will be released provided she follows the protocol; is greater or equal to $F + F_C$
$alloc$	the space allocation needed by tx_{ins}
$cost$	the total amount paid by the user, equal to $F + F_C + P$
$\alpha < 1$	the part of P the Insurer gets when $alloc$ is wasted

Table 5.2: Protocol parameters.

- **Allocation Wasted**: The functionality "Allocation Wasted" needs a fee that is calculated as follows: a) The size is based on the contract used, the input for the fee and the output $F + C + \alpha \cdot P$. b) The script cost results from the execution of the contract. c) The fee includes the fee for wasting $alloc$, which is equal that of tx_{ins}, so F_{tx}
- **Payment Retrieved**: The fee required for the functionality "Payment Retrieved" is calculated as follows: a) The size is based on the contract used, the input for the fee and the output $F + F_C$. b) The script cost results from the execution of the contract.

We can see that the contract has three possible conclusions: a) "Contract Fulfilled", b) "Allocation Wasted" and c) "Payment Retrieved". All the conclusions need roughly the same fees, we denote by F. "Payment Retrieved" needs less fees, as it does not include F_{tx}. However, for simplicity's sake we consider it to be equal to F as well. We have declared C to be larger than $F + F_C$ so it covers the fee of this conclusion and it does not alter the analysis as it is lost anyways in this conclusion.

We denote by $cost$ the total amount $F + F_C + P$ paid by the User.

Table 5.2 summarizes the notation presented above.

Chapter 6

Game Theory Analysis

6.1 Simple Game

In this section, we analyze the protocol of Section 5.3 by first describing a related extensive-form game and then showing that the protocol described is an SPE under a suitable assumption about the protocol parameters.

The game represents the second phase of the protocol. That is, the results of this section apply when the two parties have already created a contract. We do so to calculate the guaranteed utilities derived from the contract when it is successfully created. We will later compare the results to the expected utilities of the normal transaction submission.

6.1.1 Game description

The game describing our protocol follows the perfect-information extensive-form with chance moves model. This model captures the sequential nature of the game played by the User and Insurer, who take actions creating the different states of the game. The random distribution which determines the final delay on the network in the second phase interval is modeled as a third player whose choice is the result of the distribution. We shall refer to the third player as Nature. Nature obviously has no utility for any outcome of the game, so we will consider utility 0. It follows a mixed strategy with probabilities given by the distribution. Figure 6.1 presents the game tree of the second phase of our protocol.

We begin our analysis by identifying the utilities of players. User, as the initiator, has a strong utility for tx_{ins} to be included in the ledger, specifically in the second phase interval, while this utility declines when the inclusion is delayed beyond $deadline$. Therefore, we denote with v_{in} the User's utility when tx_{ins} gets included in the desired time and v_{out} when it's not. We denote by $u(\text{state})$ the vector consisting of the utilities of the two parties in a specific state, denoted by $u_U(\text{state})$ and $u_I(\text{state})$. We continue identifying the utilities and costs for both players.

The game can be split into two different subgames regarding whether a contract was created or not. We shall first analyze the subgame "Contract Completed" and prove that following the protocol is an SPE. Then we proceed to analyze the utilities of the subgame "No Creation" and when creating the contract is the optimal choice for the User and the Insurer.

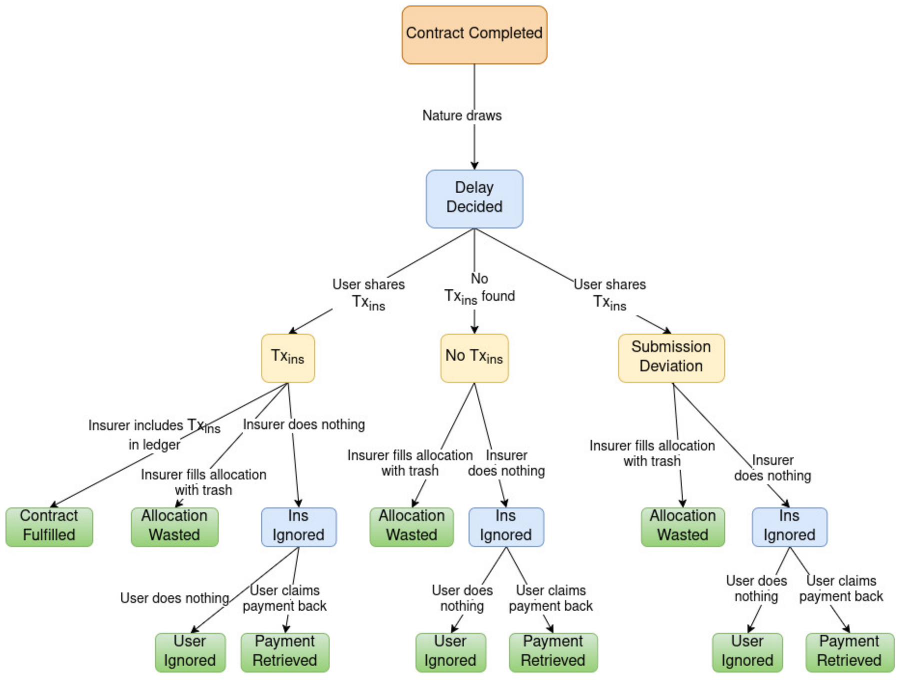


Figure 6.1: Contract Completed

6.1.2 Analysis

Theorem 6.1.1. *We prove that our protocol provides ϵ-predictable service, for* $\epsilon = \frac{F+F_C+P}{v_{in}}$

Proof. We break the subgame ”Contract Completed” into two subgames and analyze them one at a time.

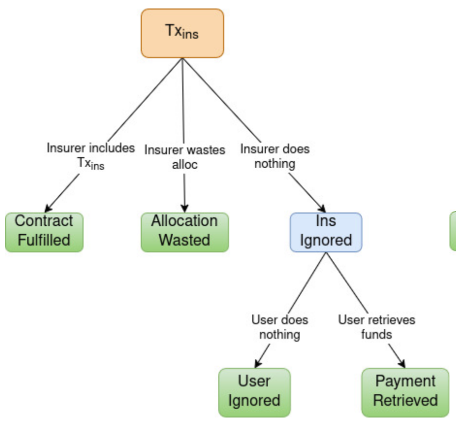
- The Subgame ”Delay Small” is played when a contract has been created and the delay is small enough so that tx_{ins} can be included in time by anyone.
- The Subgame ”Delay Large” is played when a contract has been created and the delay is so large that only a producer can include tx_{ins} in time.

The subgames ”Delay is small” and ”Delay is large” are broken down in turn into three subgames.

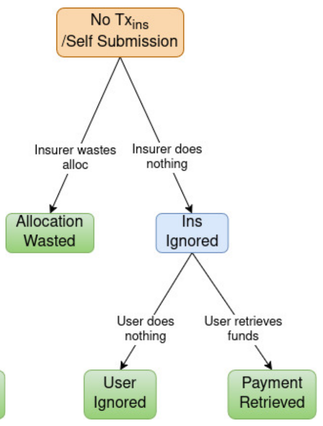
– tx_{ins} The User sends tx_{ins} to the Insurer for inclusion, as shown in Figure 6.2a

– No tx_{ins} The User sends nothing to the Insurer and waits for the $deadline$, as shown in Figure 6.2b

– Self Submission The User submits tx_{ins} instead of sending it to the Insurer, as shown in Figure 6.2b



(a) tx_{ins} sent



(b) No tx_{ins} sent

The subtrees ”No tx_{ins}” and ”Self Submission” have the exact actions available for the players in every state, but they differ in the utilities and thus are analyzed separately.

Claim 1. *The Subgame ”tx_{ins}” when delay is small is in SPE when the protocol is followed and the Insurer fulfills the contract. The Subgame is depicted in Figure 6.2a*

Proof of Claim. We use the Backwards Induction Algorithm 1 to determine the utilities of the Subgames, working from the bottom up and selecting the best response for the respective player.

- u_U(ContractFulfilled)$= v_{in} - cost$, u_I(ContractFulfilled)$= P$
  In outcome ”Contract Fulfilled”, the whole process is followed as designed, the contract was created, the user sent to the insurer tx_{ins} and the insurer included it in the ledger. The Insurer gets utility from the price of the contract paid by the User and so her utility is P. To find the user’s utility, we first note that tx_{ins} was included before *deadline* as the contract requires and the User’s utility is v_{in}minus the payment F+P and the fee for the contract F_C, that is $v_{in} - (F + F_C + P)$.

- u_U(Allocation Wasted) $= v_{out} - cost$, u_I(Allocation Wasted) $= \alpha \cdot P$
  In outcome ”Allocation Wasted”, the contract was created, but the Insurer deviated and did not include tx_{ins} in the block. The insurer chose to waste the space allocation, filling it with garbage, falsely declaring that they did not receive tx_{ins}. When doing so, the contract does not release the whole P but rather a portion of it. So, the Insurer’s utility is $\alpha \cdot P$. tx_{ins} in this case does not get in time in the ledger, so the User has paid *cost*, getting nothing in return. Therefore, the User’s utility is $v_{out} - (F + F_C + P)$.

- u_U(Payment Retrieved)$=v_{out}$, u_I(Payment Retrieved)$=-C$
  In outcome ”Payment Retrieved” *deadline* passed with no move by the Insurer, so the User retrieved all funds she had stored in the contract. [1] By ignoring the created contract,

[1] Note that this can happen whether tx_{ins} was provided by the user or not.

the Insurer loses the collateral submitted in the contract, so her utility is $-C$. The user does not get tx_{ins} included but retrieves $cost$. So, her utility is v_{out}.

- u_U(User Ignored) $= v_{out} - cost,$ u_I(User Ignored) $=-C$
  In outcome "User Ignored" $deadline$ passed, but the User did nothing. The Insurer's utility is the same as before, while the User's is the same as in the "Allocation Wasted" state.

- u_U(Ins Ignored) $= v_{out},$ u_I(Ins Ignored) $= -C$, since
  u_U(Payment Retrieved) > u_U(User Ignored)
  On node "Ins Ignored", the user chooses from states "User Ignored" and "Payment Retrieved", so picks "Payment Retrieved" as it gives a strictly greater utility. So, "Ins Ignored" has the same utilities as "Payment Retrieved".

So the claim holds as

$$u_I(\text{Contract Fulfilled}) > u_I(\text{Allocation Wasted}) > u_I(\text{Ins Ignored})$$

and the utilities of the Subgame in SPE are

$$u_U(tx_{ins}) = v_{in} - cost, \qquad u_I(tx_{ins}) = P$$

□

Claim 2. *The Subgame "No tx_{ins}" when delay is small is in SPE when the protocol is followed and the Insurer invokes "Allocation Wasted". The Subgame is depicted in Figure 6.2b*

Proof of Claim. The subgame "No tx_{ins}" differs from "tx_{ins}" only in missing the choice "Contract Fulfilled" of the Insurer. The rest states of the two games are the same. So for this subgame's analysis, we only have to consider the Insurer's choice between "Allocation Wasted" and "Ins Ignored".

So the claim holds as

$$u_I(\text{Allocation Wasted}) > u_I(\text{Ins Ignored})$$

and the utilities of the Subgame in SPE are

$$u_U(\text{No } tx_{ins}) = v_{out} - cost, \qquad u_I(\text{No } tx_{ins}) = \alpha \cdot P$$

□

Claim 3. *The Subgame "Self Submission", when delay is small,l is in SPE when the protocol is followed and the Insurer invokes "Allocation Wasted". The Subgame is depicted in Figure 6.2b*

Proof of Claim. Since the delay is small, when the user submits herself tx_{ins}, she manages to get it included in time, paying F. So, the utilities are almost the same as in "No tx_{ins}" except the User has $v_{in} - F_{tx}$ instead of v_{out}.

- u_U(Allocation Wasted) = $v_{in} - F_{tx} - cost$
  u_I(Allocation Wasted) = $\alpha \cdot P$
  User has already self-submitted tx_{ins} in time as delay is small, so they get $v_{in} - F_{tx}$ but they also lose $cost$.

- u_U(User Ignored) = $v_{in} - F_{tx} - cost$
  u_I(User Ignored) = $-C$

- u_U(Payment Retrieved) = $v_{in} - F_{tx}$
  u_I(Payment Retrieved) = $-C$

- u_U(Ins Ignored) = $v_{in} - F_{tx}$
  u_I(Ins Ignored) = $-C$
  As u_U(Payment Retrieved) > u_U(User Ignored)

So the claim holds as

$$u_I(\text{Allocation Wasted}) > u_I(\text{Ins Ignored})$$

like previously and the utilities of the Subgame in SPE are

$$u_U(\text{Self Submission } d_{small}) = v_{in} - F_{tx} - cost, \qquad u_I(\text{Self Submission } d_{small}) = \alpha \cdot P$$

□

Claim 4. *The Subgame "Delay is Small" is in SPE when the protocol is followed and the User sends $tx_{ins}$to the Insurer as the protocol dictates.*

Proof of Claim. The User chooses her action between "tx_{ins}", "No tx_{ins}" and "Self Submission". According to Claims 1, 2 and 3 we have

$$u_U(tx_{ins}) > u_U(\text{Self Submission}) > u_U(\text{No } tx_{ins})$$

So, the claim holds and

$$u_U(\text{Delay Small}) = v_{in} - cost, \qquad u_I(\text{Delay Small}) = P$$

□

Claim 5. *The Subgame "tx_{ins}" when delay is large is in SPE when the protocol is followed and the Insurer fulfills the contract. The Subgame is depicted in Figure 6.2a*

Proof of Claim. When the User follows the protocol with no deviations and sends tx_{ins} to the Insurer, there is no difference if the delay is large or small. The Insurer produces a block and thus can get tx_{ins} included in time regardless of the delay. So the claim holds for the same reason as Claim 1 and the utilities of the subgame are

$$u_U(tx_{ins}) = v_{in} - cost, \qquad u_I(tx_{ins}) = P$$

□

Claim 6. *The Subgame "No tx_{ins}" when delay is large is in SPE when the protocol is followed and the Insurer invokes "Allocation Wasted". The Subgame is depicted in Figure 6.2a*

Proof of Claim. As with "tx_{ins}", the Subgame "No tx_{ins}" is the same whether the delay is large or small, as there is no submission before $deadline$, and thus tx_{ins} will not be included in any case. So, the claim holds for the same reason as Claim 2 and the utilities of the Subgame are

$$u_U(\text{No } tx_{ins}) = v_{out} - cost, \qquad u_I(\text{No } tx_{ins}) = \alpha \cdot P$$

□

Claim 7. *The Subgame "Self Submission", when delay is large,e is in SPE when the protocol is followed and the Insurer invokes "Allocation Wasted". The Subgame is depicted in Figure 6.2a*

Proof of Claim. Since the delay is large, when the User submits tx_{ins} herself, it will not get included in time. This means that the User's utilities for this Subgame differ from the respective one when delay is small, in that they get v_{out}instead of v_{in}.

- u_U(Allocation Wasted) = $v_{out} - F_{tx} - cost$
  u_I(Allocation Wasted) = $\alpha \cdot P$

  User has already self-published tx_{ins} but it's not included in time as delay is large, so they get $v_{out} - F$ but they also lose $cost$.

- u_U(User Ignored) = $v_{out} - F_{tx} - cost$
  u_I(User Ignored) = $-C$

- u_U(Payment Retrieved) = $v_{out} - F_{tx}$
  u_I(Payment Retrieved) = $-C$

- u_U(Ins Ignored) = $v_{out} - F_{tx}$
  u_I(Ins Ignored) = $-C$
  As u_U(Payment Retrieved) > u_U(User Ignored)

So, the claim holds as

$$u_I(\text{Allocation Wasted}) > u_I(\text{Ins Ignored})$$

and the utilities of the Subgame in SPE are

$$u_U(\text{Self Submission } d_{large}) = v_{out} - F_{tx} - cost, \qquad u_I(\text{Self Submission } d_{large}) = \alpha \cdot P$$

□

Claim 8. *The Subgame "Delay is Large" is in SPE when the protocol is followed and the User sends tx_{ins} to the Insurer as the protocol dictates.*

Proof of Claim. As in Claim 4, the User chooses her action between "tx_{ins}", "No tx_{ins}" and "Self Submission". According to Claims 5, 6 and 7 we have

$$u_U(tx_{ins}) > u_U(\text{Self Submission}) > u_U(\text{No } tx_{ins})$$

So, the claim holds and

$$u_U(\text{Delay Large}) = v_{in} - cost, \qquad u_I(\text{Delay Large}) = P$$

□

We conclude that in both cases, according to the Claims 4 and 8, the two parties are incentivized to follow the protocol, so it is a Subgame Perfect Equilibrium. Furthermore, "Delay Small" and "Delay Large" utilities are the same and independent of the actual delay. Thus, "Contract Completed" utilities are independent of the delay as well and equal to

$$\boxed{\begin{aligned} u_U(\text{Contract Completed}) &= v_{in} - cost && (6.1)\\ u_I(\text{Contract Completed}) &= P && (6.2)\end{aligned}}$$

We have proven that the game is an SPE and the User gets guaranteed utility $v_{in} - cost = v_{in} - (F + F_C + P)$. Thus, Theorem 6.1.1 holds, as

$$\begin{aligned} v_{in} - (F + F_C + P) &= (1 - \epsilon)v_{in} \\ 1 - \frac{F + F_C + P}{v_{in}} &= 1 - \epsilon \\ \epsilon &= \frac{F + F_C + P}{v_{in}} \end{aligned}$$

□

6.2 Extended Game

6.2.1 Different Epoch Analysis

Up until now, we took for granted that the contract is created in the same epoch as the desired time of tx_{ins} inclusion. That means that the Insurer knew whether she would be the block producer at said time and could decide, respectively, to accept the contract or not. But what happens when the inclusion time is in a different epoch than the contract creation? The insurer can only make a prediction that she will be a block producer and a risk is introduced as there is a possibility (large or small) of the prediction being mistaken. We denote with p_{prod} the probability of the Insurer being a block producer.

We consider a new chance move to express the allocation of block creation duties. This allocation happens in the beginning of the epoch, which contains the time of inclusion. However, the User is not notified about the outcome, as only the Insurer knows whether they will be creating a block and when. To model this, we consider the new chance move to happen after the User decides to send tx_{ins} for inclusion or not. So the subtrees after this decision are split into two cases, one with the Insurer producing a block and one where they don't. Apart from that, the new game is similar to the previous one. The new version of the game is shown in Figure 6.3.

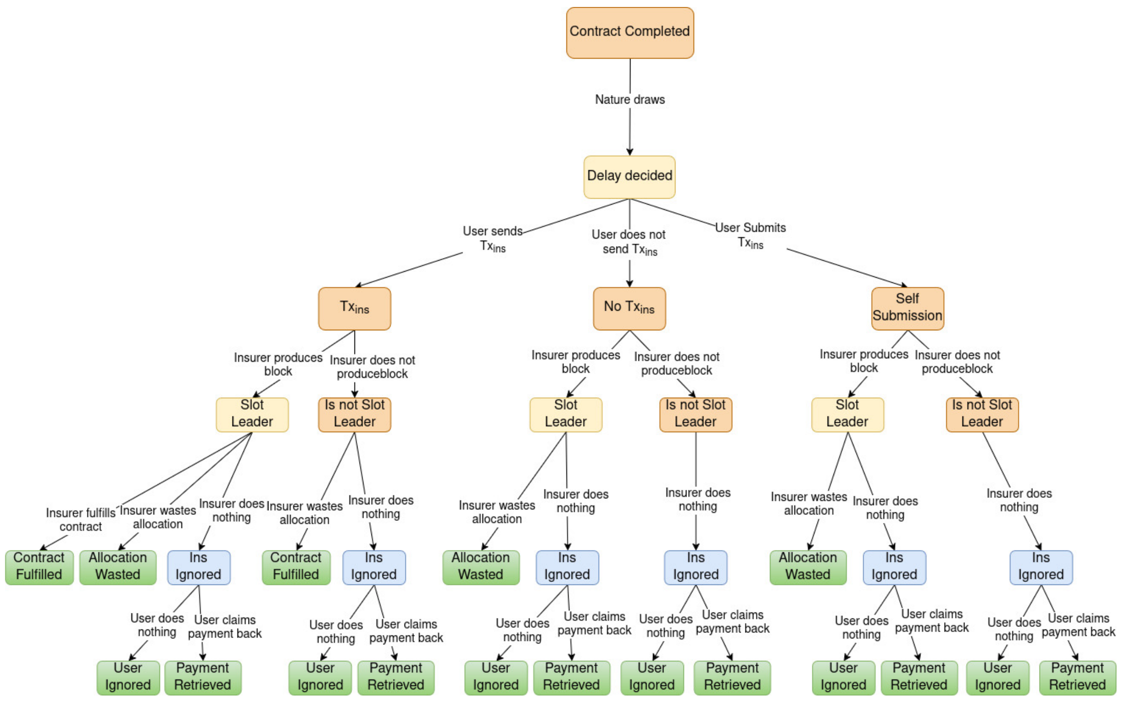


Figure 6.3: Extended Game

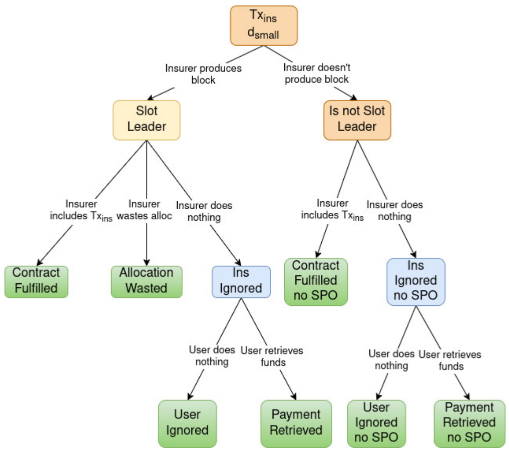
We denote by $u'(\text{state})$ the vector consisting of the utilities of the new game denoted by $u'_U(\text{state})$ and $u'_I(\text{state})$. Following simple game analysis, we are going to break the whole game down and work with its subgames before recomposing them and deriving the final results. We first focus on the subgame where a contract is created, proving the following theorem.

Theorem 6.2.1. *We prove that, provided $P < \frac{F_{tx}}{1-p_{prod}} - F - F_C$ our protocol provides ϵ-predictable service, for*

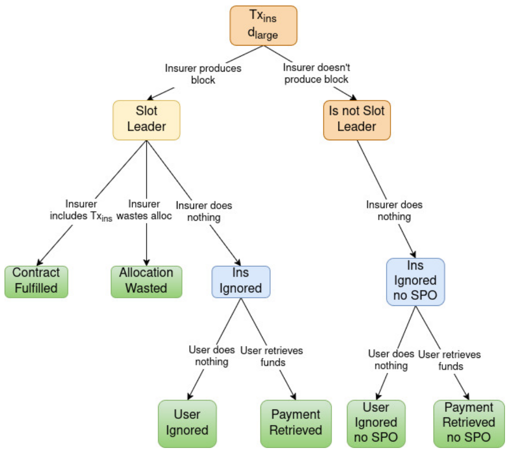
$$\epsilon = 1 - p_{prod} - \frac{p_{prod} \cdot (F + F_C + P) + (1 - p_{prod}) \cdot v_{out}}{v_{in}}$$

Proof. Just like in the simple game, we analyze the two subgames "Delay Small" and "Delay Large". Both of them differ from the ones in the simple game in that they include one more chance move, determining whether the Insurer produces a block in the specified time or not. Again, those subgames are broken down further into three

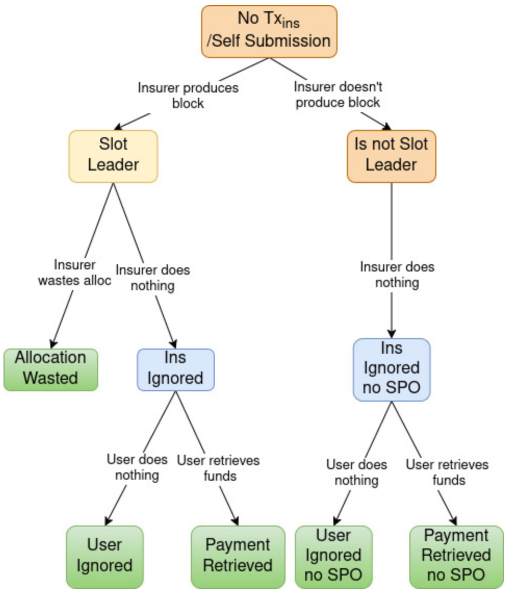
- tx_{ins}sent



(a) $tx_{ins}$with small delay



(b) $tx_{ins}$with large delay



(c) No Contract/Self Submission

- No tx_{ins}sent
- Self Submission

All three subgames differ from the ones in the simple game in that they include one more chance move, determining whether the Insurer produces a block in the specified time or not.

Claim 9. *The Subgame "tx_{ins}", when delay is small, is in SPE when the protocol is followed and the Insurer fulfills the contract. The Subgame is depicted in Figure 6.4a*

Proof of Claim. In this Subgame, the User follows the protocol as intended and sends tx_{ins} to the insurer. As the delay is small, the Insurer can still carry out the protocol even if they don't produce a block. Thus, the claim holds for the same reason as Claim 1 and the payoffs are the exact same as in the previous game.

$$u'_U(tx_{ins}) = v_{in} - cost, \qquad u'_I(tx_{ins}) = P$$

□

Claim 10. *The Subgame "No tx_{ins}" when delay is small is in SPE when the protocol is followed and the Insurer invokes "Allocation Wasted". The Subgame is depicted in Figure 6.4c.*

Proof of Claim. In this Subgame, the User does not send tx_{ins} and waits for $deadline$ to possibly receive the reimbursement.

- u'(Is Slot Leader No tx_{ins}) = u(No tx_{ins})
  When the Insurer produces a block, the utilities are the same as in the previous game.
- u'(Ins Ignored no Slot Leader) = u(Ins Ignored)
  The state "Ins Ignored" has the same utilities as the Insurer does nothing.
- u'(Is not Slot Leader No tx_{ins}) = u'(Ins Ignored no Slot Leader)
  The insurer has no option of "Allocation Wasted" when they do not produce a block and the only option is to do nothing. Thus, the utilities are the same as in "Ins Ignored no Slot Leader".
- u'_U(No tx_{ins}) = $p_{prod} \cdot (v_{out} - cost) + (1 - p_{prod}) \cdot v_{out}$
  u'_I(No tx_{ins}) = $p_{prod} \cdot \alpha \cdot P - (1 - p_{prod}) \cdot C$
  The expected utility of "No tx_{ins}" is calculated as explained previously by the two possible options of the chance move.

So the claim holds and the resulting utilities of the Subgame in SPE are

$$u'_U(\text{No } tx_{ins}) = p_{prod} \cdot (v_{out} - cost) + (1 - p_{prod}) \cdot v_{out},$$
$$u'_I(\text{No } tx_{ins}) = p_{prod} \cdot \alpha \cdot P - (1 - p_{prod}) \cdot C$$

□

Claim 11. *The Subgame "Self Submission", when delay is small, is in SPE when the protocol is followed and the Insurer invokes "Allocation Wasted". The Subgame is depicted in Figure 6.4c.*

Proof of Claim. In this Subgame, shown in Figure 6.4c, the User submits tx_{ins} and sends nothing to the Insurer. As the delay is small, tx_{ins} is included in time and the User gets v_{in}but pays F_{tx} for the inclusion.

- u'(Is Slot Leader) = u(Self Submission d_{small})
  When the Insurer produces a block, the utilities are the same as in the previous game as derived in Claim 3.

- u'_U(Payment Retrieved) = $v_{in} - F_{tx}$, u'_I(Payment Retrieved) = $-C$
  The User retrieves the funds of the contract and has the utility gained from self inclusion. The Insurer loses the collateral as they ignore the contract.

- u'_U(User Ignored) = $v_{in} - F_{tx} - cost$, u'_I(User Ignored) = $-C$
  The User does not retrieve the funds.

- u'_U(Ins Ignored) = $v_{in} - F_{tx}$, u'_I(Ins Ignored) = $-C$
  The obvious best move for the User is to retrieve the funds and the utilities are derived accordingly.

- u'_U(Is not Slot Leader) = $v_{in} - F_{tx}$, u'_I(Is not Slot Leader) = $-C$
  The utilities are the same as in "Ins Ignored" because the Insurer has no different choice.

- u'_U(Self Submission) = $p_{prod}\cdot(v_{in}-F_{tx}-cost)+(1-p_{prod})\cdot(v_{in}-F_{tx})$ u'_I(Self Submission) = $p_{prod} \cdot \alpha \cdot P - (1 - p_{prod}) \cdot C$ Once more, we calculate the expected utilities from the chance move.

So the claim holds and the resulting utilities of the Subgame in SPE are

$$\begin{aligned} u'_U(\text{Self Submission}) &= p_{prod} \cdot (v_{in} - F_{tx} - cost) + (1 - p_{prod}) \cdot (v_{in} - F_{tx}) \\ u'_I(\text{Self Submission}) &= p_{prod} \cdot \alpha \cdot P - (1 - p_{prod}) \cdot C \end{aligned}$$

□

Claim 12. *The Subgame "Delay is Small" is in SPE when the protocol is followed and the User sends $tx_{ins}$to the Insurer as the protocol dictates, provided that $P < \frac{F_{tx}}{1-p_{prod}} - F - F_C$.*

Proof of Claim. To find the SPE of the whole Subgame we need to find the best move for the User at the initial state "Delay Small. We have to compare the User's resulting utility in each of the three possible actions as derived in Claims 9, 10 and 11

$$\begin{aligned} u'_U(tx_{ins}) &= v_{in} - cost \\ u'_U(\text{No } tx_{ins}) &= p_{prod} \cdot (v_{out} - cost) + (1 - p_{prod}) \cdot v_{out} \\ u'_U(\text{Self Publish}) &= p_{prod} \cdot (v_{in} - F_{tx} - cost) + (1 - p_{prod}) \cdot (v_{in} - F_{tx}) \end{aligned}$$

We have established that $v_{in} - F > v_{out}$ or else the User would not bother including tx_{ins} much more, creating the contract. Thus, $u'_U(tx_{ins})$ > u'_U(No tx_{ins}). However, it is not clear which of the other two utilities is greater. For our contract to be an equilibrium, we want the User to have no incentive in deviating; in other words, we want $u'_U(tx_{ins})$ > u'_U(Self Publish).

$$\begin{aligned}
u'_U(tx_{ins}) &> u'_U(\text{Self Publish}) \\
v_{in} - cost &> p_{prod} \cdot (v_{in} - F_{tx} - cost) + (1 - p_{prod}) \cdot (v_{in} - F_{tx}) \\
v_{in} - cost &> v_{in} - F_{tx} - p_{prod} \cdot cost \\
p_{prod} \cdot cost &> cost - F_{tx} \\
p_{prod} &> \frac{cost - F_{tx}}{cost} \\
\text{or } F_{tx} &> (1 - p_{prod}) \cdot cost \\
\frac{F_{tx}}{1 - p_{prod}} &> cost \\
\frac{F_{tx}}{1 - p_{prod}} &> F + F_C + P \\
\frac{F_{tx}}{1 - p_{prod}} - F - F_C &> P \\
P &< \frac{F_{tx}}{1 - p_{prod}} - F - F_C
\end{aligned}$$

So, the claim holds, as provided the assumption we have $u'_U(tx_{ins})$ > u'_U(Self Publish) and the utilities of the Subgame in SPE are

$$\begin{aligned}
u'_U(\text{delay small}) &= v_{in} - cost \\
u'_I(\text{delay small}) &= P
\end{aligned}$$

□

Claim 13. *The Subgame "tx_{ins}" when delay is large is in SPE when the protocol is followed and the Insurer fulfills the contract. The Subgame is depicted in Figure 6.4b*

Proof of Claim. In this subgame, the user follows the protocol and sends tx_{ins} to the Insurer for inclusion.

- u'(Is Slot Leader) = $u(tx_{ins})$
  When the Insurer produces a block, the utilities are the same as in the previous game shown in Claim 1.
- u'_U(Is not Slot Leader) = u_U(Ins Ignored)
  u'_I(Is not Slot Leader) = $-C$
  The delay is large, so if the Insurer does not produce a block, there is no time to fulfill the contract by including tx_{ins}. Thus, when the Insurer does not produce a block, the utilities are the same as if the User never sent tx_{ins}as it was analyzed in Claim 10.
- $u'_U(tx_{ins})$ = $p_{prod} \cdot (v_{in} - cost) + (1 - p_{prod}) \cdot v_{out}$
  $u'_I(tx_{ins})$ = $p_{prod} \cdot P - (1 - p_{prod}) \cdot C$
  We calculate the expected utility of the chance move.

So, the claim holds and the resulting utilities of the Subgame in SPE are

$$\begin{aligned}\text{So, } u'_U(tx_{ins}) &= p_{prod} \cdot (v_{in} - cost) + (1 - p_{prod}) \cdot v_{out} \\ u'_I(tx_{ins}) &= p_{prod} \cdot P - (1 - p_{prod}) \cdot C\end{aligned}$$

□

Claim 14. *The Subgame "No tx_{ins}" when delay is large is in SPE when the protocol is followed and the Insurer invokes "Allocation Wasted". The Subgame is depicted in Figure 6.4c.*

Proof of Claim. In this subgame, shown in Figure 6.4c, the User lets $deadline$ pass doing nothing and then gets the reimbursement or not. We note that when the User does not send tx_{ins}, the utilities are the same whether the delay is large or small.

So, the claim holds for the same reason as Claim 10 and the resulting utilities are once more

$$\begin{aligned}u'_U(\text{No } tx_{ins}) &= p_{prod} \cdot (v_{out} - cost) + (1 - p_{prod}) \cdot v_{out} \\ u'_I(\text{No } tx_{ins}) &= p_{prod} \cdot \alpha \cdot P - (1 - p_{prod}) \cdot C\end{aligned}$$

□

Claim 15. *The Subgame "Self Submission", when delay is large,e is in SPE when the protocol is followed and the Insurer invokes "Allocation Wasted". The Subgame is depicted in Figure 6.4c.*

Proof of Claim. In this subgame, the user submits tx_{ins} for inclusion and sends nothing to the insurer. We note that when the delay is large, tx_{ins} is not included in time this way. Thus, the utilities are the same as if the User let the time pass, except that the User paid F_{tx} more in this case.

The claim holds for the same reason as in Claim 11, but this time the resulting utilities are

$$\begin{aligned}u'_U(\text{Self Submission}) &= p_{prod} \cdot (v_{out} - F_{tx} - cost) + (1 - p_{prod}) \cdot (v_{out} - F_{tx}) \\ u'_I(\text{Self Submission}) &= p_{prod} \cdot alpha \cdot P - (1 - p_{prod}) \cdot C\end{aligned}$$

□

Claim 16. *The Subgame "Delay is Large" is in SPE when the protocol is followed and the User sends $tx_{ins}$to the Insurer as the protocol dictates.*

Proof of Claim. The claim holds as $u'_U(tx_{ins}) > u'_U(\text{Self Submission}) > u'_U(\text{No } tx_{ins})$, according to Claims 13, 14 and 15. So the User's best action is to send $tx_{ins}$to the Insurer and thus we have

$$u'_U(\text{delay large}) = p_{prod} \cdot (v_{in} - cost) + (1 - p_{prod}) \cdot v_{out} \tag{6.1}$$
$$u'_I(\text{delay large}) = p_{prod} \cdot P - (1 - p_{prod}) \cdot C \tag{6.2}$$

□

We conclude that, under the necessary assumption $P < \frac{F_{tx}}{1-p_{prod}} - F - F_C$, both parties maximize their utilities by following the protocol with no deviations, regardless of the delay being small or large.

Furthermore, having identified the utilities of the agents in the key subgames in equilibrium, we calculate the expected utilities received by the parties when they create the contract. To do so, we add the utilities of "Delay Small" and "Delay Large" multiplied by the corresponding probability. However, we can see that the utilities are bounded and even when the probability is impossible to calculate, the protocol provides a lower bound in the utilities received by the parties.

$$u'_U(\text{Contract Completed}) = p_{small} \cdot (v_{in} - cost) + (1 - p_{small}) \cdot (p_{prod} \cdot (v_{in} - cost) + (1 - p_{prod}) \cdot v_{out}) \tag{6.3}$$

$$u'_U(\text{Contract Completed}) \geq p_{prod} \cdot (v_{in} - cost) + (1 - p_{prod}) \cdot v_{out} \tag{6.4}$$

$$u'_I(\text{Contract Completed}) = p_{small} \cdot P + (1 - p_{small}) \cdot (p_{prod} \cdot P - (1 - p_{prod}) \cdot C) \tag{6.5}$$

$$u'_I(\text{Contract Completed}) \geq p_{prod} \cdot P - (1 - p_{prod}) \cdot C \tag{6.6}$$

We have proven that the game is an SPE and the User gets guaranteed utility greater or equal to

$$p_{prod} \cdot (v_{in} - cost) + (1 - p_{prod}) \cdot v_{out}$$

Thus, Theorem 6.2.1 holds, as

$$\begin{aligned}
p_{prod} \cdot (v_{in} - cost) + (1 - p_{prod}) \cdot v_{out} &\leq (1 - \epsilon)v_{in} \\
\frac{p_{prod} \cdot (v_{in} - cost) + (1 - p_{prod}) \cdot v_{out}}{v_{in}} &\leq 1 - \epsilon \\
\epsilon &\leq 1 - \frac{p_{prod} \cdot (v_{in} - cost) + (1 - p_{prod}) \cdot v_{out}}{v_{in}} \\
\epsilon &\leq 1 - p_{prod} - \frac{p_{prod} \cdot (F + F_C + P) + (1 - p_{prod}) \cdot v_{out}}{v_{in}}
\end{aligned}$$

□

6.3 Submission Without a Contract

In this section, we analyze the utilities of the two parties when no contract is created. Then we compare those utilities to those derived from the Chapters 6.1, 6.2 and determine when the parties maximize their utilities by creating a contract.

6.3.1 No Creation

Lemma 6.3.1. *When the User follows the usual submission process instead of making an agreement with an Insurer, her expected utility is dependent on p_{small} and is equal to*

$$\mathbb{E}(U) = p_{small} \cdot (v_{in} - F_{tx}) + (1 - p_{small}) \cdot v_{out}$$

Proof of Lemma. We focus on the subgame "No Creation", as shown in Figure 6.5, and use backward induction to deduce the utilities when the two players follow the best course of action.

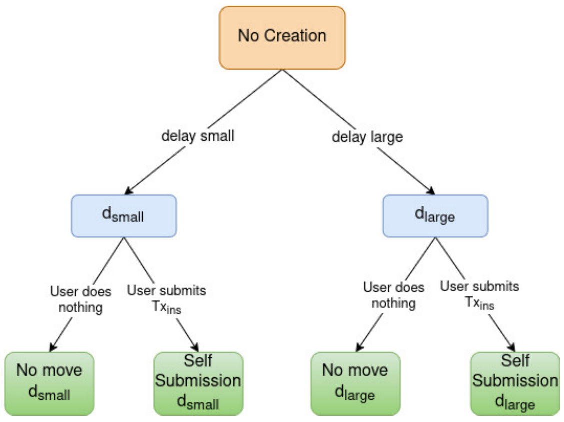


Figure 6.5: No Contract

- u_U(No Move)$= v_{out}$, u_I(No Move)$= 0$

  First of all, we consider the case of "No Move" with nothing happening, no contract creation and no tx_{ins} included in the ledger. For the User, the utility of this outcome is v_{out}. She spent no fee on including the contract nor the tx_{ins} in the ledger, nor paid anything to an insurer. However, tx_{ins} was not included. The insurer gets utility 0 as she does not interact with the protocol.

- u_U(Self Submission d_{large})$= v_{out} - F_{tx}$, u_I(Self Submission d_{large})$= 0$

  In outcome "Self Submission d_{large}", we have no contract created and the user submits tx_{ins} herself for creation. In addition, the delay is large, so tx_{ins} will be included after $deadline$. It is obvious the Insurer's utility is the same as the previous analysis, as she has no involvement with tx_{ins}. The User's utility, on the other hand,d comes from her utility for having tx_{ins} submitted minus the fee paid for the inclusion and so is $v_{out} - F_{tx}$.

- u_U(Self Submission d_{small})$= v_{in} - F_{tx}$, u_I(Self Submission d_{small})$= 0$

  The outcome "Self Submission d_{small}" is the same as the previous one, with the only difference being that the delay is small and thus tx_{ins} is included before $deadline$.

- $u_U(d_{small}) = v_{in} - F - tx$, $u_I(d_{small}) = 0$
  As the User's utility is greater than F_{tx} when tx_{ins} is included in time and the delay is small.

- $u_U(d_{large}) = v_{out}$, $u_I(d_{large}) = 0$
  As v_{out} is less than F_{tx}

The resulting expected utility is calculated by the sum of the User's utility when the delay is small or large, multiplied by the respective probability of the delay being small or large. So, the lemma holds and the resulting utilities are

$$u_U(\text{No Creation}) = p_{small} \cdot (v_{in} - F_{tx}) + (1 - p_{small}) \cdot v_{out}, \qquad u_I(\text{No Creation}) = 0$$

$\square$

6.3.2 Conclusion

We analyze the best actions of the two parties in the contract creation game, shown in Figure 6.6, by comparing the expected utilities of Lemma 6.3.1 and the ones derived from the analysis of simple game, shown in Expressions 6.1 and 6.2 and the extended one, shown in Expressions 6.3 and 6.5. That is, in the state "Beginning" where the User decides whether to initiate the protocol or not and in the state "Half Contract" where the Insurer decides whether to complete the contract or not.

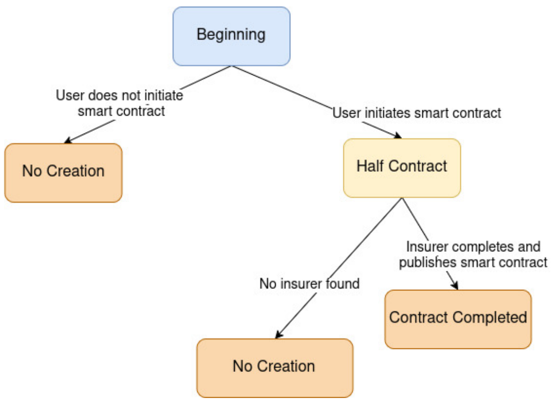

Beginning
User does not initiate smart contract
User initiates smart contract
No Creation
Half Contract
Insurer completes and publishes smart contract
No insurer found
Contract Completed
No Creation

Figure 6.6: Creation Choice

Simple Game

- $u_U(\text{Half Contract}) = v_{in} - cost,$ $\quad u_I(\text{Half Contract}) = P$, since $u_I(\text{Contract Completed}) > u_I(\text{No Creation})$
- $u_U(\text{Beginning}) = max(\mathbb{E}(U), v_{in} - cost)$
  $$u_I(\text{Beginning}) = \begin{cases} 0 & \mathbb{E}(U) \geq v_{in} - cost \\ P & \mathbb{E}(U) < v_{in} - cost \end{cases}$$

We see that when the User initiates a contract, the Insurer has a clear incentive to complete it. However, the User's decision is more complicated as the best choice depends on the probability of the future delay. A central assumption in this work is that this probability is unknown at the time of the contract initiation and an estimation made by the User is hard and has a high variance. Our protocol provides a secure outcome that guarantees a certain utility.

We consider the User having a worst-case scenario of p_{small} and deciding based on that. If $\mathbb{E}(U) \geq v_{in} - (F + F_C)$, even when taking into account the worst-case probability, then she does not initiate the contract and waits to publish tx_{ins} in the second phase. If $\mathbb{E}(U) < v_{in} - (F+F_C)$, the user initiates the contract picking a price P such that $\mathbb{E}(U) < v_{in} - (F+F_C+P)$.

Extended Game

When we compare these utilities to the ones of "No Creation", we conclude that for the Insurer to maximize her utility by completing the contract, the following assumption is needed.

$$\begin{aligned}
u'_I(\text{No Creation}) &< u'_I(\text{Contract Completed})\\
0 &< p_{small} \cdot P + (1 - p_{small}) \cdot (p_{prod} \cdot P - (1 - p_{prod}) \cdot C)\\
0 &< P - P + p_{small} \cdot P + (1 - p_{small}) \cdot p_{prod} \cdot P - (1 - p_{small}) \cdot (1 - p_{prod}) \cdot C\\
0 &< P - (1 - p_{small}) \cdot P + (1 - p_{small}) \cdot p_{prod} \cdot P - (1 - p_{small}) \cdot (1 - p_{prod}) \cdot C\\
0 &< P - (1 - p_{small}) \cdot P \cdot (1 - p_{prod}) - (1 - p_{small}) \cdot (1 - p_{prod}) \cdot C\\
0 &< (1 - (1 - p_{small}) \cdot (1 - p_{prod})) \cdot P - (1 - p_{small}) \cdot (1 - p_{prod}) \cdot C
\end{aligned}$$

The assumption can be stricter for an Insurer that considers the worst-case p_{small}.

$$0 < p_{prod} \cdot P - (1 - p_{prod}) \cdot C$$

Similarly, for the User to maximize her utility by initiating the contract, we need

$$u'_U(\text{No Creation}) < u'_I(\text{No Creation})$$

$$p_{small} \cdot (v_{in} - F_{tx}) + (1 - p_{small}) \cdot v_{out} < p_{small} \cdot (v_{in} - cost) + (1 - p_{small}) \cdot (p_{prod} \cdot (v_{in} - cost) + (1 - p_{prod})v_{out})$$

$$p_{small} \cdot (v_{in} - F_{tx}) + (1 - p_{small}) \cdot v_{out} < p_{small} \cdot v_{in} - p_{small} \cdot cost + (1 - p_{small}) \cdot (p_{prod} \cdot (v_{in} - cost) + (1 - p_{prod})v_{out})$$

$$-p_{small} \cdot F_{tx} < -p_{small} \cdot cost + (1 - p_{small}) \cdot p_{prod} \cdot (v_{in} - cost) + (1 - p_{small}) \cdot (1 - p_{prod}) \cdot v_{out} - (1 - p_{small}) \cdot v_{out}$$

$$p_{small} \cdot cost - p_{small} \cdot F_{tx} < (1 - p_{small}) \cdot p_{prod} \cdot (v_{in} - cost) - (1 - p_{small}) \cdot p_{prod} \cdot v_{out}$$

$$p_{small} \cdot (cost - F_{tx}) < (1 - p_{small}) \cdot p_{prod} \cdot (v_{in} - cost - v_{out})$$

As in the simple game, we consider the User to have an estimation of p_{small}. When this estimation is hard or highly variant, the User might prefer to secure a guaranteed utility by initiating the protocol.

Chapter 7

Final Remarks

7.1 Conclusion

In this work, we delved into one of the main issues of blockchain systems, scalability. More specifically, we considered the aspect of dealing with a greater number of transactions while keeping the system fair and providing a quality service to the users.

We looked at TFMs, the core mechanism used by blockchain systems to provide a solution to the problem. We referred to different mechanisms and noted various works analyzing them and searching for the ideal mechanism.

However, the issue of quality of service persists, as the unpredictability of service arises. The need for users to know ahead of time the service they will receive creates an open space for work. Block-space derivatives, an underexplored line of work, aim to give a solution by providing ways of buying space for transactions earlier than the submission.

We give a formal definition for the predictable service, a notion describing a protocol that guarantees a future utility and gives a solution to the unpredictability problem. We prove that, assuming minimum prediction capabilities, dynamic-fee systems cannot provide predictable service.

Finally, we describe in detail our service predictable protocol, based on a fixed-fee setting. We model this protocol into a game in two different cases. In the first case, the agreement for the block-space is made closer to the submission of the transaction, while at the second it can be arbitrarily earlier. We prove the game to be an SPE for the first case and an SPE under a suitable assumption for the second case. Furthermore, the expected utility, when no protocol is initiated, is calculated and compared to the guarantees of our protocol.

We conclude that our protocol provides a predictable service, as this was defined in this work.

7.2 Future Work

The most immediate continuation of our work is the implementation of our protocol into a smart contract. The protocol was written having the Cardano ecosystem primarily in mind and the implementation in Plutus is the natural next step. However, the adaptation to other fixed-fee protocols is possible and encouraged.

Furthermore, our protocol could be extended to include the insurance of more than one future transaction. The User could specify multiple future transactions to be included in different intervals in the future. A single agreement could be made to accommodate all of those transactions with a single payment for the Insurer after the successful completion of the agreement, or individual payments after every single transaction.

In terms of research, our work focused on the basic model of a single User - Insurer pair and the proofs omit taking into account the possibility of having multiple contracts. In a generalized setting, a User might create multiple contracts with multiple insurers for the same transaction, to achieve greater security. In addition, the Insurer might create multiple contracts with different users and abandon one for the sake of another with greater payment. The analysis of this generalized setting goes beyond the scope of this work.

Bibliography

[1] 1inch Network, “1inch introduces chi gastoken,” June 2020, accessed: 2025-05-26. [Online]. Available: https://blog.1inch.io/1inch-introduces-chi-gastoken/

[2] C. Badertscher, P. Gazi, A. Kiayias, A. Russell, and V. Zikas, “Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability,” Feb 2019. [Online]. Available: https://eprint.iacr.org/2018/378

[3] C. Badertscher, P. Gaži, A. Kiayias, A. Russell, and V. Zikas, “Ouroboros chronos: Permissionless clock synchronization via proof-of-stake,” Cryptology ePrint Archive, Paper 2019/838, 2019. [Online]. Available: https://eprint.iacr.org/2019/838

[4] A. Bailly, B. W. Bush, S. Coretti-Drayton, Y. Hauser, and H. Lahe, “Cip-0140: Ouroboros peras - faster settlement,” https://github.com/cardano-foundation/CIPs/blob/master/CIP-0140/README.md, 2024, cardano Improvement Proposals (CIPs), Cardano Foundation, Proposed Status.

[5] S. Basu, D. Easley, M. O’Hara, and E. G. Sirer, “Stablefees: A predictable fee market for cryptocurrencies,” Cornell University, Working Paper, January 2019, available at SSRN. [Online]. Available: https://ssrn.com/abstract=3318327

[6] M. Baxter. (2024, September) The five development phases of the cardano roadmap. Accessed: 2025-05-18. [Online]. Available: https://www.learningcardano.com/the-five-development-phases-of-cardano/

[7] J. Benson, “Ethereum london hard fork to make some tokens worthless,” *Decrypt*, July 2021, accessed: 2025-05-26. [Online]. Available: https://decrypt.co/77345/ethereum-london-hard-fork-make-some-tokens-worthless

[8] J. Bier, *The Blocksize War: The Battle for Control of Bitcoin’s Protocol.* Amazon KDP, 2021.

[9] L. Breidenbach, P. Daian, F. Tramer, C. Clauss, and S. Tatvamasi, “Gastoken.io - cheaper ethereum transactions, today,” 2018, accessed: 2025-05-26. [Online]. Available: https://gastoken.io/

[10] L. Brünjes, “How cardano’s transaction fees work,” October 2017, accessed: 2025-05-11. [Online]. Available: https://iohk.io/en/blog/posts/2017/10/19/how-cardanos-transaction-fees-work/

[11] L. Brünjes, A. Kiayias, E. Koutsoupias, and A.-P. Stouka, “Reward sharing schemes for stake pools,” in *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*, 2020, pp. 256–275.

[12] J. Burdges, A. Cevallos, P. Czaban, R. Habermeier, S. Hosseini, F. Lama, H. K. Alper, X. Luo, F. Shirazi, A. Stewart, and G. Wood, “Overview of polkadot and its design considerations,” 2020. [Online]. Available: https://arxiv.org/abs/2005.13456

[13] V. Buterin, E. Conner, R. Dudley, M. Slipper, I. Norden, and A. Bakhta, “EIP-1559: Fee market change for ETH 1.0 chain,” https://eips.ethereum.org/EIPS/eip-1559, 2019, ethereum Improvement Proposal 1559.

[14] V. Buterin and V. Griffith, “Casper the friendly finality gadget,” 2019. [Online]. Available: https://arxiv.org/abs/1710.09437

[15] V. Buterin and M. Swende, “Eip-3529: Reduction in refunds,” April 2021, accessed: 2025-05-26. [Online]. Available: https://eips.ethereum.org/EIPS/eip-3529

[16] Cardanians, “Understanding cardano monetary policy,” Authored by: https://cardanians.io, November 2023, accessed: 2025-05-14. [Online]. Available: https://cexplorer.io/article/understanding-cardano-monetary-policy?utm_source=chatgpt.com

[17] M. M. T. Chakravarty, J. Chapman, K. MacKenzie, O. Melkonian, M. Peyton Jones, and P. Wadler, “The extended utxo model,” in *Financial Cryptography and Data Security*, M. Bernhard, A. Bracciali, L. J. Camp, S. Matsuo, A. Maurushat, P. B. Rønne, and M. Sala, Eds. Cham: Springer International Publishing, 2020, pp. 525–539.

[18] J. Chen and S. Micali, “Algorand,” 2017. [Online]. Available: https://arxiv.org/abs/1607.01341

[19] H. Chung, T. Roughgarden, and E. Shi, “Collusion-resilience in transaction fee mechanism design,” in *Proceedings of the 25th ACM Conference on Economics and Computation*, ser. EC ’24. ACM, Jul. 2024, p. 1045–1073. [Online]. Available: http://dx.doi.org/10.1145/3670865.3673550

[20] H. Chung and E. Shi, “Foundations of transaction fee mechanism design,” in *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2023, pp. 3856–3899. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9781611977554.ch150

[21] S. Coretti-Drayton, M. Fitzi, A. Kiayias, G. Panagiotakos, and A. Russell, “High-throughput blockchain consensus under realistic network assumptions,” May 2024, iOHK Research Paper. [Online]. Available: https://iohk.io/en/research/library/papers/high-throughput-blockchain-consensus-under-realistic-network-assumptions/

[22] D. Crapis, C. C. Moallemi, and S. Wang, “Optimal dynamic fees for blockchain resources,” 2023. [Online]. Available: https://arxiv.org/abs/2309.12735

[23] B. David, P. Gaži, A. Kiayias, and A. Russell, "Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain," in *Advances in Cryptology – EUROCRYPT 2018*, J. B. Nielsen and V. Rijmen, Eds. Cham: Springer International Publishing, 2018, pp. 66–98.

[24] J. Day, *Patterns in Network Architecture: A Return to Fundamentals.* Prentice Hall, 01 2008.

[25] Degenerative Finance, "ugas token," 2021, accessed: 2025-05-26. [Online]. Available: https://web.archive.org/web/20220119144017/https://docs.degenerative.finance/

[26] B. Edgington, "Upgrading ethereum: A technical handbook on ethereum's move to proof of stake and beyond," https://eth2book.info/latest/consensus/gasper/, 2025, section: Consensus – Gasper.

[27] Ethereum Foundation, "Ethereum london hard fork," https://ethereum.org/en/history/#london, 2021, accessed May 2025.

[28] M. V. X. Ferreira, D. J. Moroz, D. C. Parkes, and M. Stern, "Dynamic posted-price mechanisms for the blockchain transaction-fee market," in *Proceedings of the 3rd ACM Conference on Advances in Financial Technologies*, ser. AFT '21. ACM, Sep. 2021, p. 86–99. [Online]. Available: http://dx.doi.org/10.1145/3479722.3480991

[29] Y. Gafni and A. Yaish, "Discrete & bayesian transaction fee mechanisms," 2024. [Online]. Available: https://arxiv.org/abs/2210.07793

[30] P. Gaži, A. Kiayias, and D. Zindros, "Proof-of-stake sidechains," in *2019 IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 139–156.

[31] N. Houy, "The economics of bitcoin transaction fees," GATE - Groupe d'Analyse et de Théorie Économique, Working Paper 1407, February 2014, available at SSRN. [Online]. Available: https://ssrn.com/abstract=2400519

[32] O. Hryniuk and N. Burgess. (2025, January) Chang. Accessed: 2025-05-18. [Online]. Available: https://docs.cardano.org/about-cardano/evolution/upgrades/chang

[33] J. C. Hull, *Options, Futures, and Other Derivatives.* Harlow, Essex, England: Pearson Education Limited, 2012, authorized adaptation from the United States edition published by Pearson Education as Prentice Hall. [Online]. Available: https://www.pearsoned.co.uk

[34] IOHK. Cardano roadmap. Accessed: 2025-05-18. [Online]. Available: https://roadmap.cardano.org/en/

[35] T. Kerber, M. Kohlweiss, A. Kiayias, and V. Zikas, "Ouroboros crypsinous: Privacy-preserving proof-of-stake," May 2019. [Online]. Available: https://eprint.iacr.org/2018/1132

[36] A. Kiayias, E. Koutsoupias, P. Lazos, and G. Panagiotakos, "Tiered mechanisms for blockchain transaction fees," 2023. [Online]. Available: https://arxiv.org/abs/2304.06014

[37] ——, “Blockchain space tokenization,” Cryptology ePrint Archive, Paper 2024/1154, 2024. [Online]. Available: https://eprint.iacr.org/2024/1154

[38] A. Kiayias and A. Russell, “Ouroboros-BFT: A simple byzantine fault tolerant consensus protocol,” Cryptology ePrint Archive, Paper 2018/1049, 2018. [Online]. Available: https://eprint.iacr.org/2018/1049

[39] A. Kiayias, A. Russell, B. David, and R. Oliynykov, “Ouroboros: A provably secure proof-of-stake blockchain protocol,” Cryptology ePrint Archive, Paper 2016/889, 2016. [Online]. Available: https://eprint.iacr.org/2016/889

[40] A. Kiayias and D. Zindros, “Proof-of-work sidechains,” in *Financial Cryptography and Data Security*, A. Bracciali, J. Clark, F. Pintore, P. B. Rønne, and M. Sala, Eds. Cham: Springer International Publishing, 2020, pp. 21–34.

[41] R. Lavi, O. Sattath, and A. Zohar, “Redesigning bitcoin’s fee market,” 2022. [Online]. Available: https://arxiv.org/abs/1709.08881

[42] S. Leonardos, D. Reijsbergen, B. Monnot, and G. Piliouras, “Optimality despite chaos in fee markets,” 2022. [Online]. Available: https://arxiv.org/abs/2212.07175

[43] Y. Liu, Y. Lu, K. Nayak, F. Zhang, L. Zhang, and Y. Zhao, “Empirical analysis of eip-1559: Transaction fees, waiting times, and consensus security,” in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS ’22. ACM, Nov. 2022, p. 2099–2113. [Online]. Available: http://dx.doi.org/10.1145/3548606.3559341

[44] A. Lotem, S. Azouvi, P. McCorry, and A. Zohar, “Sliding window challenge process for congestion detection,” 2022. [Online]. Available: https://arxiv.org/abs/2201.09009

[45] K. McKenzie, “An overview of the plutus core cost model,” https://github.com/IntersectMBO/plutus/blob/master/doc/cost-model-overview/cost-model-overview.pdf, Intersect MBO, October 2024, accessed: 2025-05-11.

[46] M. Morini, “Cardano economic parameters,” February 2025, accessed: 2025-05-14. [Online]. Available: https://ucarecdn.com/cb24f376-d3bd-40c3-9c29-04573a644c8a/

[47] M. Nadler, “Master thesis on dlt and fintech,” Master’s thesis, University of Basel, WWZ, 2020, accessed: 2025-05-26. [Online]. Available: https://wwz.unibas.ch/fileadmin/user_upload/wwz/00_Professuren/Schaer_DLTFintech/Lehre/Teaching_Page/Nadler_MasterThesis_2020-1_copy.pdf

[48] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[49] A. Ndiaye, “Blockchain price vs. quantity controls,” *SSRN Electronic Journal*, April 2024, available at SSRN: https://ssrn.com/abstract=4538155.

[50] O. Network, “Pitch lake,” *Available at SSRN 4123018*, 2022.

[51] G. Papadoulis, D. Balla, P. Grontas, and A. Pagourtzis, "AQQUA: Augmenting quisquis with auditability," Cryptology ePrint Archive, Paper 2024/1181, 2024. [Online]. Available: https://eprint.iacr.org/2024/1181

[52] D. Reijsbergen, S. Sridhar, B. Monnot, S. Leonardos, S. Skoulakis, and G. Piliouras, "Transaction fees on a honeymoon: Ethereum's eip-1559 one month later," 2022. [Online]. Available: https://arxiv.org/abs/2110.04753

[53] P. R. Rizun, "A transaction fee market exists without a block size limit," August 2015, accessed: 2025-05-26. [Online]. Available: https://www.bitcoinunlimited.info/resources/feemarket.pdf

[54] T. Roughgarden, "Transaction fee mechanism design," *SIGecom Exch.*, vol. 19, no. 1, p. 52–55, Jul. 2021. [Online]. Available: https://doi.org/10.1145/3476436.3476445

[55] F. Sanchez, "Eutxo handbook for the education community," August 2022, accessed: 2025-05-11. [Online]. Available: https://ucarecdn.com/3da33f2f-73ac-4c9b-844b-f215dcce0628/EUTXOhandbook_for_EC.pdf

[56] Y. Shoham and K. Leyton-Brown, *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations.* Cambridge University Press, 2008. [Online]. Available: https://books.google.gr/books?id=bMR_qScakukC

[57] E. G. Sirer, D. Laine, S. Buttolph, and K. Sekniqi, "Avalanche platform whitepaper," Ava Labs, Whitepaper, 2020, available at https://cdn.prod.website-files.com/5d80307810123f5ffbb34d6e/6008d7bbf8b10d1eb01e7e16_Avalanche%20Platform%20Whitepaper.pdf.

[58] B. v. Stengel, *Equilibrium Computation for Two-Player Games in Strategic and Extensive Form.* Cambridge University Press, 2007, p. 53–78.

[59] F. Stoyanov. (2024, May) Allegra. Accessed: 2025-05-18. [Online]. Available: https://docs.cardano.org/about-cardano/evolution/upgrades/allegra

[60] ——. (2024, May) Alonzo. Accessed: 2025-05-18. [Online]. Available: https://docs.cardano.org/about-cardano/evolution/upgrades/alonzo

[61] ——. (2024, May) Byron to shelley. Accessed: 2025-05-18. [Online]. Available: https://docs.cardano.org/about-cardano/evolution/upgrades/byron-to-shelley

[62] ——, "Cardano keys," May 2024. [Online]. Available: https://docs.cardano.org/about-cardano/learn/cardano-keys

[63] ——. (2024, May) Development phases and eras. Accessed: 2025-05-18. [Online]. Available: https://docs.cardano.org/about-cardano/evolution/eras-and-phases

[64] ——, "Fee structure," May 2024, accessed: 2025-05-11. [Online]. Available: https://docs.cardano.org/about-cardano/explore-more/fee-structure

[65] ——. (2024, May) Mary. Accessed: 2025-05-18. [Online]. Available: https://docs.cardano.org/about-cardano/evolution/upgrades/mary

[66] ——. (2024, May) Vasil. Accessed: 2025-05-18. [Online]. Available: https://docs.cardano.org/about-cardano/evolution/upgrades/vasil

[67] F. Stoyanov and O. Hryniuk, "Ouroboros overview," March 2025, accessed: 2025-05-11. [Online]. Available: https://docs.cardano.org/about-cardano/learn/ouroboros-overview

[68] W. Tang and D. D. Yao, "Transaction fee mechanism for proof-of-stake protocol," 2023. [Online]. Available: https://arxiv.org/abs/2308.13881

[69] I. Tsabary, A. Manuskin, R. Bar-Zur, and I. Eyal, "Ledgerhedger: Gas reservation for smart contract security," in *Financial Cryptography and Data Security*, J. Clark and E. Shi, Eds. Cham: Springer Nature Switzerland, 2025, pp. 248–270.

[70] P. Vinogradova, W. Gould, and (potentially) Alexey, "Cip-118: Nested transactions," https://github.com/polinavino/CIPs/blob/polina/CIP0118/CIP-0118/README.md, 2024, cardano Improvement Proposals.

[71] A. C.-C. Yao, "An incentive analysis of some bitcoin fee designs," 2018. [Online]. Available: https://arxiv.org/abs/1811.02351

[72] L. Zhang and F. Zhang, "Understand waiting time in transaction fee mechanism: An interdisciplinary perspective," 2023. [Online]. Available: https://arxiv.org/abs/2305.02552

[73] Ε. Πετράκης, *Σημειώσεις Θεωρίας Παιγνίων*, 2011, available at https://mathbooksgr.wordpress.com/wp-content/uploads/2011/08/gt_simiwseis.pdf.