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

Πρωτόκολλα τυχερών παιγνίων με χρήση τεχνολογίας αλυσίδας συναλλαγών (**blockchain**)

Διπλωματική Εργασία
Κουτσός Βλάσιος

Επιβλέπων: Αριστείδης Παγουρτζής
Αναπληρωτής Καθηγητής Ε.Μ.Π.

Εργαστήριο Λογικής και Επιστήμης Υπολογισμών
Αθήνα, Ιούλιος 2018

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

Πρωτόκολλα τυχερών παιγνίων με χρήση τεχνολογίας αλυσίδας συναλλαγών (**blockchain**)

Διπλωματική Εργασία
Κουτσός Βλάσιος

Επιβλέπων: Αριστείδης Παγουρτζής
Αναπληρωτής Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 20η Ιουλίου 2018.

....	....	....
Αριστείδης Παγουρτζής	Αντώνιος Συμβώνης	Παναγιώτης Τσανάκας
Αν. Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Εργαστήριο Λογικής και Επιστήμης Υπολογισμών
Αθήνα, Ιούλιος 2018

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

Περίληψη

Το παίξιμο τυχερών παιγνίων είναι ένα από τα χαρακτηριστικά της ανθρώπινης φύσης από την αρχαιότητα και η ρουλέτα ειδικότερα είναι ένα από τα πιο δημοφιλή παιχνίδια στη σημερινή εποχή.

Η τεχνολογία αλυσίδας συναλλαγών τα τελευταία δέκα χρόνια έχει φέρει επανάσταση σε πολλές πτυχές της ζωής μας εισάγοντας τα κρυπτονομίσματα καθώς και άλλες καινοτόμες κατασκευές, όπως τα έξυπνα συμβόλαια.

Ο συνδυασμός των δύο έχει εξεταστεί και έχουν προταθεί διάφορες κατασκευές, αλλά όλες είναι κεντροποιημένες ή απαιτούν εμπιστοσύνη σε κάποιο επίπεδο, απαιτούν αμοιβές και δεν προσφέρουν επιλογές όσον αφορά το παιχνίδι στους συμμετέχοντες. Στο πρωτόκολλό μας αντιμετωπίζουμε αυτά τα προβλήματα και τα επιλύουμε.

Παρουσιάζουμε ότι δημιουργώντας μια πλευρική αλυσίδα σε οποιαδήποτε αλυσίδα συναλλαγών ο καθένας μπορεί να δημιουργήσει το δικό του παιχνίδι. Επίσης, χρησιμοποιώντας ένα σχήμα υπογραφής παράλληλα με το σχήμα κρυπτογράφησης ElGamal και τη συνάρτηση κατακερματισμού SHA256 , αποδεικνύουμε την τυχαιότητα του αποτελέσματος στο μοντέλο τυχαίου μαντείου. Περιλαμβάνουμε επίσης ένα μηχανισμό αποκατάστασης για την τιμωρία των παικτών που αποκλίνουν από το πρωτόκολλο και ενός αλγορίθμου για την εξασφάλιση της ακεραιότητας των περιουσιακών στοιχείων των παικτών. Από όσο γνωρίζουμε, αυτό είναι το πρώτο πλήρος αποκεντρωμένο, μηδενικής εμπιστοσύνης και μηδενικής προμήθειας κρυπτογραφικό πρωτόκολλο το οποίο εκτελεί ένα παιχνίδι ρουλέτας, παρέχοντας επιλογές σε σχέση με τα τυχερά παιχνίδια
.

Λέξεις Κλειδιά

τυχερά παίγνια, τεχνολογία αλυσίδας συναλλαγών, πλευρικές αλυσίδες

Abstract

Gambling has been one of the traits of human nature since the beginning of time and roulette in particular is one of the most popular games today.

Blockchain technology for almost ten years now has revolutionized many aspects of our lives introducing cryptocurrencies as well as other innovative constructions such as smart contracts.

The combination of the two has been examined and various constructions have been proposed but all of them are either centralized or require trust at some level, require fees and do not offer choices with respect to gambling to players. In our protocol we address these problems and solve them.

We show that by generating a sidechain to any blockchain everyone can create his own game. Also, by using a signature scheme alongside the ElGamal encryption scheme and the hash function SHA256 we prove the randomness of the result in the random oracle model. We include as well a recovery scheme for penalizing players who deviate from the protocol and an algorithm to ensure the integrity of the players' assets. To the best of our knowledge this is the first fully-decentralized, zero-trust, zero-fee cryptographic protocol which executes a game of roulette, providing options with respect to gambling.

Keywords

gambling, blockchain technology, sidechain,

Ευχαριστίες

Η διπλωματική αυτή αποτελεί προσωπική επιλογή του συγγραφέα και είναι ένα έργο για το οποίο δαπανήθηκαν πολλές ημέρες και ακόμα περισσότερες νύχτες μελέτης, αναζήτησης και αδιεξόδων μέχρι να φτάσει στο επίπεδο των προσδοκιών του συγγραφέα. Η ολοκλήρωση της δεν θα ήταν δυνατή εάν δεν υπήρχαν συγκεκριμένα άτομα που μου παρείχαν την απαραίτητη υποστήριξη τόσο ψυχολογική αλλά και σε επίπεδο γνώσεων.

Αρχικά, θα ήθελα να ευχαριστήσω τον καθηγητή και επιβλέποντα αυτής της διπλωματικής κ. Παγουρτζή, ο οποίος ήταν αρωγός της συγκεκριμένης εργασίας και προσέφερε τη βοήθεια του καθ΄ όλη την διάρκεια της με όλους τους δυνατούς τρόπους.

Τον καθηγητή κ. Παπαδόπουλο για τις χρησιμότατες γνώσεις του στο κομμάτι των blockchains κυρίως, αλλά όχι μόνο, το ανιδιοτελές ενδιαφέρον του για τη συγκεκριμένη διπλωματική και την πολύτιμη βοήθεια που μου παρείχε.

Τέλος, θα ήθελα να ευχαριστήσω τους φίλους μου και κυρίως την οικογένεια μου για την αμέριστη στήριξη τους σε κάθε μου βήμα.

Contents

List of Figures

List of Tables

Κεφάλαιο 1

Περιγραφή Πρωτοκόλλου

Εισαγωγή

Η τεχνολογία πάντα έπαιζε σημαντικό ρόλο στην ανθρώπινη ιστορία. Ολόκληρες εποχές έχουν λάβει το όνομά τους από σημαντικά τεχνολογικά άλματα όπως η λίθινη εποχή και η εποχή του χαλκού. Τα τελευταία 20 χρόνια διανύουμε αναμφίβολα την εποχή του διαδικτύου. Πρόκειται για μία εποχή που μπορεί να χαρακτηριστεί εξίσου από την ψηφιοποίηση των πάντων, από το κομμάτι των επικοινωνιών, από όπου και ξεκίνησε κιόλας, μέχρι τα βιβλία και τους πίνακες εκμάθησης σε σχολεία. Η χρήση της τεχνολογίας όμως, όπως και οποιουδήποτε άλλου πράγματος στον κόσμο, εξαρτάται από το χειριστή της. Οπότε, είναι καθήκον μας όχι μόνο να αναπτύσουμε τεχνολογικά κορυφαία προιόντα αλλά και να διασφαλίζουμε πως αυτά λειτουργούν ¨σωστά¨, προς όφελος του κοινωνικού συνόλου. Κλασσικό παράδειγμα της διττής χρήσης του διαδικτύου παραδείγματος χάριν, παρατηρείται στο κομμάτι των επικοινωνιών. Ορισμένοι τη χρησιμοποιούν για να έρθουν σε άμεση επαφή με απομακρυσμένα πρόσωπα, ενώ άλλοι για να κάνουν επιθέσεις σε διάφορες υπηρεσίες και να διαπράτουν απάτες, χρησιμοποιώντας διαφορετικά τις ίδιες δυνατότητες. Αυτή η εργασία προσανατολίζεται στο να παρέχει μία νέα δυνατότητα στο χώρο των τυχερών παιχνιδίων.

Εαν κάποιος τη χρονική περίοδο της συγγραφής της παρούσας εργασίας επιθυμούσε να εμπλακεί σε ένα τυχερό παιχνίδι θα είχε τρεις επιλογές. Να είναι παίκτης είτε σε κάποιο αναλογικό καζίνο, είτε σε κάποιο ψηφιακό που δε χρησιμοποιεί αλυσίδα συναλλαγών, είτε σε κάποιο που χρησιμοποιεί. Στην πρώτη και στη δεύτερη περίπτωση ο παίκτης θα έπρεπε να εμπιστευτεί το εν λόγω καζίνο ως προς τη διαδικασία υπολογισμού του αποτελέσματος. Αυτό συμβαίνει διότι ο ίδιος για εμπορικούς λόγους κυρίως δε θα είχε πρόσβαση για επιθεώρηση ούτε στις αναλογικές ρουλλέτες, ούτε στον κώδικα που παράγει τα ψευδοτυχαία αποτελέσματα. Μόνο στην περίπτωση που χρησιμοποιείται τεχνολογία αλυσίδας συναλλαγών μπορεί ο παίκτης να επιβεβαιώσει τα αποτελέσματα και ουσιαστικά τη διαδικασία παραγωγής τους. Ωστόσο το ένα ζήτημα που δε μπορεί να αντιμετωπιστεί στην τρίτη περίπτωση, είναι ο ¨άχρηστος χώρος' πάνω στην αλυσίδα. Αυτό σημαίνει πως κατά την εκτέλεση ενός πρωτοκόλλου ρουλλέτας τα μηνύματα που ανταλάσουν οι παίκτες αποθηκεύονται στην αλυσίδα, αλλά μετά την ολοκλήρωση του παιχνιδιού αυτά τα δεδομένα δεν έχουν καμία ουσιαστική αξία και παραμένουν άχρηστα καθώς λόγω της δομής της τεχνολογίας αλυσίδας δε μπορούν να διαγραφούν ποτέ. Τέλος, ένα αρνητικό και των τριών υπάρχουσων επιλογών αποτελεί η πληρωμή επιπλέον δασμών των παικτών για τη συντήριση αυτών των κεντροποιημένων εταιριών που παρέχουν τις υπηρεσίες τυχερών παιγνίων.

Αναγνωρίζοντας τις αδυναμίες των υπάρχουσων επιλογών, λοιπόν, στόχος της συγκεκριμένης εργασίας αποτελεί η δημιουργία ενός αποκεντροποιημένου πρωτόκολλου ως προς το παιχνίδι της ρουλλέτας που θα έχει μηδενική προμήθεια, μηδενική εμπιστοσύνη, σε ένα πλήρως αποκεντροποιημέο περιβάλλον, παρέχοντας εππιπλέον επιλογές ως προς τον τρόπο παιχνιδιού του τζογαδόρου.

Πρωτογενείς Διαδικασίες

Πριν παρουσιαστεί το πρωτόκολλο, χρειάζεται να αναλυθούν μερικές κρυπτογραφικές πρωτογενείς διαδικασίες. Αρχικά, θα παρουσιαστεί το κρυπτογραφικό σχήμα που χρησιμοποιείται, έπειτα θα εξηγηθούν οι ψηφιακές υπογραφές, που είναι ζωτικό κομμάτι του πρωτοκόλλου μας και εν τέλει θα αναλυθεί η τεχνολογία αλυσίαδας συναλλαγών μαζί με την τεχνολογία των πλευρικών αλυσιδών.

Κρυπτογράφηση Ελ-Γκαμάλ

Το ΕλΓκαμαλ κρυπτοσύστημα παρουσιάστηκε το 1984 και η ασφάλειά του βασίζεται σε ένα δυσεπίλυπτο πρόβλημα που ονομάζεται ¨πρόβλημα του διακριτού λογαρίθμου¨. Πιο συγκεκριμένα βασίζεται στην ανταλλαγή κλειδιού Diffie- Hellman, οπότε ορίζεται σε μια κυκλική ομάδα $\langle g \rangle$ ενός πρώτου τάξης m. Το σχήμα παρουσιάζεται εκτενέστερα στο παραπάνω σχήμα(1.1).

$$
\begin{array}{ll}
\mathcal{G}(1^\lambda): & \langle pk, sk \rangle \leftarrow \mathcal{G}(1^\lambda) \\
 & x \overset{\mathrm{r}}{\leftarrow} \mathbb{Z}_m, h = g^x \bmod p \\
 & pk = \langle \langle p, m, g \rangle, h \rangle \\
 & sk = x \\
 & \\
\mathcal{E}(pk, M): & M \in \langle g \rangle \\
 & r \overset{\mathrm{r}}{\leftarrow} \mathbb{Z}_m \\
 & \text{υπολογίζουμε τα } G = g^r \bmod p,\ H = h^r M \bmod p \\
 & \text{επιστρέφουμε το } \langle G, H \rangle \\
 & \\
\mathcal{D}(sk, G, H): & \text{υπολογίζουμε το } M = H/G^x \bmod p \\
 & \text{επιστρέφουμε το } M
\end{array}
$$

Εικόνα 1.1: Σχήμα κωδικοποίησης ElGamal [1]

Ψηφιακές Υπογραφές

Η κρυπτογραφία, ωστόσο, δεν περιορίζεται στην κωδικοποίηση ή αποκωδικοποίηση μηνυμάτων. Μία από τις άλλες της μορφές είναι η αυθεντικοποίηση μηνυμάτων. Ο τρόπος με τον οποίο επιτυγχάνεται αυτό είναι με τη χρήση ψηφιακών υπογραφών. Η δημιουργία μιας ψηφιακής υπογραφής απαιτεί έναν συνδυασμό πληροφοριών σχετικά με το ίδιο το μήνυμα και το μήνυμα του υπογράφοντος. Επειδή οι ψηφιακές πληροφορίες μπορούν να αντιγραφούν και να επικολληθούν, θα πρέπει να υπάρξει σύνδεση μεταξύ του μηνύματος και της ίδιας της ψηφιακής υπογραφής.[4][5] Διαφορετικά, ο παραλήπτης μπορεί να τροποποιήσει το μήνυμα πριν εμφανίσει το ζεύγος υπογραφής μηνύματος σε έναν δικαστή. Ακόμη χειρότερα, θα μπορούσε να επισυνάψει την υπογραφή σε οποιοδήποτε μήνυμα, καθόσον είναι αδύνατο να ανιχνευθεί η ηλεκτρονική «κοπή» και «επικόλληση». Συνεπώς, οι ακόλουθες επιθυμητές ιδιότητες μπορούν να εξαχθούν από μια ψηφιακή υπογραφή:

- Η ψηφιακή υπογραφή πρέπει να είναι εξαρτόμενη από το μήνυμα.
- Μόνο ο δημιουργός ενός ηλεκτρονικού μηνύματος μπορεί να υπολογίσει τη σωστή ψηφιακή υπογραφή.
- Οποιοσδήποτε λαμβάνει ένα μήνυμα και την αντίστοιχη ψηφιακή υπογραφή μπορεί να την επιβεβαιώσει και εν συνεχεία να είναι βέβαιος πως για την καταγωγή και την ακεραιότητα του μηνύματος.

Σύμφωνα με τα παραπάνω προκύπτουν οι ακόλουθοι ορισμοί για τα δομικά στοιχεία του σχήματος των ψηφιακών υπογραφών. Ψηφιακή υπογραφή είναι ένα μαθηματικό σχήμα το οποίο αποδεικνύει τη γνησιότητα ενός ψηφιακού κειμένου ή μηνύματος και την ταυτότητα του συγγραφέα του. Αλγόριθμος παραγωγής ψηφιακών υπογραφών είναι η μέθοδος η οποία ακολουθείται για την παραγωγή ψηφιακών υπογραφών. Αλγόριθμος επαλήθευσης ψηφιακών υπογραφών είναι ο αλγόριθμος που εξασφαλίζει τη γνησιότητα της ψηφιακής υπογραφής. Ένα σχήμα ψηφιακών υπογραφών αποτελείται από έναν αλγόριθμο παραγωγής υπογραφών και έναν σχετικό αλγόριθμο επαλήθευσης.[4] Μια διαδικασία υπογραφής ψηφιακών υπογραφών συνίσταται σε έναν αλγόριθμο παραγωγής ψηφιακών υπογραφών, σε συνδυασμό με την μέθοδο μορφοποίησης των δεδομένων σε μηνύματα. Μια διαδικασία επαλήθευσης ψηφιακών υπογραφών αποτελείται από έναν αλγόριθμο επαλήθευσης, μαζί με μια μέθοδο ανάκτησης δεδομένων από το μήνυμα που χρειάζεται να υπογραφεί.

Τεχνολογία Αλυσίδας Συναλλαγών

Δεν υπάρχει, προς γνώση του συγγραφέα ένας καθολικός ορισμός και μετάφραση για τον όρο "blockchain". Στην εργασία αυτή αναφερόμενοι σε ¨τεχνολογία αλυσίδας συναλλαγών' αναφερόμαστε στον Αγγλικό όρο ' blockchain technology ¨. Πολλοί χρησιμοποιούν το Bitcoin ως εφαλτήριο για ννα εξηγήσουν τον όρο, από την αρχαιότερη εφαρμογή, τα κρυπτονομίσματα. Πολλοί ορισμοί έχουν δοθεί προσπαθώντας να αποσαφινηστεί πλήρως η ορολογία. Μία επιτροπή μέσα στον οργανισμό ISO που ασχολείτε με το θέμα δεν έχει κάποιο καταληκτικό εποτέλεσμα προς το παρόν αλλά περιγράφει το blockchain ως: ¨ένα κοινό, αμετάβλητο βιβλίο που μπορεί να καταγράφει συναλλαγές σε διάφορες βιομηχανίες, [...]. Πρόκειται για μια ψηφιακή πλατφόρμα που καταγράφει και επαληθεύει τις συναλλαγές με διαφανή και ασφαλή τρόπο, καταργώντας την ανάγκη για μεσάζοντες και αυξάνοντας την εμπιστοσύνη μέσω του ιδιαίτερα διαφανούς χαρακτήρα της' [6]. Η IBM προτείνει έναν παρόμοιο ορισμό λέγοντας ότι ένα blockchain είναι ένας κοινός, αμετάβλητος ηγέτης για την καταγραφή του ιστορικού των συναλλαγών'[7]. Για λόγους απλότητας θα κινηθούμε γύρω από αυτή την περιγραφή και θα θεωρήσουμε πως όλα τα δεδομένα που αποθηκεύονται στην αλυσίδα είναι υπό μορφή συναλλαγών. Για να εμβαθύνουμε λίγο περισσότερο, όμως, πρέπει να διασπάσουμε δομικά την τεχνολογία πρώτα. Αρχικά, λοιπόν, μία αλυσίδα συναλλαγών είναι ένα δίκτυο μεταξύ χρηστών. Αυτοί, μπορούν να χωριστούν σε δύο κατηγορίες, τους πελάτες και τους μεταλλωρύχους(σημαντικό αποτελεί πως αυτός ο διαχωρισμός δεν είναι αποκλειστικός, δηλαδή ένας πελάτης μπορεί να είναι και μεταλλωρύχος και ανάστροφα).

- **Πελάτης:** Ένας χρήστης που χρησιμοποιεί το δίκτυο για να εκτελέσει και να λάβει πληρωμές αναλόγως τις επιθυμίες του.
- **Μεταλλωρύχος:** Ένας χρήστης που προσπαθεί να λύσει ένα πρόβλημα υπο τη μορφή: Δοσμένου του α βρες β ούτως ώστε Γ(β)=α. Εάν βρει τη λύση, τότε αυτός εκπέμπει σε όλο το δίκτυο τη λύση μαζί με το αμέσως δημιουργηθέν block που περιέχει τις συναλλαγές που θέλει να ενσωματώσει ο ίδιος.

Κάθε block από κατασκευής δείχνει στο αμέσως προηγούμενό του και συνεπώς δημιουργείται με αυτόν τον τρόπο η αλυσίδα. Όλοι οι χρήστες λοιπόν, αποδεχόμενοι τη λύση πυο τους εκπέμπεται έχουν μία καθολική όψη της αλυσίδας συναλλαγών και συμφωνούν στην εγκυρώτητά της.

Υπάρχει η περίσπτωση, ωστόσο, ένας κόμβος του δικτύου να έχει μία διαφορετική όψη της αλυσίδαςμ, διότι δύο τουλάχιστον από τους γείτονές του, του εξέπεμψαν διαφορετικές λύσεις και άρα έχει ένα πρόβλημα απόφασης ως προς το ποια λύση θα δεχτεί. Ένας απλός κανόνας υπάρχει για την επίλυση αυτών των διαμαχιών: Κάθε κόμβος συνεχίζει να δουλεύει στην όψη της αλυσίδας που έχει, μέχρις ότου λάβει μία η οποία είναι μεγαλύτερη. Σε αυτή την περίπτωση, σταματά να εργάζεται στην προηγούμενη και συνεχίζει στην τελευταία. Αυτό βεβαιώνει πως σε βάθος χρόνου μία και μόνο μία αλυσίδα θα υπερτερύσει αν η πλειοψηφία των μεταλλωρύχων είναι τίμιοι.

Είναικορυφαίας σημασίας να τονιστεί το ότι οι χρήστες ενός δικτύου καταλήγουν σε βάθος χρόνου σε μία κκοινή αλυσίδα, σημαινεί ουσιαστικά πως καταλίγουν σε μία και μόνο μία κοινή αλήθεια μεταξύ των, που δε μπορεί να αμφισβητηθεί από κανέναν εντός του δικτύου. Έτσι, το πρωτόκολλο κάθε αλυσίδας συναλλαγών είναι, στην πραγματικότητα, ένα πρωτόκολλο συμφωνίας συναίνεσης. Το πιο ενδιαφέρον κομμάτι αυτής της αλήθειας είναι η σκληρότητα της αναστρεψιμότητας της αλυσίδας. Προκειμένου ένας αντίπαλος να αλλάξει ένα από τα block , ας πούμε το κ-τελευταίο block , πρέπει να κάνει όλη την προηγούμενη δουλειά (εύρεση νέων λύσεων κ-φορές) και να βγάλει ολοκαίνουργια $(\kappa + 1)$ block για να πείσει όλους τους άλλους κόμβους να δεχτούν τη δική του αλυσίδα ως τη μεγαλύτερη και επομένως την αληθινή. Εάν δεχθούμε την ειλικρινή πλειοψηφία στο δίκτυο αυτό ισχύει με πιθανότητα $1/2^{k+1}$ και προφανώς, αυτή η πιθανότητα καθίσταται αμελητέα καθώς η υπολογιστική δύναμη του αντιπάλου μειώνεται επίσης ή όσο βαθύτερα θέλει να αλλάξει κάτι στην αλυσίδα. Ουσιαστικά, ο μηχανισμός που εξηγείται παραπάνω, καθιστά πιο δαπανηρή την πλαστογράφηση μιας συναλλαγής από το δυνητικό κέρδος. Χωρίς έναν κατάλληλο αλγόριθμο για την επίτευξη συναίνεσης για το blockchain , δεν θα υπήρχε εμπιστοσύνη στο blockchain-system του Bitcoin , αφού οποιοσδήποτε με πρόσβαση στην ιστορία των συναλλαγών (όλοι οι κόμβοι) θα μπορούσε να ξαναγράψει το ιστορικό και να το δημοσιεύσει ως το πραγματικό.

Επομένως, ο ακόλουθος ορισμός φαίνεται να καλύπτει όλα τα παραπάνω, δηλαδή τις κύριες πτυχές ενός blockchain . Μία αλυσίδα συναλλαγών είναι μια κατανεμημένη αρχιτεκτονική υπολογιστών όπου ένας υπολογιστής ονομάζεται κόμβος εάν συμμετέχει στο δίκτυο. Κάθε κόμβος έχει πλήρη γνώση όλων των συναλλαγών που έχουν συμβεί, οι πληροφορίες μοιράζονται. Οι συναλλαγές ομαδοποιούνται σε blocks που προστίθενται διαδοχικά στην κατανεμημένη βάση δεδομένων. Μόνο ένα block κάθε φορά μπορεί να προστεθεί και για να προστεθεί ένα νέο block πρέπει να περιέχει μια μαθηματική απόδειξη που να επαληθεύει ότι ακολουθεί ακολουθία από το προηγούμενο block . Τα block συνδέονται μεταξύ τους με χρονολογική σειρά. Ο παραπάνω ορισμός είναι πολύ ευρύς και καλύπτει όλες σχεδόν τις υπάρχουσες υλοποιήσεις των αλυσίδων συναλλαγών.

Πλευρικές Αλυσίδες

Οι πλευρικές αλυσίδες είναι η έννοια των παράλληλων αλυσίδων συναλλαγών που επιτρέπουν τη μεταβίβαση περιουσιακών στοιχείων από ένα blockchain σε ένα άλλο. Η ιδέα ανακοινώθηκε επισήμως το 2014 από την Blockstream , μια ιδιωτική εταιρεία που αποτελείται από πολλά από τα μέλη της ομάδας ανάπτυξης του Bitcoin . Ενώ η ιδέα προέρχεται από το Bitcoin , η ευρύτερη θεωρία πίσω από τις πλευρικές αλυσίδες εφαρμόζεται σε οποιοδήποτε σχέδιο αλυσίδας. Αρχικά, η πρόθεση των πλευρικών αλυσίδων ήταν να επιτρέψει τη μεταφορά bitcoins σε άλλα blockchains , επιτρέποντας έτσι στις πλευρικές αλυσίδες να λειτουργήσουν ως εναλλακτικά συστήματα κρυπτονομισμάτων χωρίς να απαιτείται η κοπή νέων κερμάτων. Ωστόσο, ο αρχικός σχεδιασμός πλευρικής αλυσίδας ανακαλύφθηκε ότι περιέχει μη τετριμμένα ελαττώματα ασφαλείας. Εν τω μεταξύ, τα νέα σχέδια βρίσκονται σε εξέλιξη, αλλά κατά τη στιγμή της γραφής δεν έχουν φθάσει σε ωριμότητα ή παραγωγή.[8]

Παρόλα αυτά, η μεταφορά στοιχείων από μία αλυσίδα σε μία άλλη, είναι στην πραγματικότητα ήδη εφικτή από το σχεδιασμό του sidechain . Η ιδέα είναι αρκετά απλή, με την εισαγωγή μιας λειτουργίας που ονομάζεται αμφίδρομη σύνδεση, τα περιουσιακά στοιχεία θα επιτρέπεται να μεταφέρονται από ένα blockchain σε ένα άλλο και πίσω. Μια από τις βασικές αρχές της αμφίδρομης σύνδεσης είναι ότι είναι αδύνατο να επιστραφούν περισσότερα στοιχεία στην «μητρική αλυσίδα» από όσα προέρχονται από αυτήν, οπότε ο συνολικός αριθμός περιουσιακών στοιχείων της μητρικής αλυσίδας δεν μπορεί να τεθεί σε κίνδυνο. Έτσι, νέοι κανόνες μπορούν να εφαρμοστούν στην πλευρική αλυσίδα χωρίς να δημιουργούν ενδεχόμενο πρόβλημα για τη μητρική αλυσίδα.

Οι πλευρικές αλυσίδες μπορούν να επεκτείνουν τη λειτουργικότητα ενός γονικού blockchain εισάγοντας νέα χαρακτηριστικά στην πλευρική αλυσίδα. Για παράδειγμα, επειδή μια εξουσιοδοτημένη αλυσίδα μπορεί να χρησιμοποιεί ένα μερισματικό σχήμα ψηφιακών υπογραφών ως μοντέλο συναίνεσης, αυτό επιτρέπει σχεδόν άμεση επιβεβαίωση χρόνων μίας αρχικά μη εξουσιωδοτημένης αλυσίδας, όπως το bitcoin. Ένα παράδειγμα τέτοιας πλευρικής αλυσίδας είναι αυτό της πλευρικής αλυσίδας Liquid, που διατηρείται από το Blockstream για διάφορες συναλλαγές Bitcoin.[9] Άλλα παραδείγματα χαρακτηριστικών των πλευρικών αλυσίδων που μπορούν δυνητικά να προσφέρουν νεές δυνατότητες σε η εξουσιωδοτημένα δίκτυα είναι στοιχεία όπως η υποστήριξη πολλαπλών τύπων περιουσιακών στοιχείων από διαφορετικές αλυσίδες συναλλαγών που υπάρχουν σε αμοιβαίες πλευρικές αλυσίδες, όπως τα έξυπνα συμβόλαια.

Το Παιχνίδι

Η ρουλλέτα αποτελεί ένα από τα πιο δημοφιλή παιχνίδια σε κάθε καζίνο. Το παιχνίδι έχει μερικούς βασικούς κανόνες και παίζεται σε γύρους. Αρχικά, υπάρχουν 37 νούμερα στα οποία μπορεί κάποιος να στοιχηματίσει και το κάθε ένα έχει απόδοση 36. Αυτό σημαίνει πως ο ιδιοκτήτης του παιχνιδιού έχει πλεονέκτημα έναντι των άλλων παικτών 2,7%. Παιχνίδι παίζεται ως ακολούθως: Οι παίκτες πoντάρουν το ποσό που επιθυμούν, ο ιδιοκτήτης του παιχνιδιού χρησιμοποιεί ένα τρόπο παραγωγής τυχαίων αποτελεσμάτων (παραδείγματος χάριν στο αναλογικό καζίνο το γύρισμα μιας μπίλιας σε ένα λείο,κεκλινόμενο, κυκλικό κουλουάρ, που στο εσωτερικό του έχει 37 διαχωρισμένες θέσεις. Σε όποια από αυτές πέσει η μπίλια, αποτελεί και το αποτέλεσμα του γύρου. Αυτό το παιχνίδι θα προσπαθήσουμε να περιγράψουμε στο πρωτόκολλό μας παρακάτω.

Το Πρωτόκολλο

Το πρωτόκολλο βασίζεται στην παρακάτω ιδέα. Υπάρχει μία αλυσίδα συναλλαγών και όποιος χρήστης της θέλει να δημιουργήσει το δικό του παιχνίδι της ρουλέττας, δημιουργεί μία πλευρική αλυσίδα την Roulettechain . Όποιος θέλει να παίξει στο εν λόγω παιχνίδι, μεταφέρει κεφάλαιο από τη μητρική αλυσίδα στην Roulettechain . Τα blocks δημιουργούνται από τον μόνο μεταλλορύχο της αλυσίδας και δημιουργό της RM . Για να επικοινωνήσουν οι παίκτες τις επιθυμίες τους στέλνουν υπογεγραμμένα τα μηνύματά τους στον RM . Το παιχνίδι παίζεται σε γύρους, όπου ένας γύρος περιγράφεται στο πρωτόκολλο. Υπάρχει ο αλγόριθμος Calcplayers για να υπολογίζονται σε κάθε γύρο οι παίκτες που μπορούν να συμμετέχουν στο πρωτόκολλο. Η τυχαιότητα του αποτελέσματος διασφαλίζεται από μία δίπλευρη πράξη, χρησειμοποιώντας τη συνάρτηση κατακερματισμού SHA256 και τον τελεστή $\oplus$. Επίσης, ένας μηχανισμός ασφαλείας υπάρχει για να είναι βέβαιο πως αν κάποιος από τους παίκτες ξεφύγει από τη ροή του πρωτοκόλλου, θα τιμωρηθεί. Για τα παραπάνω, τέλος, ισχύουν και τρεις υποθέσεις, το ότι υπάρχει τίμια πλειοψηφία στη μητρική αλυσίδα, ότι αυτή έχει ζωτικότητα και ευρωστία καθώς ακόμα πως υπάρχει και ένας τίμιος παίκτης σε κάθε γύρο του προτωκόλλου. Το πρωτόκολλο που έχει ως στοιχεία του τους παίκτες $Player_i$ με τον $Player_0$ να είναι ο RM, το αντίστοιχα κεφάλαιο του καθενός balance[i], τα αιτήματα των παικτών requests[n] τέλος τα πονταρίσματα $bet_i[j]$ που οδηγούν στα αποτελέσματα του γύρου $result_k$[i], παρατίθεται στο παρακάτω σχήμα.

Protocol $\pi_{Roulette}$

Protocol $\pi_{Roulette}$ is executed by n players with identities (id_0 ,. . . , id_n), parameterized by a timeout limit τ and interacting with the stateful contract functionality F_{SC}. We assume that the parties agree on a generator g of a group G of order p for the El-Gamal encrypion scheme EGE and also on a EUF-CMA secure digital signature scheme SIG. Moreover, a nonce unique to each protocol execution and protocol round(e.g. a hash of the public protocol transcript up to the current round) is implicitly attached to every signed message to avoid replay attacks.
Recovery Triggers: Whenever a signature is published, its validity is tested. If the test fails, the party proceeds to the recovery phase. The same happens if a party does not receive an expected message until a timeout limit τ.

Request phase: For i = 0, . . . , n, the party with id_i proceeds as follows:

- Verifies information in the previous block, secretkey[i], that is the other player's secret key as well as $result_k$[i] for the previous (k) round.
- Generates the keys of the signature scheme
  ($SIG.vk_i$, $SIG.sk_i$) ← SIG.Gen (1^λ).
- Uses the keys above forn the encryption scheme, such as,
  $EGE.sk_i = SIG.sk_i$ and $EGE.pk_i = SIG.vk_i$
- Sends:
  - **Check-in:** (*Check-in,M, walletadress*$_i$ $SIG.vk_i$), where M is the maximum sum of money intended to be played for the round and $walletaddress_i$ points to the transaction of a block in Roulettechain from where the money would be bet, if she wants to play.

Figure 1.2: Protocol $\pi_{Roulette}$

- Or else:
  - **Check-out:** (*Check-out, balance, σ*) to F_{SC}, where σ contains all signatures on balance, waits for confirmation from F_{SC} and stops execution, otherwise.
- RM, that is, $Player_0$ with id_0 publishes request[i] on the sidechain.

Calculating plyers phase:

- Each player verifies information in the previous block, that is the other player's request.
- $Player_0$ runs Algorithm *CalcPlayers* to determine the players of the round P_i and publishes them on the sidechain alongside her queue[0].

Betting phase:

- Each player verifies information in the previous block, that is players for the round and queue[0].
- Each $Player_i$ updates their queue[i].
- Each player chooses a randomness r and sends her $bets_i$[j]= $Enc_r(m_{ij})$ where m includes information about the number intended to be bet on alongside the respective sum of money.
- $Player_0$ publishes $bets_i$[j] on the sidechain.

Tallying result phase:

- Each player verifies information in the previous block, that is the $bets_i$[j].
- Each player sends his EGE.sk_i to $Player_0$.
- $Player_0$ decrypts $bets_i$[j] and calculates $result_k$[i] = $(SHA_{256}(\sum_{j=1}^{m}(\text{Dec}(Enc_r(m_{i,j})) \oplus (\text{Dec}(Enc_r(m_{0,j})))) \bmod 37$.
- $Player_0$ publishes $result_k$[i] and secretkey[i] = EGE.sk_i on the sidechain .

Recovery request: If a party P_i enters the recovery phase at any step of a given phase, it halts the execution of the protocol and sends a message (report, $Player_i$, sk_i, $Block_z$) to F_{SC} , where $Block_i$ is the block that contains information on which the verification test failed and thus contains forged or false data.

Figure 1.3: Protocol $\pi_{Roulette}$ (continuation)

Γενίκευση πρωτοκόλλου σε άλλα παιχνίδια

Σε αυτή την ενότητα συζητάμε άλλα παιχνίδια που μπορούν να εκτελεστούν από πρωτόκολλα που έχουν κατασκευαστεί με τον ίδιο τρόπο, δηλαδή έχοντας μία αλυσίδα συναλλαγών από ένα μπλοκ του οποίου ένας χρήστης ξεκινά μια πλευρική αλυσίδα, εκτελώντας το παιχνίδι σε ένα πρωτόκολλο εκτός αλυσίδας και αποθηκεύοντας μόνο ζωτικής σημασίας πληροφορίες σχετικά με αυτό πάνω της. Με τον σχεδιασμό αυτόν, το πλησιέστερο παιχνίδι στη ρουλέτα είναι οι λαχειοφόρες αγορές, επειδή και οι τα δυο μοιράζονται τους ίδιους μηχανισμούς με τις πιο διαφορετικές πτυχές τους να αποτελούν οι πληρωμές των παικτών. Θα μπορούσε κανείς να υποστηρίξει ότι η ρουλέτα είναι μια ξεχωριστή περίπτωση παιχνιδιών λαχειοφόρων αγορών και αυτό δεν θα ήταν αναληθές. Αυτό σημαίνει πρακτικά, πως με την αλλαγή μόνο τον τρόπο παραγωγήες αποτελεσμάτων και του αλγορίθμου για τον υπολογισμό των παικτών, σύμφωνα με τον αριθμό των επιλογών που έχουν οι παίκτες και τη μέγιστη απόδοση κάθε στοιχήματος, μπορεί κανείς να κατασκευάσει το πρωτόκολλο λαχειοφόρου αγοράς της επιλογής τους χωρίς να χρειάζεται να αλλάξει τίποτα άλλο από το σχεδιασμό μας.

Τα ζάρια μπορούν επίσης να θεωρηθούν ένα παιχνίδι κοντά στη ρουλέτα. Στην πραγματικότητα, η μόνη διαφορά που έχουν είναι αυτή της έγχυσης τυχαιότητας στο παιχνίδι. Εφ ΄όσον κάποιος τροποποιεί ανάλογα των αποδόσεων και τις επιλογές των παικτών θα είναι σε θέση να εκτελέσει το παιχνίδι των ζαριών επίσης.

Τέλος, μια άλλη εφαρμογή αυτού του τύπου πρωτοκόλλου είναι σε συγκεκριμένα παιχνίδια τράπουλας. Αυτό οφείλεται στο γεγονός ότι τα παιχνίδια ¨ένας εναντίων ενός' συναντόνται συνήθως στα παιχνίδια τράπουλας και από το γεγονός ότι χρησιμοποιώντας μια συμβολοσειρά 256 δυαδικών ψηφίων μπορούν να αναπαρασταθούν όλες οι ανακατεμένες τράπουλες ($52! < 2^{256}$). Το blackjack είναι ένα παιχνίδι που θα μπορούσε να εκτελεστεί από το πρωτόκολλό μας με κάποιες παραπάνω μικροαλλαγές, οι σημαντικότερες των οποίων είναι ότι θα χρειαζόταν μια φάση ρυθμίσεων για το ανακάτεμα της τράπουλας μετά τη φάση υπολογισμού των παικτών και μια φάση εκτέλεσης χεριού μετά από αυτό. Φυσικά ο αλγόριθμος και οι πληρωμές θα πρέπει επίσης να διορθωθούν.

Συμπεράσματα και μελλοντική εργασία

Σε αυτή τη διπλωματική παρουσιάστηκε ένα νέο πρωτοποριακό πρωτόκολλο για το παιχνίδι της ρουλέτας. Συγκεκριμένα, χρησιμοποιήθηκε μια συνδεδεμένη πλευρική αλυσίδα στην οποία αποθηκεύτηκαν κρίσιμες πληροφορίες σχετικά με την εκτέλεση του παιχνιδιού μαζί με έναν μηχανισμό ανάκτησης για την επιβολή κυρώσεων στις αποκλίσεις παικτών από το πρωτόκολλο. Όλα τα αποτελέσματα βασίζονται στο επιχείρημα της μη διακρισιμότητας, για παράδειγμα, οι παίκτες περιορίζονται στο τι μπορούν να διακρίνουν ή αν το θέσουμε θετικά οι παίκτες χρειάζονται αρκετές πληροφορίες για να είναι σε θέση να διακρίνουν το σενάριο στο οποίο βρίσκονται.

Μια σειρά ανοικτών ερωτήσεων προκύπτουν από αυτή την εργασία και περιγράφονται παρακάτω:

- Υλοποίηση του πρωτοκόλλου σε κρυπτοοικονομικό σύστημα μέσω έξυπνων συμβολαίων.

- Γενίκευση του πρωτοκόλλου ώστε να μπορεί να εκτελεί γενικά 'ένας εναντίον ενός' τυχερά παιχνίδια.

- Βελτιστοποίηση του μηχανισμού για τη μεταφορά περιουσιακών στοιχείων από μία αλυσίδα συναλλαγών σε μία άλλη.

Chapter 2

Introduction

Technology has always played an important role in human history. Whole eras were named after the technological advancements that shaped them. For example tool making technologies defined three large periods of human history, the stone, bronze and iron age. It is no exaggeration to say that for the last twenty years we are living in the age of the Internet. However, as with any technological achievement, whether its use is for better or for worse, is entirely determined by the ones who use it. Internet is not an exception to this rule. In the past few years, it was made clear, especially after the Snowden leaks that through the Internet one can achieve massive privacy violations with little effort. Therefore, it is not only in our hands to guard and support the fair use of this technology, but our responsibility as well to keep advancing its security in new, innovative ways that will ultimately improve one's everyday life. This work is towards this direction and tries to explore and combine in a unique way, some new capabilities of the Internet that were only in the sphere of our imagination till the last 10 years.

The milestone mentioned above is no other than the invention of the blockchain. Its impact has been growing exponentially since the first blockchain -the Bitcoin- was released in 2009 by Satoshi Nakamoto and has gathered the attention of numerous scholars as well as entrepreneurs. As of 2014 more and more companies are exploring the option of adopting blockchain technology in order to boost the security of some or even every aspect of their services. IBM, Walmart, Maersk, British Airways and UPS are some of the early innovators that have already adopted blockchains and numerous others are on their way, no matter the field of their expertise. Blockchain is best known from and associated closely with the Bitcoin, so one may naturally assume that a blockchain is just a decentralized payment system. In reality though, its most favored aspect -and the one industry is greatly interested in- is that it maintains a public transaction ledger in a distributed manner, in fact consisting the communication protocol of a blockchain a decentralized consensus agreement protocol. Conclusively, this means that there exists a network where its nodes have limited trust over each other and yet they can agree on a common truth.

Motivation

A case where trust is almost considered a prerequisite is gambling. While using a service either analog or digital the user is trusting the gaming company at some level to execute the game randomly and with no bias towards him. In the case of analog casinos or even house games total trust is required. The same holds for e-gambling platforms that do not use blockchain as the digital mechanism used for the random generation of the numbers is not publically available for profitability reasons and as a result, the randomness of the outcome cannot be verified from the user.

Blockchain's consensus agreement protocol is bringing new light in some obscure problems concerning communication and provides new opportunities to boost security in some others. Namely, the mental poker problem which was stated in 1979 by A.Shamir et all. was not solvable up until lately by B.David et all. in 2018. Although this, previous work, is a breakthrough in cryptographic protocols, its implementation is in the static scenario and covers only card games. Furthermore, online casinos using blockchain like FunFair, Dao or Sp8de have blockchain implemented in their services but they require fees in addition to the gambling bet of the players. It came to our attention, thus, that there exists no decentralized, zero-trust, zero-fee, cryptographic protocol for games such as roulette or lotteries, especially in the dynamic scenario.

In this work we will construct a blockchain based cryptographic protocol to execute the game of roulette. We will need a blockchain on which we shall build a pegged sidechain. The communication of the players is being done over the insecure private network of the sidechain in which crucial information is stored and message verification is achieved through a secure signature scheme. The outcome of every round is being determined by a commitment scheme between the players and there exists a recovery mechanism for correcting wrongdoings, when a player deviates from the protocol. In addition, the setup of the table for every round is calculated by an algorithm to ensure maximum resources on the table at any given round, while respecting priority of sitting. Finally there will be no fees needed when someone intends to participate in any game or intends to host a game. Therefore the scope of this work is to introduce a fully-decentralized, zero-trust, zero-fee cryptographic protocol which executes a game of roulette, providing options with respect to gambling.

Outline

The rest of the thesis is organized as following. Chapter 2 analyzes thoroughly both the "blockchain" and its "pegged sidechain". Focus will be given into their constructions, their uses as well as their interoperability. Moving on, in Chapter 3 cryptographic protocols are being introduced, their designing issues and security properties are explored. Moreover, the adversary model is being analyzed and lastly the differences between on-chain and off-chain protocols. Chapter 4 contains our construction which executes the game of roulette and its generalization to other games. More precisely, we will see who interacts with the protocol, the progress of the idea behind the protocol, the protocol π_{Roulette} and how this protocol with some twists can execute other games as well. Finally, in Chapter 5 conclusions and future work are stated.

Chapter 3

Cryptographic background

Introduction

The roots of cryptography can be traced back to thousands of years. When Julius Caesar sent messaged his generals, he didn't trust his messengers. So he replaced every A in his messages with a D, every B with an E, and so on through the alphabet. Only someone who knew the "shift by 3" rule could decipher his messages. So the types of data can be distinguished from the previous example. Data that can be read and understood without any special measures is called plaintext or cleartext. The method of disguising plaintext in such a way as to hide its substance is called encryption. Encrypting plaintext results in unreadable gibberish called ciphertext. You use encryption to make sure that information is hidden from anyone for whom it is not intended, even those who can see the encrypted data. The process of reverting ciphertext to its original plaintext is called decryption.

Cryptography is the science of using mathematics to encrypt and decrypt data. Cryptography enables you to store sensitive information or transmit it across insecure networks (like the Internet) so that it cannot be read by anyone except the intended recipient. While cryptography is the science of securing data, cryptanalysis is the science of analyzing and breaking secure communication. Classical cryptanalysis involves an interesting combination of analytical reasoning, application of mathematical tools, pattern finding, patience, determination, and luck. Cryptanalysts are also called attackers.

Cryptography has been advanced through the ages little by little, but since the invention of the Internet its advancement has been exponential. Some of the cryptographic primitives developed till the time of the writing of this thesis are explained below and will be crucial in understanding our construction.

Discrete logarithm

We shall now state three assumptions that are commonly used in cryptographic schemes based on a discrete log setting.

- The **Discrete Logarithm (DL) assumption** for group $\langle g \rangle$ states that it is hard to compute x given a random group element g^x .

- The **Diffie-Hellman (DH) assumption** for group ⟨g⟩ states that it is hard to compute g^xy given random group elements g^x and g^y.
- The **Decisional Diffie-Hellman (DDH) assumption** for group ⟨g⟩ states that it is hard to distinguish g^xy from a random group element g^z given random group elements g^x and g^y.

The DH assumption is sometimes called the Computational Diffie-Hellman (CDH) assumption to stress the difference with the DDH assumption. Evidently, these assumptions satisfy: DL $\Leftarrow$ DH $\Leftarrow$ DDH.[10] Therefore, it is better if a scheme can be proved secure under just the DL assumption.[4] It turns out, however, that in many cases the security can only be proved under the DH assumption, or even only under the DDH assumption.

Diffie-Hellman key exchange

The Diffie-Hellman key exchange protocol enables two parties A and B to arrive at a shared key K by exchanging messages over a public channel. Key K remains unknown to any eavesdropper. The protocol runs as follows. Suppose parties A and B have agreed upon a group $G_n = \langle$ g$\rangle$, where we require n to be prime. Party A picks a value $x_A \epsilon Z_n$ uniformly at random, and sends $h_A = g^{x_A}$ to party B. Similarly, party B picks a value $x_B \epsilon Z_n^*$ uniformly at random, and sends $h_B = g^{x_B}$ to party A. Upon receipt of h_B , party A computes key $K_{AB} = h_B^{xA}$. Similarly, party B computes $K_{BA} = h_A^{xB}$. Clearly, K = $K_{AB} = K_{BA}$ is a shared key for A and B, which means (i) that it is the same for A and B, and (ii) that it is a private key (only known to A and B), and (iii) that the key is actually equal to $g^{x_A x_B}$, hence uniformly distributed.

A passive attacker (eavesdropper) learns the values $h_A = g^{x_A}$ and $h_B = g^{x_B}$. Under the DL assumption, it is hard to determine x_A and x_B from h_A and h_B , respectively. However, this does not guarantee that the value of K = $h_A^{x_B}$ cannot be determined given just h_A and h_B . To exclude this possibility we need the DH assumption. A stronger assumption is needed to ensure that an eavesdropper does not learn any information whatsoever on K. In general, an eavesdropper may learn some partial information on K, while full recovery of K is infeasible. For example, an eavesdropper might be able to determine the parity of K, viewing K as an integer, which would mean that the eavesdropper learns one bit of information. To exclude such possibilities we need the DDH assumption.

The Diffie-Hellman protocol only withstands passive attacks. A first, but general, idea to obtain a key exchange protocol withstanding active attacks is to authenticate the communication between A and B. For instance, we may assume that A and B know each other's public keys in a digital signature scheme. There are many solutions to this problem, but only few have been proved correct [11]. We will not give a formal security analysis at this point. The protocol is as follows. Party A picks $x_A \epsilon Z_n^*$ uniformly at random, and sends $h_A = g^{x_A}$ along with a signature on (h_A , B) to party B. Similarly, party B picks $x_B \epsilon Z_n^*$ uniformly at random, and replies with $h_B = g^{x_B}$ along with a signature on (h_A, h_B, A) to party A. As before, the agreed upon key is K = $g^{x_A x_B}$. A protocol of this type is secure under the DDH assumption, also assuming that the digital signature scheme is secure.

Commitment schemes

The functionality of a commitment scheme is commonly introduced by means of the following analogy. Suppose you need to commit to a certain value, but you do not want to reveal it right away. For example, the committed value is a sealed bid in some auction scheme. One way to do this is to write the value on a piece of paper, put it in a box, and lock the box with a padlock. The locked box is then given to the other party, but you keep the key. At a later time, you present the key to the other party who may then open the box, and check its contents.

An immediate application of commitment schemes is known as "coin flipping by telephone." Two parties, say A and B, determine a mutually random bit as follows. Party A commits to a random bit $b_A \epsilon$ R $\{0,1\}$ by sending a commitment on b_A to party B. Party B then replies by sending a random bit $b_B \epsilon$ R $\{0,1\}$ to A. Finally, party A opens the commitment and sends b_A to B. Both parties take b=$b_A \oplus b_B$ as the common random bit.

If at least one of the parties is honest, the resulting bit b is distributed uniformly at random, assuming that A and B cannot cheat when revealing their bits. Note that party B sees the commitment of A before choosing its bit b_B , so no information on bit b_A should leak from the commitment on b_A. Similarly, party A could try to influence the value of the resulting bit b (after seeing the bit b_B) by opening the commitment on b_A as a commitment on $1{-}b_A$. Clearly, party A should not be able to "change its mind" in such a way.

Generating mutually random bits is a basic part of many protocols. Commitments are used as an auxiliary tool in many cryptographic applications, such as zero-knowledge proofs and secure multi-party computation. [11]

A commitment scheme consists of two protocols, called commit and reveal, between two parties, usually called the sender and the receiver. In many cases, the protocols commit and reveal can be defined in terms of a single algorithm, requiring no interaction between the sender and receiver at all. Such commitment schemes are called non-interactive. More precisely, by definition:

Let commit : $\{0,1\}^k\{0,1\}^* \Rightarrow \{0,1\}^*$ be a deterministic polynomial time algorithm, where k is a security parameter. A (non-interactive) commitment scheme consists of two protocols between a sender and a receiver:

Commit Phase. A protocol in which the sender commits to a value x $\epsilon\{0,1\}^*$ by computing C = commit(u, x), where u $\epsilon R\{0,1\}^k$, and sending C to the receiver. The receiver stores C for later use.

Reveal Phase. A protocol in which the sender opens commitment C = commit(u, x) by sending u and x to the receiver. The receiver computes commit(u, x) and verifies that it is equal to the previously received commitment.

In the special case that the committed value is a bit, that is, x $\epsilon R\{0,1\}$, one speaks of a bit commitment scheme. The security requirements for a bit commitment scheme are the following. The commitment must be binding, i.e., for any adversary E, the probability of generating u, u' $\epsilon R\{0,1\}^k$ satisfying commit(u, 0) = commit(u' , 1) should be negligible (as a function of k). Furthermore, the commitment must be hiding, i.e., the distributions induced by commit(u, 0) and commit(u, 1) (with u $\epsilon R\{0,1\}^k$) are indistinguishable.

Moreover, one makes the following distinctions. A commitment scheme is called computationally binding if the adversary E is restricted to be a p.p.t. algorithm. If no such restriction is made (in other words, the adversary may be unlimitedly powerful), the scheme is called information-theoretically binding. Similarly, if the distributions induced by commit(u, 0) and commit(u, 1) are computationally indistinguishable the scheme is called computationally hiding and the scheme is called information-theoretically hiding if these distributions are statistically (or even perfectly) indistinguishable.

The security properties are easily extended to the case that x is an arbitrary bit string. Note that the above security requirements only cover attacks by either the sender or the receiver. For example, suppose party A acts as the sender and party B acts as the receiver, and A sends a commitment C to B. Then there is no guarantee that B will notice if an attacker replaces C by a commitment C' = commit(u' , x') during the commit protocol, and replaces u, x by u' , x' during the reveal protocol. Such attacks may be stopped by using an authenticated channel between A and B. [11]

A natural question is whether there exists a commitment scheme which is both information-theoretically binding and information-theoretically hiding. The following informal argument shows that such a scheme cannot exist.

Consider any bit commitment scheme which is information-theoretically binding. For such a scheme there cannot exist any u, u' such that commit(u, 0) = commit(u' , 1), because then the (unlimitedly powerful) sender would be able to compute both u and u' , and open the commitment at its liking. However, if the sender commits to 0, say, using C = commit(u, 0) for some u, the (unlimitedly powerful) receiver will notice, by exhausting the finite set of possibilities, that there does not exist any u' with C = commit(u', 1), hence the receiver knows that the committed bit must be 0.

Zero Knowledge Proofs

In mathematics and in life, we often want to convince or prove things to others. Typically, if one knows that X is true, and she wants to convince another person of that, she tries to present all the facts she knows and the inferences from those facts imply that X is true. A simple example of that is the following:

Let's suppose that there exists to entities Alice and Bob and Alice wants to prove to Bod that she knows that 26781 is not a prime. The knowledge of that derives from the factoring she knows, that is 113 times 237 equals 26781. So she presents these factors to Bod and thus, demonstrates that she knows a factoring pair of 26781.

Now, a typical byproduct of a proof is that you gained some knowledge, other than that you are now convinced that the statement is true. In the example before, not only are you convinced that 26781 is not a prime, but you also learned its factorization.

However, A zero knowledge proof tries to avoid it.[12] In a zero-knowledge proof Alice will prove to Bob that a statement X is true, Bob will completely convinced that X is true, but will not learn anything as a result of this process. That is, Bob will gain zero knowledge.

Zero knowledge proofs were invented by Goldwasser, Micali and Rackoff (GMR) [13]. Zero-knowledge proofs (and interactive proofs in general, also introduced in that paper) turned out to be one of the most beautiful and influential concepts in computer science, with applications ranging from practical signature schemes to proving that many NP-complete problems are hard even to approximate.

The motivation behind their development was double: One was philosophical: the notion of a proof is basic to mathematics and to people in general. It is a very interesting quesiton whether a proof inherently carries with it some knowledge or not. The other one is practical: zero knowledge proofs have foundmany applications. Most practical applications fall into two types:

Protocol design: A protocol is an algorithm for interactive parties to achieve some goal. For example, we saw the Diffie-Hellman key exchange protocol. In that protocol, we assume that both parties follow the instructions of the protocol, and the only thing we worried about was a passive easvesdropping adversary Eve. However, in crypto we often want to design protocols that should achieve security even when one of the parties is "cheating" and not following the instructions. This is a hard1e problem since we have no way of knowing the exact way the party will cheat. One way to avoid cheating is the following: If Alice runs a protocol with Bob, to show Bob she is not cheating she will send Bob all the inputs she had, and then Bob can verify for himself that if one runs the prescribed instructions on these inputs, you will indeed get the outputs (messages) that Alice sent. However, this way will be often unacceptable to Alice: the only reason they are running this protocol is that they don't completely trust each other, and the inputs she had may be secret, and she does not want to share them. Zero-knowledge offer a solution to this conundrum. Instead of sending her inputs, Alice will prove in zero knowledge that she followed the instructions. Bob will be convinced, but will not learn anything about her inputs he did not know before. In fact, we will see that it is possible to do this in a very general way, applying essentially to all cryptographic protocols. Thus, a general technique (invented by Goldreich, Micaliand Wigderson , GMW) is to design a cryptographic protocol first assuming everyone will follow the instructions, and then "force" them to follow instruction using a zero knowledge proof system.

Identification scheme: A somewhat simpler and more direct application is to identification schemes. Suppose that we want to control access to a database. One way to do that is to give authorized people a secret PIN number, and have a box on the door where type the PIN number on that box. (A more convenient but essentially equivalent way is that the authorized people have a card that transmits the PIN number to the box.) A drawback of this approach is that the box remains outside all the time, and if someone could examine the box, they would perhaps be able to view its memory and extract the secret keys of all people. Thus, from a security standpoint, it is much better if the box contains no secret information at all, and even if someone installed a "fake box" they would not learn anything about the secret PIN. Zero-knowledge proofs help us in the following way.

- Have the box contain an instance of a hard problem. For example, the box can contain a composite number n without its factorization.
- Give the authorized people the solution to the instance. For example, they can get the factorization of n to $n = p * q$.
- The authorized people will prove to the Box they know the factorization in zero knowledge. (Of course, there is a question of how do you prove that you know something, but this was also shown by GMR (and further developed by others.)

Zero knowledge is an elusive concept in the sense that not only it's not clear how to construct such things, it's also not clear even how to define such creatures. We will start by explaining some of the generalizations to the notion of proofs that are needed. Then, we will give an example for a zero knowledge proof for a particular family of statements (or in more standard terms, for a particular language). We will then talk about the definition of zero knowledge proofs. Next lecture we will see that the extremely useful fact, shown by GMW, that any NP-statement can be proven in zero knowledge.

The standard mathematical notion of a proof is the following: you have axioms and inference rules, and the proof for x is a string π that derives x from the axiom using the inference rules.

- A proof system is sound if you can never derive false statements using it. Soundness is a minimal condition, in the sense that unsound proof systems are not very interesting.
- A proof system is complete if you can prove all true statements using it. Similarly, we say it is complete for a family L of true statements, if you can prove all statements in L using it.

Thus the traditional notion assumes that the proof π is a static string that was written down somewhere and anyone can verify. A valid proof gives absolute certainty that the statement is true. GMR generalized this notion to think of a proof as a game between a prover and a verifier. The game can be interactive, where the verifier asks questions and the prover answers, and the goal of the game is for the prover to convince the verifier that the statement is true. They even further generalized it to the notion of a probabilistic proof system. That is, the verifier does not convinced with absolute certainty that the statement is true but "only" with 99.999 %certainty. What is crucial here is that no matter what the prover does and how she tries to cheat, if the statement is false she will fail with this probability. One example for a probabilistic interactive proof is proving that Alice can distinguish between Coke and Pepsi using the following protocol: Alice turns her back, Bob flips a coin and puts either Coke or Pepsi into a paper cup according the result, Alice tastes and announces whether she thinks it was Coke or Pepsi. If they repeat this k times and Alice always answers correctly then Bob can conclude with $1 - 2^{-k}$ probability that she really can tell the difference. [14]

El-Gamal encryption scheme

In 1984 Taher ElGamal presented a cryptosystem which is based on the Discrete Logarithm Problem discussed in the last section.[15] It relies on the assumption that the DL cannot be found in feasible time, while the inverse operation of the power can be computed efficiently. The original public key system proposed by Diffie and Hellman requires interaction of both parties to calculate a common private key. This poses problems if the cryptosystem should be applied to communication systems where both parties are not able to interact in reasonable time due to delays in transmission or unavailability of the receiving party. Thus ElGamal simplified the Diffie-Hellman key exchange algorithm by introducing a random exponent k. This exponent is an replacement for the private exponent of the receiving entity. Due to this simplification the algorithm can be used to encrypt in one direction, without the necessity of the second party to take actively part. The key advance here is that the algorithm can be used for encryption of electronic messages, which are transmitted by the means of public store-and-forward services. In this section, the ElGamal cryptosystem will be introduced to the reader.

Firstly, the key generation mechanism is explained. As discussed above, the basic requirement for a cryptographic system is at least one key for symmetric algorithms and two keys for asymmetric algorithms. The key generation steps are similar to the general scheme explained above. With ElGamal, only the receiver needs to create a key in advance and publish it. Following our naming scheme from above, we will now follow Bob through his procedure of key generation. Bob will take the following steps to generate his keypair:

- **Prime and group generation:** First Bob needs to generate a large prime p and the generator g of a multiplicative group Z_p^* of the integers modulo p.
- **Private key selection:** Now Bob selects an integer b from the group Z by random and with the constraint $1 \leq b \leq p - 2$. This will be the private exponent.
- **Public key assembling:** From this we can compute the public key part g b mod p. The public key of Bob in the ElGamal cryptosystem is the triplet (p, g, g^b) and his private key is b.
- **Public key publishing:** The public key now needs to be published using some dedicated keyserver or other means, so that Alice is able to get hold of it.

Now, the encryption scheme works as follows. To encrypt a message M to Bob, Alice first needs to obtain his public key triplet (p, g, g^b) from a key server or by receiving it from him via unencrypted electronic mail. There is no security issue involved in this transmission, as the only secret part, b, is sent in g^b . Since the core assumption of the ElGamal cryptosystem says that it is infeasible to compute the discrete logarithm, this is safe.

For the encryption of the plaintext message M , Alice has to follow these steps:

- **Obtain the public key:** As described above, Alice has to acquire the public key part (p, g, g^b) of Bob from an official and trusted keyserver.

- **Prepare M for encoding:** Write M as set of integers $(m_1, m_2, \dots)$ in the range of $\{1, \dots, p-1\}$. These integers will be encoded one by one.
- **Select random exponent:** In this step, Alice will select a random exponent k that takes the place of the second party's private exponent in the Diffie-Hellman key exchange. The randomness here is a crucial factor as the possibility to guess the k gives a sensible amount of the information necessary to decrypt the message to the attacker.
- **Compute public key:** To transmit the random exponent k to Bob, Alice computes g k mod p and combines it with the ciphertext that shall be sent to Bob.
- **Encrypt the plaintext:** In this step, Alice encrypts the message M to the ciphertext C. For this, she iterates over the set created in step 2 and calculates for each of the m_i:

$$c_i = m_1 * (g^b)^k$$

The ciphertext C is the set of all $c_i with 0 \leq i \leq |M|$. The resulting encrypted message C is sent to Bob together with the public key g^k mod p derived from the random private exponent. Even if an attacker would listen to this transmission, and in a second step would also acquire the public key part g b of bob from a keyserver, he would still not be able to derive $g^b * k$ as can be seen from the Discrete Logarithm problem. [16] Elgamal advises to use a new random k for each of the single message blocks m_i . This greatly improves security, as knowledge of one message block m_j does not lead the attacker to the knowledge of all other m i . The reason for this ability is that if $c_1 = m_1 * (g^b)^k mod p and c_2 = m_2 * (g^b)^k mod p$, from knowing only m_1 the next part of the message m_2 can be calculated by the following formula:

$$\frac{m_1}{m_2} = \frac{c_1}{c_2}$$

After receiving the encrypted message C and the randomized public key g k , Bob has to use the encryption algorithm to be able to read the plaintext M . This algorithm can be divided in a few single steps:

- **Compute shared key:** The ElGamal cryptosystem helped Alice to define a shared secret key without Bobs interaction. This shared secret is the combination of Bobs private exponent b and the random exponent k chosen by Alice. The shared key is defined by the following equation:

$$(g^k)^{p-1-b} = (g^k)^{-b} = b^{-bk}$$

- **Decryption:** For each of the ciphertext parts c_i Bob now computes the plaintext using:

$$m_i = (g^k)^{-b} * c_i mod p$$

After combining all of the m_i back to M he can read the message sent by Alice.

Digital Signatures

As stated before cryptography is not explicitly restricted to encryption. One of its other forms is authentication, which is another fundamental property of cryptographic schemes. The way to achieve authentication mostly lies with digital signatures. Devising a digital signature requires a combination of information concerning the message itself and that of the signer. Because digital information can be cut copied and pasted, there should be a link between the message and the digital signature itself.[4][5] Otherwise, the recipient could modify the message before showing the message-signature pair to a judge. Even worse, he could attach the signature to any message whatsoever, since it is impossible to detect electronic 'cutting' and 'pasting'.[17] The following desirable properties can therefore be deduced from a digital signature:

- The signature is message dependent.
- Only the originator of an electronic message can compute the correct digital signature.
- Anyone who receives a message and a digital signature can verify the signature and consequently be certain of the origin and integrity of the message.

This is were the one-way function enters. The one-way function can make a unique fingerprint of a message. This unique fingerprint is then encrypted with the trap-door one-way function. This is called the digital signature. The signature is message dependent because of the unique digital fingerprint and it is uniquely bound to the issuer because of the encryption with the unique private key.

For our scenarios we suppose that A and B (also known as Alice and Bob) are as the encryption key of X, two users of a public-key cryptosystem. If we denote E_x then we will distinguish the encryption and decryption procedures of A and B with E_A, D_A, E_B, D_B[18].

If Alice wants to send a signed message to Bob, then the digital signature, is computed as follows:

- Of the plaintext a unique fingerprint is calculated with the help of a one-way function, h(p).
- The result of the calculation is then "signed" with her private key D_A. Thus, the digital signature now consist of: D_A(h(p)).
- Alice then sends the plaintext, D_A along with her signature, D_A(h(p)) , to Bob.

Bob will do the following after receiving the message:

- Of the plaintext he will calculate the unique fingerprint with the help of the same one-way function, h(p).
- He will then decrypt the unique fingerprint with the help of the public key of Alice, E_A, which is available in the public file, $E_A(D_A(h(p)))$.

- Bob will then compare the value of his calculation with the value calculated by Alice. If the two match then the message has been signed by Alice and the message is unaltered.

Alice cannot later deny having send message, P, because only she could have signed the message. Furthermore, she, or anyone else, can not modify plaintext P because a message P' would produce a different signature. Bob on the other hand cannot use the signature for any other message because it is unique.

Chapter 4

Blockchain Technology

Introduction

There exists, to the knowledge of the author, no single, formal definition of blockchain technology, which is generally accepted. Many use Bitcoin as a starting point, explaining blockchain technology by its first application, cryptocurrency. However, there are systems that are not captured very well by that definition and still are generally classified as blockchains. As for alternative definitions there is one by Vitalik Buterin, the founder of Ethereum: " a blockchain is a magic computer that anyone can upload programs to and leave the programs to self-execute, where the current and all previous states of every program are always publicly visible and which carries a very strong cryptoeconomically secured guarantee that programs running on the chain will continue to execute in exactly the way that the blockchain protocol specifies." [6]. The definition of blockchain made by Buterin is not very rigorous or technical and is certainly not identical to Bitcoin, but manages to include many characteristics of blockchain systems. An attempt to classify different blockchain technologies was made by Okada, Yamasaki and Bracamonte in 2017[19], but it fails to provide a satisfactory definition. A technical committee has been formed within ISO to define areas for standardisation [20]. They have yet to publish a formal definition but do describe the blockchain as: a shared, immutable ledger that can record transactions across different industries, [...]. It is a digital platform that records and verifies transactions in a transparent and secure way, removing the need for middlemen and increasing trust through its highly transparent nature. IBM proposes a similar definition saying that a blockchain is a shared, immutable ledger for recording the history of transactions[7].

All the above definitions cover some but not all, not even the most, aspects of a blockchain. For simplistic reasons we assume, for now, that all data stored in a blockchain are in the form of transactions. To gain a better insight on what it is and how it operates one must break it down. First and foremost, a blockchain is a P2P network. Its users can be separated into two categories, the clients and the miners (keeping in mind that a node may be both at different times, according to her wishes).

- **Clients:** A user that is using the network to execute and receive payments acoording to her wishes. In order to make a transaction, the sum is needed alongside the addresses of the sender and the receiver. Moreover, an additional fee is required that will be given to the miner for including the desired transaction into her freshly mined block.

- **Miner:** A user that is trying to solve an one-way function (OWF). This means that miners are given a value y=f(x), where f is an OWF and they are trying to find x. This is being done by brute forcing inputs into the OWF until the desirable output is presented or until receiving the solution from another node of the network. If a solution is found, the miner broadcasts it to the whole network as well as the newly minted block which contains the transactions she wished to include in it. Should the network accept this block, then she gets a reward in addition to the fees from the transactions that were included.

Each minted block, by design, points to its immediate ancestor, thus forming the "chain of blocks", or rather the blockchain. All users conclusively upon receiving the solution and the data stored in the latest block, have a universal understanding of the blockchain and are agreeing to its validity.

There is a case though when a node of the network has a view of the blockchain and is receiving a different one from some of her neighbors. This is leading to a conflict because each node must decide which one trully is "the blockchain". To dissolve this dispute a simple rule is exercized: each node keeps working on the view of the chain she has, unless she receives one which is longer. In that case she throws the former away and starts working on the latter. This ensures that in the long run one and only chain will remain if and only if the majority of the nodes are so-called honest.

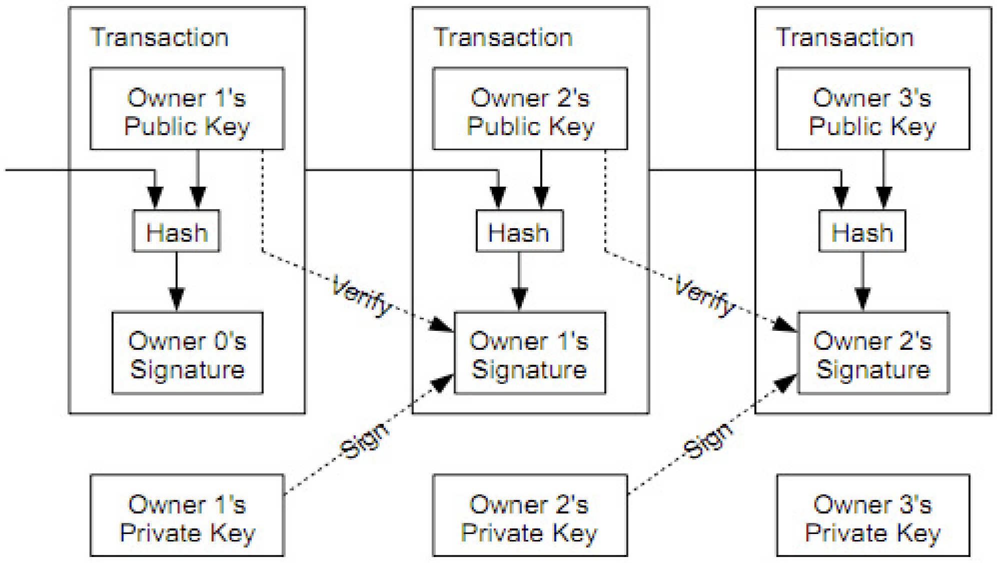
Another substantial aspect of the blockchain ecosystem is the payment of the miners. Block reward as well as average transactions fees differ from chain to chain and their values are generally not constant through time. For example, Bitcoin's block reward was at the first block 50BTC and now is at 12.50BTC with their respective average transaction fees 0.0001BTC and 0.7640BTC with an all-time maximum of 55.16BTC at 22/12/17. Ethereum also has fluctuations with block reward being at start 5.01163ETH and now 4.35727ETH with their average transaction fees being 0.0566 and 5.528ETH which is also the all-time maximum for this blockchain.

Finally, of utmost importance is the fact that the one and only chain, which each and every node can agree upon, is a truth that is indisputable for all the nodes participating in the network. So, the protocol of every blockchain is, in fact, a consensus agreement protocol. The most interesting part of this truth is the hardness of the reversability of the blockchain. In order for an adversary to change one of the blocks, let's say the k-last block, he must do all the previous work (finding new solutions k-times for the OWF) and come up with brand new (k+1)-blocks in order to convince all the other nodes to accept his blockchain as the biggest and therefore the true one. If we assume honest majority in the network this holds with probability $1/2^{k+1}$ and obviously, this probability becomes negligible as the computing power of the adversary diminishes as well or the deeper she wants to meddle in the blockchain. Essentially, the mechanism explained above, makes it more expensive to fake a transaction than the potential gain. Without an appropriate algorithm for establishing consensus on the blockchain, there could be no trust in the blockchain-system of Bitcoin, since anyone with access to the history of transactions (all nodes), could re-write history and publish it as the true one.

So, the following definition seems to cover all of the above, that is the main aspects of a blockchain. A blockchain is a distributed computing architecture where a computer is called a node if it is participating in the blockchain network. Every node has full knowledge of all the transactions that have occurred, information is shared. Transactions are grouped into blocks that are successively added to the distributed database. Only one block at a time can be added, and for a new block to be added it has to contain a mathematical proof that verifies that it follows in sequence from the previous block. The blocks are connected to each other in a chronological order. The above definition is a very wide one, encompassing almost all existing implementations of blockchains.

Bitcoin - The first blockchain

Blockchain technology stems from the seminal white paper[21], outlining how the cryptocurrency Bitcoin could be constructed. Bitcoin solved a very important problem in the field of electronic money called double-spending. This problem is refering to one trying to buy two separate items while using the same monetary tokens. Up until the invention of the blockchain, as well as nowadays for everyone not using a cryptocurrency, this was solved through a central authority, such as a bank or another trusted third party. To decentralize this procedure Nakamoto proposed a time-stamp server, which ensures all transactions are appearing chronologically in the data structure called "the blockchain". So, the blockchain of the Bitcoin is a chain of linked blocks containing transactions from one user to another as shown in the figure below.



In Bitcoin, users do not have accounts or account balances but instead sign transactions using their private key. Each bitcoin is linked to a public key through an unspent transaction output (UTXO) and the user who possesses the corresponding private key is the owner and can control the usage of it. The UTXO is there because, in Bitcoin, all coins sent to an address have to be spent, even if the user actually doesn't want to spend the entire amount. It is however, possible to split a transaction. Assume that a certain address contains 3 XBT (abbreviation for the Bitcoin currency), and the owner of the private key to that address, i.e. the owner of the bitcoins, wants to pay 1 XBT to another address. The new 1 XBT transaction will use the entire 3 XBT as input and the 3 XBT are thereby spent. So, the change in this transaction, the 2 XBT, will be sent back to the same user but as a new input using a new address. A user who has taken part in payments, whether as payer or payee, will have a collection of addresses, all summing up to the total balance of her wallet.

The block holds transactions between users. In order for a transaction to be considered valid it has to be included in a block. For the creation of the blocks, the author(s) proposes the use of a Proof-of-Work (PoW) algorithm (finding the solution to a problem, as explained above) for establishing consensus on which chain is the correct one. This establishes an incentive for users to be correct in the validation of transactions. Especially in Bitcoin's blockchain, this is being done through solving the SHA-256 hash function for some output parameterized by a security parameter λ . This parameter is designed to change the hardness of the given value so as, in average, one block being generated every ten minutes. More precisely, a miner tries to find x in order to satisfy y=SHA-256(x), where y ends in λ zeros. For example, if there is a radical advancement in the computational power of even one miner, then the system will dynamically change the parameter accordingly as to maintain the balance intended by its creator(s).

Historically, the first blockchain to be used was Bitcoin, therefore, there was at first no distinction between Bitcoin and blockchain. All initial applications for blockchain were within cryptocurrencies or financial processes. Many blockchain-based use cases are still within the financial sector, but the benefits of disintermediation of trust have proven to be useful in other areas as well. This was brought forth by the advent of Ethereum, a Bitcoin-like cryptocurrency but with added functionality for smart contracts.

Post-Bitcoin blockchains

Some properties of Bitcoin have been abstracted and rebuilt into what is now called blockchain technology or distributed ledger technology. While still maintaining the main properties of Bitcoin, new blockchains are often more flexible in their applications and what actions they allow. It is a technology very much under development where new approaches and applications are being published frequently, most often through white papers published by start-ups or a group of corporate researchers[22]. Still the basics of blockchain remain the same, it is a distributed, time-stamped database with consensus-establishing peers.
Blockchain technology is characterised by the following traits:

- **Distributed:** Nodes are considered equal in the sense that they all have a full copy of the entire history of the database. There can also be less equal nodes, also called lightweight nodes, which only have a couple of the last blocks stored locally. Generally, communication between nodes is done over the Internet with private-key cryptography.

- **Time-stamped:** Since every block of transactions is hashed into all the subsequent blocks, it becomes increasingly difficult to change history the further away in time the current block is. The blockchain at hand becomes a provably correct auditing tool.
- **Consensus:** Nodes establish one truth about which version of the database is the correct one through a consensus-algorithm. This serves to validate transactions as well as to discourage for example double-spending attacks. The type of consensus-algorithm being used is highly dependent on the structure and purpose of the blockchain.

Permissions and specialization

As the development of blockchain technology progressed past Bitcoin, two different options developed as to who should be allowed to participate in the validation and observing of the network. The dichotomy is essentially between permissioned and permissionless blockchains, although there is in some cases some flexibility for hybrid solutions to be implemented. A blockchain which exists openly on the internet is called permissionless, classic examples of such are Bitcoin and Ethereum. This type of structure is what was defined in the Section 2.2. However, the more actions that are allowed, the more possibilities to hack the blockchain there are. This was seen during the infamous DAO-hack where approximately USD50 million were siphoned from an ether fund. [23]. Also, since the data on the blockchain is open to anyone who wishes to join the network, data has to be kept completely anonymised (as not completely successfully attempted by Bitcoin,Vasek and Moore, 2015) if it's necessary to keep it private. Since, in some cases, it is not possible to anonymise all the data or it is simply not desirable that everyone can participate in a network, permissioned blockchains were developed.

The principle of permissioned blockchains is that there is a regulation of who is allowed to join and participate in the network. This can be done by a consortium of companies, governmental agencies or other organisations, either by inviting new members one by one, or by predefining a set of criteria. The benefits, besides the increase in privacy, include the potential for more flexibility in adapting the network, better scalability and faster transactions. Sometimes, depending on the consensus algorithm at play, permissioned blockchains can be more susceptible to unintended changes of its history. In other words, the speed, privacy and scalability are sometimes being traded for immutability and censorship-resistance[3]. This is because a permissioned blockchain doesn't necessarily require a PoW-consensus algorithm, but can use one with less resource expenditure, thus making the process of concurrency easier.

Blockchains can also be created more or less flexible, or specific, in what actions are permitted on them. For example, Bitcoin and most coins, is an example of highly specialized chains with one purpose - to safely transmit the tokens of the cryptocurrency. On the other hand, there is Ethereum, with a virtual machine built in, as well as the possibility to deploy smart contracts in a turing-complete manner. Ethereum was explicitly created to allow for the creation of decentralised applications (DApps), and has at the time of writing this thesis, roughly 1700 applications listed on https://www.stateofthedapps.com/. Ethereum is, however,

Specialization\Permission	Permissionless	Permissioned
General Purpose	Ethereum	Monax's eris-db
Specialized	Bitcoin	Multichain

Table 4.1: Generalized vs. Specialized blockchains and Permissioned vs. Permissionless.[2]

permissionless and isn't the right platform for all DApps, necessarily. In table 2.1 the matrix of permissionless/ permissioned and generalized/specialized blockchains is shown with examples of a blockchain or platform for each category. A generalized blockchain is one which is not optimised for performing one specific task, in opposition to a specialized one that is. Both Ethereum and Bitcoin are permissionless, but on the permissioned spectrum, there is the multi-purpose eris-db from Monax and the specialized Multichain platform. Eris-db is a blockchain client containing a permissions layer, an implementation of the EVM and uses by default Tendermint consensus, although that can be modified. Tendermint is a Proof-of-Validation (PoV) algorithm, where scarce tokens are deposited and are threatened to be deleted if voting is dishonest. Eris-db is different from MultiChain in that, MultiChain is a fork from the Bitcoin Core source code and is in many ways optimised to work with the Bitcoin network. Multichain is specialized because it is optimised to perform transactions, whereas Monax is built to supply a great number of services. It does not, however, mean that it is not possible to build customised services on Multichain, or high-performance payment systems on Monax, just that it is made easier by design. It doesn't use PoW, but has a type of algorithm where the maximum amount of votes that a miner can cast during a specific timeframe is limited.[24]

Smart Contracts and Ethereum

The name smart contracts is arguably a misnomer since they are in fact neither smart nor contracts in the common sense. Smart contracts are, in the context of blockchain, simply logic that is published on a blockchain, can receive or perform transactions like any address (transactions may be rejected or require special arguments to function) and can act as an immutable agreement. The purpose of smart contracts is to act as a "computerised transaction protocol that executes terms of a contract"[25] and was first coined by cryptographer Nick Szabo. The basic idea, is that certain parts of contracts can be included in software in such a way that the breach of them is either expensive or impossible. Smart contracts are sometimes confused with Ricardian contracts, which is the digital recording and connection to other systems of a contract at law. This is not what is meant by smart contracts, since they do not need to be legal in any way, nor connected to outside systems. One could however, find value through the connection of smart contracts with Ricardian ones to "outsource" functionality of legal contracts to smart ones.

According to Szabo, contracts need to have some specific characteristics to be defined as, truly, smart. These characteristics are: visibility, online enforceability, verifiability and privity. Visibility (Szabo uses the term observability) means that participants in the contract should be able to see each other's performance of the terms of the contract, or to be able to prove the fulfilment of their own terms to other participants. It is also referring to the visibility of actions taken by the logic in the contract; a Point-Of-Sale screen showing the amount to be paid to the customer but omitting the fact that data is being saved from the credit card is an example of such a hidden action. Online enforceability refers to making certain that the terms of a contract are being fulfilled. The measures that can be taken in order to achieve this can be categorised into proactive and reactive ones. Proactive measures seek to make it technically impossible to breach terms or to allow either party to drop out of the contract if there is a valid breach on another part. Reactive measures deter malicious behaviour through reputation or enforcement, but also by recovering potential assets after breach of contract. Smart contracts also need to be verifiable, or auditable, should there be a conflict. Lastly, smart contracts should be as private as possible, meaning that knowledge and control of data involved in a smart contract should only be available to participants if necessary.

One might notice that the objectives of smart contracts just mentioned; visibility, online enforceability, verifiability and privity, results in two separate directions. Privity is exerting a controlling force over the contracts, wanting to minimise openness to outside parties. Diametrically opposed, there are the other three objectives, visibility, enforceability and verifiability, which require access to contractual data to be handed out to participants or auditors. Therefore, an optimum must be found where as little information and control as possible is given to external parties, yet the possibility to verify, observe and enforce is still available. In 1997, before blockchain technology and advances in zero-knowledge proofs, as well as, secure multi-party computations, Szabo's solution to the optimisation problem was to trust an intermediary, a third party, such as an auditor.[26][27] A problem of that approach is that by inducing judgement in the system, on one hand luck is induced and on the other it becomes automatically semi-decentralized. Nowadays, though, there exist the tools to eliminate such entities from our design maintainng all the desired properties of the smart contract in question.

The Ethereum platform is a general blockchain, with a virtual machine (Ethereum Virtual Machine, EVM) to run smart contracts. Since the environment exists only on the blockchain in the form of a virtual machine, the smart contracts are completely isolated from network, file-system or other processes on the node machines. A high-level, Turing complete language was created to write smart contracts with on Ethereum. However, that language, Solidity, has now become standard also for other platforms with smart contract capabilities. Solidity is similar to JavaScript in syntax, but is written in a completely different style. After a contract has been written in Solidity, it is compiled into EVM bytecode and then deployed at a specific Ethereum address. To deploy and interact with smart contracts on Ethereum however, a special JavaScript RPC-library is used alongside a web API.3. Because smart contracts programming started with Ethereum and Solidity, it is still a discipline under development. The Solidity language has a number of known peculiarities and a list of changes to come, meaning that code being written now may not be fully functional with the next update. There are a couple of programming best practices that are specific to smart contracts development, gathered in the (relatively) short time that Solidity has been in use. There are two main reasons behind the extra considerations of security that should be taken into account for smart contracts development; Solidity contracts are likely to process the ownership of valuable tokens, items or rights to something; the execution of smart contracts occurs on a blockchain, meaning that all participants can observe it and the source code for it. Common security guidelines that have been gathered during the short time that Solidity has been used are:

- **Damage Control:** If possible, the amount of tokens stored in a smart contract should be limited since, if the source code, the platform or the compiler would contain a bug, then the tokens may be stuck in the contract.
- **Modularity:** Smart contracts should be kept as tiny and simple as possible. Local variables and length of functions should be limited to keep the contracts as readable as possible. The more modular the contracts are, the easier it is to improve a system of smart contracts.
- **Check-Effects:** Functions should perform precondition checks at the first step of the algorithm. Then, as a second step, changes to state-variable should be made. Finally interactions with other contracts should occur.

Consensus algorithms

Consensus algorithms are of the highest relevance to blockchain technology since the purpose of Bitcoin was to transfer value in an unregulated, distrusting environment, where a sure way of validating transactions was needed. The goal of the consensus algorithm is to ensure a single history of transactions exists and that the history in question does not contain invalid or contradictory transactions. For example, that no account is attempting to spend more than the account contains, or to double-spend. In Table 2.2, different important consensus algorithms are compared to each other. Below, a brief introduction to a few of them is given, but for more details, the reader is referred to[21][28].

Bitcoin solved the consensus problem by announcing a target for each new block. This target has to be equal or more than the hash of the previous block, the hash of the current block and a variable nonce. Since the output of the hashing function is evenly distributed, it's impossible to create a block such that it will be easy to reach the target, with certainty. Therefore, there is a race between the mining computers in the network to find the right nonce. Once a target is reached, the mining computer broadcasts that block to the network and other participants validate the transactions. If enough validating nodes find the transactions to add up, they agree upon that block being added to the blockchain. This procedure is called proof-of-work (PoW). Since the goal is, not to give too much power to a single person or organisation, a limited resource has to be chosen which will be spent upon voting for the validity of a block. In PoW, that resource is computing power[28]. Since computing power is getting cheaper and more available with Moore's Law and cloud computing, the difficulty of the hashing problem is regulated according to the frequency with which the previous problems were solved. A common critique of PoW is however, that the "waste" of computing power also means a large waste of energy. There are miners who only mine in winter, and use the exhaust heat from the mining farm to warm up their house. What this essentially means is that miners are forced to pool resources into what can ultimately be a handful of giant Bitcoin farms, thus having centralised the decentralised network. Additionally, Bitcoin does not have a very high throughput of transactions since the block time stays constant at about 10 minutes and block size as well (about 1 MB). The energy waste and throughput are two reasons why alternatives have emerged. The most relevant are Proof-of-Stake (PoS) and Tendermint which are very similar. Neither uses computing power as a scarce resource, but rather the ownership of the inherent tokens of the blockchain. The principle is that owners of tokens put a certain amount of tokens at "stake" by betting on the version of the blockchain that they believe is the correct one. This will increasingly incentivise validators to behave according to the rules depending on how much they possess. Validators in the Tendermint consensus algorithm are nodes who take turns proposing blocks of transactions and then vote on them. If a block fails to get enough votes, the protocol moves to the next validator to propose a block. To successfully commit a block, there are two stages that need to be passed: pre-commit and pre-vote. A block is committed when more than 2/3 of validators pre-commit for the same block on the same round. As long as no more than 1/3 of validators are byzantine, it is impossible for conflicting blocks to be committed at the same height of the blockchain. Tendermint can be modified to act as a Proof-of-Stake algorithm by assigning different "weights" to the votes of different validators. In PoS, there is an attack, or a problem, called the nothing-at-stake-attack. The core of it is that there is no reason why a validator couldn't bet on all different proposed versions, thus being certain to win. The Ethereum wiki-page explains it as: an attacker may be able to send a transaction in exchange for some digital good (usually another cryptocurrency), receive the good, then start a fork of the blockchain from one block behind the transaction and send the money to themselves instead, and even with 1%of the total stake the attacker's fork would win because everyone else is mining on both.

Consensus Algorithm	Resource being used	Benefits	Drawback	Examples
PoW	Computing power	Trustless, immutable, highly decentralized	Energy consumption, transaction output	Bitcoin
PoS	Ownership of fixed ammount of tokens	efficient in energy and throughput, scalable	Nothing-at-stake problem	Cardano
Delegated PoS	Ownership of scarce tokens and peer reputation	Allegedly more efficient than PoS	Voter apathy in elections can lead to excessive centralization and reduced robustness	BitShares
Proof-of-Validation (PoV)	Security deposit of scarce tokens subject to burn if voting dishonestly	Has the benefits of PoS without almost any of its drawbacks	Nothing-at-stake problem still persists	Eris-Db

Table 4.2: Consensus algorithms for usage in blockchains. Adapted from [3]

Pegged Sidechains

Sidechains are the concept of parallel blockchains that allow assets from one blockchain to be transferred into another. The concept was formally announced in 2014 by Blockstream, which is a private company consisting of many of the Bitcoin core development team members. While the concept originated from Bitcoin, the broader theory behind sidechains is applicable to any blockchain design. At first, the intention of sidechains was to enable the transfer of bitcoins to other blockchains, thus enabling sidechains to act as alternative cryptocurrency systems without requiring the minting of new coins. However, the initial permissionless sidechain design was discovered to contain non-trivial security flaws. Meanwhile, newer designs are currently under development, but have at the time of writing not reached maturity or production.[8]

Nevertheless, transferring assets from one blockchain to a permissioned blockchain is in fact already workable by the sidechain's design. The idea is fairly simple, with the introduction of a function called a two-way peg, assets will be allowed to be transferred from one blockchain to another and back. One of the key principles of the two-way peg is that it is impossible to return more assets to the "parent chain" than what originated from it, thus, the total number of assets in the parent chain cannot be compromised by the implementation of the peg. Thus, any new rules can be implemented in the sidechain without posing a risk to the parent chain.

Sidechains can extend the functionality of a parent blockchain by introducing new features on the sidechain. For example, since a permissioned sidechain can leverage a threshold signature scheme consensus model, this allows for near-instant confirmation times of an originally permissionless token such as the bitcoin. One example of such a sidechain is the sidechain Liquid, maintained by Blockstream for various Bitcoin exchanges.[9] Other examples of features sidechains can potentially bring to permissionless networks are things such as supporting multiple asset types from different blockchains to exist on a mutual sidechain such as smart contracts.

A two-way peg can be either symmetric or assymetric. In the first case, assets are locked on the parent chain through the means of multi-party escrow, meaning that they are sent to an address which requires a multisignature to unlock. This multisignature is formed by the permissioned entities on the sidechain. When the assets on the parent chain are in escrow, the sidechain can allow the creation of these assets on its own blockchain. In order to reintroduce the assets from the sidechain on the parent chain, the owners of the assets on the sidechain must prove that they have destroyed the coins on the sidechain by sending them to an unspendable address.[29] When this proof is provided, the permissioned entities on the parent chain release the same amount of assets from the parent chain escrow. The proof provided for this whole scheme is a simplified payment verification proof (or SPV proof).[8] Essentially, an SPV proof is composed of a list of blockheaders demonstrating proof-of-work and a cryptographic proof that an output was created in one of the blocks. This allows verifiers to check that some amount of work has been committed to the existence of an output. Such a proof may be invalidated by another proof demonstrating the existence of a chain with more work which does not include the block which created the output. So the symmetric peg works as follows: to transfer parent chain coins into sidechain coins, they are sent to a special output on the parent chain that can only be unlocked by an SPV proof of possession on the sidechain. The two blockchains need to be synchronized and in order to achieve that, the definition of two waiting periods is needed.

- **Confirmation period:** The duration for which a coin must be locked on the parent chain before it can be transferred to the sidechain. A typical confirmation period would be on the order of a day or two in the originally proposed idea.

- **Contest period:** A duration in which a newly-transferred coin may not be spent on the sidechain. A typical contest period would also be on the order of a day or two in the originally proposed idea.

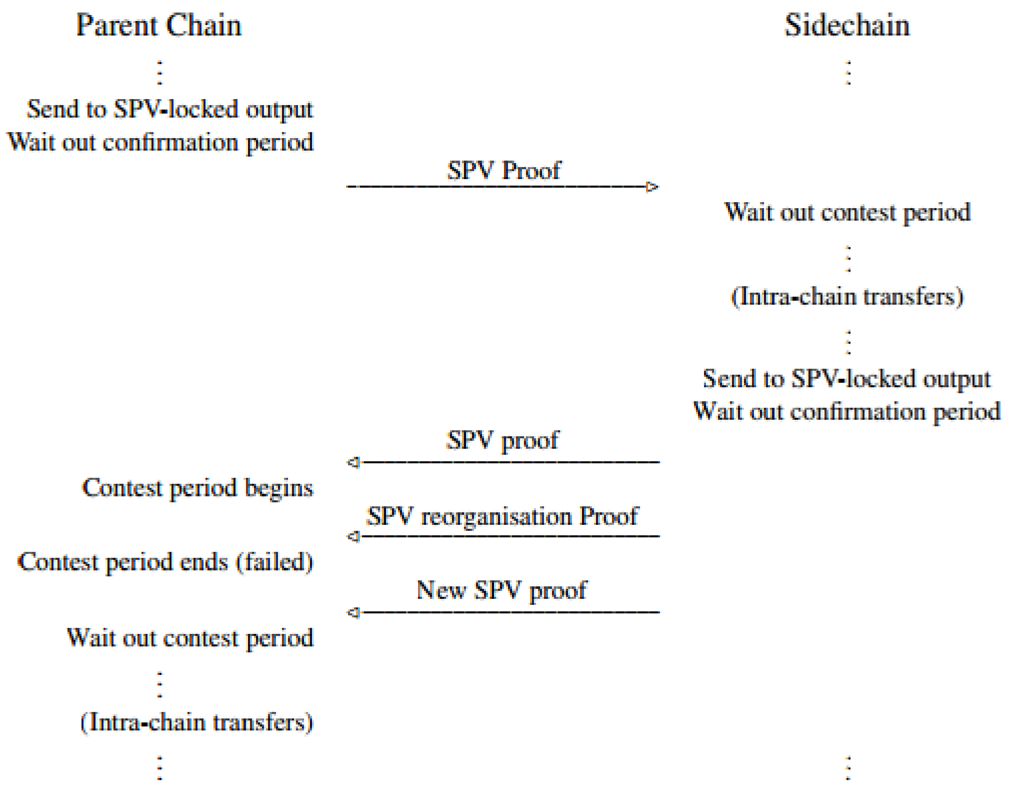


Figure 4.1: Example two-way peg protocol.

The assymetric peg, on the other hand, works somewhat differently. The users of the sidechain are full validators of the parent chain, and transfers from parent chain to sidechain do not require SPV proofs, since all validators are aware of the state of the parent chain. Still, though, the parent chain is unaware of the sidechain, so SPV proofs are required to transfer back. As a result, prerequisite for the adoption of this scheme consists that the sidechain's validators are forced to track the parent chain, whereas in the symmetric scenario no such thing is needed. In conclusion, it is obvious that the symmetric peg has a dreadful drawback concerning its waiting periods, whilst the assymetric one requires more information to be stored in the clients of the sidechain's users.

SPV proofs

In order to transfer coins from a sidechain back to Bitcoin, we need to embed proofs that sidechain coins were locked in the Bitcoin blockchain. These proofs should contain (a) a record that an output was created in the sidechain, and (b) a DMMS proving sufficient work on top of this output. Because Bitcoin's blockchain is shared and validated by all of its participants, these proofs must not impose much burden on the network. Outputs can be easily recorded compactly, but it is not obvious that the DMMS can be.

Compact SPV Security. The confidence in an SPV proof can be justified by modelling an attacker and the honest network as random processes. These random processes have a useful statistical property: while each hash must be less than its target value to be valid, half the time it will be less than half the target; a third of the time it will be less than a third the target; a quarter of the time less than a quarter the target; and so on. While the hash value itself does not change the amount of work a block is counted as, the presence of lower-than-necessary hashes is infact statistical evidence of more work done in the chain. We can exploit this fact to prove equal amounts of work with only a few block headers. It should therefore be possible to greatly compress a list of headers while still proving the same amount of work. We refer to such a compressed list as a compact SPV proof or compressed DMMS.

However, while the expected work required to produce a fraudulent compact SPV proof is the same as that for a non-compact one, a forger's probability of success no longer decays exponentially with the amount of work proven: a weak opportunistic attacker has a much higher probability of succeeding "by chance"; i.e., by finding low hashes early. To illustrate this, suppose such an attacker has 10%of the network's hashrate, and is trying to create an SPV proof of 1000 blocks before the network has produced this many. Following the formula in we see that his likelihood of success is $\approx 10^{-196}$. To contrast, the same attacker in the same time can produce a single block proving 1000 blocks' worth of work with probability roughly 10 , a much higher number.

For now we will describe an implementation of compact SPV proofs, along with some potential solutions to block this sort of attack while still obtaining significant proof compaction. Note that we are assuming a constant difficulty. We observe that Bitcoin's difficulty, while non- constant, changes slowly enough to be resistant to known attacks. We therefore expect that corrections which take into account the adjusting difficulty can be made.

Implementation. The inspiration for compact SPV proofs is the skiplist, a probabilistic data structure which provides log-complexity search without requiring rebalancing (which is good because an append-only structure such as a blockchain cannot be rebalanced). We require a change to Bitcoin so that rather than each blockheader committing only to the header before it, it commits to every one of its ancestors. These commitments can be stored in a Merkle tree for space efficiency: by including only a root hash in each block, we obtain a commitment to every element in the tree. Second, when extracting SPV proofs, provers are allowed to use these commitments to jump back to a block more than one link back in the chain, provided the work actually proven by the header exceeds the total target work proven by only following direct predecessor links. The result is a short DMMS which proves just as much work as the original blockchain.

How much smaller is this? Suppose we are trying to produce an SPV proof of an entire blockchain of height N. Assume for simplicity that difficulty is constant for the chain; i.e., every block target is the same. Consider the probability of finding a large enough proof to skip all the way back to the genesis within x blocks; that is, between block N –x and block N. This is one minus the probability we don't equals x over N and the expected number of blocks needed to scan back before skipping the remainder of the chain is thus N+1 over 2.

Therefore if we want to skip the entire remaining chain in one jump, we expect to search only halfway; by the same argument we expect to skip this half after only a quarter, this quarter after only an eighth, and so on. The result is that the expected total proof length is logarithmic in the original length of the chain. For a million-block chain, the expected proof size for the entire chain is only $\log 1000000 \approx 20$ headers. This brings the DMMS size down into the tens-of-kilobytes range.

However, as observed above, if an attacker is able to produce compact proofs in which only the revealed headers are actually mined, he is able to do so with non-negligible probability in the total work being proven. One such strategy is for the attacker to produce invalid blocks in which every backlink points to the most recent block. Then when extracting a compact proof, the attacker simply follows the highest-weighted link every time. We can adapt our scheme to prevent this in one of several ways[8]:

- By limiting the maximum skip size, we return to Bitcoin's property that the likelihood of a probabilistic attack decays exponentially with the amount of work being proven. The expected proof size is smaller than a full list of headers by a constant (proportional to the maximum skip size) factor.
- By using a maximum skip size which increases with the amount of work being proven it is possible to get sublinear proof sizes, at the cost of subexponential decay in the probability of attack success. This gives greater space savings while still forcing a probabilistic attacker's likelihood of success low enough to be considered negligible.
- Interactive approaches or a cut-and-choose mechanism may allow compact proofs with only a small security reduction. For example, provers might be required to reveal random committed blockheaders (and their connection to the chain), using some part of the proof as a random seed. This reduces the probability of attack while only increasing proof size by a constant factor.

If we expect many transfers per sidechain, we can maintain a special output in the parent chain which tracks the sidechain's tip. This output is moved by separate SPV proofs (which may be compacted in one of the above ways), with the result that the parent chain is aware of a recent sidechain's tip at all times. Then transfer proofs would be required to always end at this tip, which can be verified with only a single output lookup. This guarantees verifiers that there are no "missing links" in the transfer proofs, so they may be logarithmic in size without increased risk of forgery. This makes the total cost to the parent chain proportional to the number of sidechains and their length; without this output, the total cost is also proportional to the number of inter-chain transfers.

Chapter 5

Cryptographic Protocols

Introduction

Cryptographers aim to deliver protocols which preserve security requirements in the presence of an arbitrary adversary. However, such an adversary should not be expected to follow any prescribed rules, nor adhere to any particular behavioral patterns. Moreover, the ingenuity of the adversary should be considered boundless. As an analogy, let's consider, for example, the challenge faced by the architects of the 'inescapable' Alcatraz. The prison is situated 1.5 miles from San Francisco, California, and isolated from the mainland by the strong currents of San Francisco Bay. In addition to the treacherous waters and the physical security features, inmates were counted twelve times a day and the ratio of inmates to armed guards was three to one. Despite this, an elaborate escape plan was hatched by Clarence Anglin, John Anglin, Frank Morris and Allen West which involved the fabrication of: life-like dummies, fashioned from soap, toilet paper and hair (the dummies were used to avoid detection during evening head counts); tunneling equipment, built from the motor of a stolen vacuum cleaner; and a raft, constructed from the rain coats of fellow inmates. On June 11, 1962 the two Anglin brothers and Morris escaped the prison. Their whereabouts are currently unknown. Given the water temperature and tidal direction, the official FBI investigation presumes the three men drowned; however, the only thing known for certain, is the security of Alcatraz was violated. Designing a resilient prison to contain such devious inmates, whom are determined to escape, is difficult. In some sense, cryptographers face a more challenging problem: a prison may be considered inescapable if no inmate has successfully broken out; by comparison, we will only consider a protocol to be secure, if security requirements cannot be violated by any adversary.

Cryptographic protocols are small distributed algorithms that aim to provide some security-related objective over a public communication network, such as the Internet. Since the communication medium is public, an adversary may interfere with the transmitted messages. This introduces the possibility of interference by a powerful adversary and has made the design of secure protocols notoriously difficult. The canonical example is the Needham-Shroeder public key protocol[30], which is intended to provide mutual authentication of two principals. The protocol was scrutinized by experts for nearly two decades before Lowe discovered a man-in-the-middle attack[31]. This dictates the necessity for protocol verification and moreover, highlights the need for automated support to overcome the inherent human weaknesses present in the manual verification process.

Verification of protocols

When verifying protocols there are two distinct cases: the ideal and the real world. Firstly, in the ideal scenario there exists two schools as to how to verify a protocol: the computational and the formal methods one.

- **Computational:** In this school, often called the provable security school, a protocol is analyzed according to some well-defined model. A complexity theoretic reduction is given which turns an adversary against the protocol into an adversary against a component. Thus the protocol is secure if all of its components are secure. To determine whether the component is secure (which is often a cryptographic scheme) a further reduction is provided to a cryptographic primitive. These reductions can be complex, with proofs being many pages long, and needing to be produced and verified by hand.
- **Formal Methods:** In this school, often called the symbolic school, a protocol is analyzed assuming perfect cryptography. The protocol is then abstractly described in, for example, a process algebra, with the attacker's goals being described by equations and the attacker's capabilities defined by equational theories. After this stage an automated checking process is applied to ensure that the attacker can never reach the stated goals. This type of analysis is often said to use the "Dolev-Yao" model.

Each school has its advantages and disadvantages. The first school has the disadvantage that it requires hand generated and checked proofs; on the other hand it models the cryptography and adversary as close to the real world as possible. The second school has the disadvantage that it assumes perfect cryptography (which we know is not possible in real-life systems), but it allows for tools that automate devising proofs, or at the very least checking them.

Now, in the real world protocols are almost always designed, deployed, patched and extended before any formal analysis of their properties happens. This is either for historic reasons (the protocols date to a time before verification techniques were understood), or for business reasons (for example a protocol is defined "in house" and only then is made public for public analysis, after a product has been produced). Thus formal verification using either symbolic or computational means is often applied post-hoc. Often this post-hoc analysis finds faults in the standarized/deployed protocols or it can only be applied to small subsets of the actual protocol. In the latter case one can obtain verification of the small part, but not of the whole protocol. This leads to a major problem concerning protocols: their composability. A protocol when run in isolation may be secure but should it be run in composition (either sequentially or in parallel) it may not retain its security properties. Since most protocols are run in an online environment, with multiple executions happening at any one time, the issue of parallel composition is vital to ensure. Techniques exist to enable composition of protocols to be designed in from the start, for example the Universal Composability Framework, which consists the strongest composability framework today.

Ideal/ Real paradigm

One of the main objectives of cryptography, as stated before, is to construct protocols, which are "secure" even in the presence of corrupted parties. But, first of all, we have to define what secure means. In order to do so imagine what properties we would have in an ideal world and then we call a protocol secure if the real (the constructed) protocol has similar properties. This is the basic idea of the Ideal/Real paradigm.

There are two main kinds of adversaries: static and adaptive. In the first case, the adversary chooses which party it corrupts before the protocol begins[32]. In the latter case, the adversary chooses the party to be corrupted during the execution of the protocol. The network, which is used might be either authenticated, which means that the receiver always knows who the sender was, or not. It might also have secure or public channels. The former ensure that the transmitted messages reveal useful information only to the receiver, while the latter do not. Here, for simplicity, we may assume static adversaries and a network with secure, authenticated channels.

For example let us see a ZK protocol for some relation R, where generally the verifier V has as input some y and the prover P wants to prove to V that there exists some x such that (x, y) ϵ R. In an ideal world we can imagine a third party, which is honest and trustful and can communicate with both P and V . In this ideal scenario, P could give (x, y) to this trusted party the latter would check if (x, y) ϵ R and then tell V if this is true or false.

However, in the real world we do not have such trusted parties and we have to substitute them with a cryptographic protocol π between P and V . Roughly speaking, the Ideal/Real paradigm requires that for whatever information an adversary A (which plays the role of either P or V) could retrieve in the Real world, there is a way to retrieve it in the Ideal world as well.

The trusted third party can be viewed as the functionality we want to achieve and we denote it by F_{ZK} . If some protocol satisfies the above property regarding this functionality, we call it secure. The formal definition of security follows:

Definition 1. A protocol π realizes F_{ZK} if for all ppt A, there exists a ppt S such that

$$Real_A^{\pi} \approx_c Ideal_S^{F_{ZK}} .$$

Now let us see what is the role of simulator S in each case of corruption. In the case where the adversary A corrupts the verifier V , the simulator S only learns in the ideal world whether the statement is true or not, while in the real world A also sees a proof for that. Thus, S must be able to simulate an accepting proof, while only knowing that the statement is true. On the other hand, if A corrupts P , S must be able to provide the witness x to the functionality F ZK in the Ideal world. Observing that S can simulate V we see that S must be able to extract the witness from P (which is corrupted). The next theorem must be intuitively clear:

Random oracle

Another primitive is the **Random oracle model** which is commonly used in cryptographic protocols and we shall now give its definition:

A function H : $\{0,1\}^* \rightarrow \{0,1\}^k$, mapping bit strings of arbitrary length to bit strings of a fixed length k, k > 0, is called a hash function. Function H is called a cryptographic hash function, if it is easy to compute H(x) given any string x, and one or more of the following requirements are satisfied:

- **preimage resistance (onewayness)**: given a k-bit string y, it is hard to find a bit string x such that H(x) = y.
- **2nd-preimage resistance (weak collision resistance)**: given a bit string x, it is hard to find a bit string x' ≠ x such that H(x') = H(x).
- **collision resistance (strong collision resistance)**: it is hard to find a pair of bit strings (x, x') with x ≠ x' such that H(x) = H(x').

In general, collision resistance implies 2nd-preimage resistance, but collision resistance need not imply preimage resistance. In practice, however, cryptographic hash functions usually satisfy all three requirements[11].

Practical examples of cryptographic hash functions are MD5, SHA-1, SHA-256, with output lengths k = 128, k = 160, and k = 256, respectively. If collision resistance is not required, one may truncate the outputs by discarding, e.g., the last k/2 bits; the resulting hash function is still preimage resistant.

Many protocols make use of a cryptographic hash function. In order to be able to prove anything useful on the security of such protocols one commonly uses the so-called random oracle model. In this model, a cryptographic hash function is viewed as a random oracle, which when queried will behave as a black box containing a random function H : $\{0,1\}^* \rightarrow \{0,1\}^k$, say. If the oracle is queried on an input value x, it will return the output value H(x). As a consequence, if the oracle is queried multiple times on the same input, it will return the same output value, as H is a function. Moreover, if one observes the distribution of the output value for different input values, the distribution will be uniform, as H is a random function.

Note that the use of the random oracle model is a heuristic. If we prove a protocol secure in the random oracle model, it does not follow that the same protocol using, e.g., SHA-256 as its hash function is secure, since SHA-256 is simply not a random function. Thus, the practical upshot of the random oracle model is that a protocol proved secure in it can only be broken if the attacker takes into account specific properties of the concrete hash function used.

There are numerous further requirements that can be demanded of a cryptographic hash function beyond what is stated in the definition stated earlier. In general, any deviation from what can be statistically expected of a random function should be absent or unlikely, and in any case it should be infeasible to exploit such statistical weaknesses. By definition, the random oracle model is robust with respect to such additional requirements. For example, partial preimage resistance (or, local onewayness) basically states that given a k-bit string y it is hard to find (partial) information about any input x satisfying H(x) = y. Many applications of hash functions,such as the bit commitment scheme stated later, rely on this requirement rather than the (much weaker) requirement of preimage resistance[11]. Clearly, partial preimage resistance also holds in the random oracle model, in which each hash value is, by definition, statistically independent of the input value.

On-chain and off-chain protocols

Since the introduction of Bitcoin and blockchain in general, the possibility of running cryptographic protocols on a blockchain has been explored. The desirable properties provided by the use of the blockchain, such as, anonymity, verifiability and traceability of assets are are some of the reasons for doing so. The issue that was, firstly, encountered was the time of execution of such constructions, especially in the Bitcoin's blockchain. Imagine trying to run a Σ-protocol where three messages need to be exchanged. For doing so, while being close to certain that there will be proof of the protocol's execution, at the time of writing this thesis and supposing the use of Bitcoin's blockchain, the two parties would have to wait approximately three hours, for the messages to be burried deeply enough in the blockchain and thus be forever verifiable.

For even more complex designs, the time needed to be spent, before the completion of the protocol, consists in reality a prohibitive factor in designing on-chain protocols. So, the idea of constructing off-chain protocols is being entertained and in fact the first results seem to be very efficient both with respect to time and to computational complexity of the designs. In particular, an off-chain protocol for executing the game of poker was introduced recently and was part of the motivation for this thesis as well. The idea behind that construction was to use the blockchain "only when needed" through a penalizing mechanism for players' misbehavior, while, otherwise, executing the protocol over a public network.

Chapter 6

Our construction

Introduction

In this chapter the game of roulette will be executed. It is a game played in rounds, where each round has a random outcome that affects the balances of the players according to their respective bets. Specifically, a protocol will be constructed for this purpose which must have certain limitations and properties. In order to identify them the game has to be broken down. Firstly, let's think of this as a house game, where there is the owner of the game, RouletteMaster (RM) and the rest of his guests are playing against him (players). In this scenario all people have direct contact with the physical aspect of the game, a game which has some rules by default and the ones that are of major importance to our construction, specifically, are the following:

- The maximum payout in each round for every player, should one decide to go all-in on a number, with balance b is $36 * b$.
- At any given point, before the spinning of the ball, the game owner has to be able to cover the maximum payout for all of her guests.

Now, let's assume that we want to execute this game remotely, where all communication is being done over a private network. A first idea would be for all players to send their input concerning their choices, in numbers and bets, then wait for the RM to spin the ball, tally the result and inform accordingly all the players. In this case an extreme amount of trust towards the RM is required as:

- There are no guarantees of any kind that RM will not lie, concerning the input received from the players
- There can be no verification of the randomness of the result.

In fact, RM is practically incentivised to lie about the choices of the players or even generate biased results. In order to avoid this extreme trust prerequisite the idea of using a blockchain to store the communication between each player and RM was explored alongside a multi-party computation in order to generate a random result for all players. In order to go deeper into the construction let's state some preliminaries.

Preliminaries

Game overview

Initially a player places bets on numbers from 0 to 36 according to her wishes. Upon completing the bet placement a ball is spinned on a 37-spot wheel and when it stops on a number the bets are being payed from the host of the game towards the players who bet on that particular number. All other bets are lost to the game host. The spinning of the ball is used to simulate a number generation as randomnly as possible. The game is repeated until, either the player or the host loses all their money, or upon one of them wishes to terminate the game.

Actors

The actors of the protocol, as stated above, will be two: the RouletteMaster (RM) and the other players. RM uses a blockchain, namely Roulettechain and is a miner in it. Other players get Roulettechain's tokens, in order to play against RM. Finally, the latter is in charge of publishing on the blockchain crucial information concerning the communication between every other player and herself.

Digital signatures

The communication between the players is being done using secure digital signatures with Existential Unforgeability under Adaptive Chosen Message Attacks (EUF-CMA). In general, a digital signature scheme is a tuple of three PPT algorithms SIG = (SIG.Gen, SIG.Sign, SIG.Vrf) such that:

- SIG.Gen(1^λ) takes in a security parameter and outputs a verification key SIG.vk and a signing key SIG.sk.
- SIG.Sign SIG.sk (m) takes in a signing key SIG.sk and a message m, outputting a signature σ on message m under signing key SIG.sk.
- SIG.Vrf SIG.vk (m, σ) takes in a verification key SIG.vk, a message m and a signature σ, outputting 1 if the signature is valid and 0 otherwise.

Algorithm Calcplayers

The calculation of the players for each round is being done with the use of the following algorithm which ensures at the start of every given round maximum resources in game.

Idea from naive to smart

The first (naive) idea was to have RM use Roulettechain, on which the communication of the protocol would be stored. Other players wanting to play would get that blockchain's tokens. Upon wishing, RM would broadcast her intension to host a game and all interested parties would join. Should a player run out of

```
Data: queue[n], balance[n]
Result: The players for a round of roulette
initialization;
tempbalance ← balance[0];
i ← 0;
while $tempbalance > 0$ and $i \leq n$ do
    i++;
    if $tempbalance > balance[i]*36$ then
        tempbalance ← tempbalance - balance[i];
        give player with balance[i] $id_i$;
    else
        swift from i+1 space balance[n] one spot right;
        balance[i+1] ← balance[i] - tempbalance;
        balance[i] ← tempbalance/36 ;
        tempbalance ← 0 ;
        give player with balance[i] $id_i$;
    end
end
```

Figure 6.1: Algorithm *CalcPlayers*

money, they would be eliminated from the game. The above scenario has some major drawbacks needing attention in order to create a cryptographic protocol with desirable properties. In particular, RM can exclude messages from some blocks, thus pretending never to have gotten them in the first place. Another power she has over the game is reorganizing the table as seeing fit, should RM lose a round and as a result her balance would be below the threshold of being able to cover the balances of the rest of the players. An issue needing attention as well, is that the instance of the game described till now is a static one, where players simulate a tournament-type roulette game. Lastly, should all messages be written on the blockchain then, after the completion of the game, all information stored in previous blocks would be of no use to anyone, so there would be too much useless space on-chain.

Our first attempts on tackling these weaknesses were to focus on achieving dynamic-ability with respect to players in the game. This can be succeeded by constructing a one-round protocol that when executed repeatedly carries out a dynamic game of roulette. This one-round protocol would be played seperately between the host and each other player in a ”one.vs.one” setting and would consist of four phases and at the end of each crucial information would be included in a block, on Roulettechain and every player would verify its legitimacy. In particular, the four phases in time sequence would be the following:

- **Request phase:** All players send their intentions, for participating or not, to RM for the round.
- **Calculating players phase:** RM calculates the players for the round and publishes them on-chain.

- **Betting Phase:** All players send their bets to RM, who then publishes them on-chain.
- **Tallying result phase:** RM calculates the results for the round and pulishes them on-chain.

Having ensured the dynamic aspect of our game, playerwise, the next problem was space. The notion of sidechains erupted and the solution was found. In more detail, there would be, now, Roulettechain and should one wish to host a game they would start a pegged sidechain from one of Roulettechain's blocks. The advantage provided was that upon completion of the game, all of its participants could erase their locally stored sidechain with the new balances of the players being secured on Roulettechain, having avoided the excessive space on it and therefore, on the hard drives of the nodes of the network.

The issue with RM being able to reorder the table was conquered with the adoption of an algorithm that would ensure maximum resources in the game at the start of any given round. More specifically, for the algorithm to work a queue of players wanting to take part in the game as well as a queue for the requests of every player would need to be inserted into the construction. The algorithm works s follows: RM takes requests for the round according to, firstly, the players of the previous round and then her queue of players waiting to play. All these requests are added serially to her queue of requests. Then she takes the requests of the new, prospective, players and adds them as well, updating the queue of requests once more. Now, RM executes the requests respecting priority and if the game "fills up", meaning that the insertion of another player in the game could jeoprdize the second rule of the game, as stated before, then the player in question will be inserted with as much money as possible so as to maintain coverability of the RM bet-wise. The proof that this algorithm is the best for RM and all other players derives from the fact that due to the nature of the game, RM has a 2,7%advantage.

Another improvement could be considered in the area of the multi-party computation. More precisely, a commitment scheme where the commitment would be the encrypted bets of a player and the reveal would be the secret encryption key was explored and seemed to achieve even greater security.

In addition to all of the above, to prevent RM from excluding message, a recovery mechanism will be inserted to our construction. The purpose of this mechanism is to ensure that should a wrongdoing happens by the RM, the players can reverse it and incur the apropriate penalty. If a player enters the recovery at any step of a given phase, she complains about a mistake on Roulettechain, providing the encriminating evidence, while haulting the flow of the game. So, should Roulettechain have liveliness and robustness, this complaint will be inserted in a block eventually and the incident will be corrected.

Finally the realization that there is no particular need of a universal result per round and that it is rather an improvement if we consider the game of roulette as many instances of an one.vs.one games. This holds because in the latter scenario, the adversary of our model will be able to corrupt at most one player per result.

The Roulette Functionality $F_{Roullete}$

. We, now, formalize the earlier game in the ideal functionality $F_{Roulette}$ in the figure below.

Functionality $F_{Roulette}$

The functionality is executed with n players with identities $(id_0, \dots, id_n\ )$. There is one corrupted party that is controlled by S. Whenever a message is sent to S for confirmation or action selection, S should answer, but can always answer abort, in which case the recovery procedure is executed; this option will not be explicitly mentioned in the functionality description henceforth.

- **Request phase:** Wait to receive request[i] from each player and procced as following depending on their choice:
  - **Player Check-out**: Send (*Check-out, i*) to S. If S answers (checkout, i), mark P_i as inactive in the game, send (payout, coins(balance[i])) to P_i and ignore future messages from P_i . If no active player is left, stop the execution.
  - **Player Check-in**: Wait to recieve a message (*Check-in, M*) from each other player and store them in $requests_0[n]$.
- **Calculating players phase:** Send $requests_0[n]$ to S so as to run the algorithm CalcPlayers to decide on the eligible players of the round and announce the final check-ins to the other players. Keep track of the active players in the game, who are automatically considered that upon checking in the game.
- **Betting phase:** Wait to receive $bet_i[j]$ $=Enc_r(m_{ij}$ from each player and inform all the other players about them.
- **Tallying result phase:** Send $bet_i[j]$ for all i,j to S. S calculates $result_k$[i] = $(SHA_{256}(\sum_{j=1}^{m}(\text{Dec}(Enc_r(m_{i,j})) \oplus (\text{Dec}(Enc_r(m_{0,j}))))$ mod 37 and informs the other players about the results.

Figure 6.2: Functionality $F_{Roulette}$

Roulette protocol

In building our protocol, we consider all ideas mentioned above and formally express them. All actors are considered players in the manner of RM being $Player_0$,whereas everyone else engaging in the game being $Player_i$. Each player's balance is represented in a vector *balance[i]*. An array containing the requests of each of the n players for a round is needed, requests[n] as well as one for each player's bets, $bet_i[j]$. Also, the hash function SHA256(x) along with the $\oplus$ permutation is used for the calculation of the result for round k and per $player_i$, $result_k$[i].

Protocol $\pi_{Roulette}$

Protocol $\pi_{Roulette}$ is executed by n players with identities (id_0 ,. . . , id_n), parameterized by a timeout limit τ and interacting with the stateful contract functionality F_{SC}. We assume that the parties agree on a generator g of a group G of order p for the El-Gamal encrypion scheme EGE and also on a EUF-CMA secure digital signature scheme SIG. Moreover, a nonce unique to each protocol execution and protocol round(e.g. a hash of the public protocol transcript up to the current round) is implicitly attached to every signed message to avoid replay attacks.
Recovery Triggers: Whenever a signature is published, its validity is tested. If the test fails, the party proceeds to the recovery phase. The same happens if a party does not receive an expected message until a timeout limit τ.

Request phase: For i = 0, . . . , n, the party with id_i proceeds as follows:

- Verifies information in the previous block, secretkey[i], that is the other player's secret key as well as $result_k$[i] for the previous (k) round.
- Generates the keys of the signature scheme
  ($SIG.vk_i$, $SIG.sk_i$) ← SIG.Gen (1^λ).
- Uses the keys above forn the encryption scheme, such as,
  $EGE.sk_i = SIG.sk_i$ and $EGE.pk_i = SIG.vk_i$
- Sends:
  - **Check-in:** (*Check-in,M, walletadress*$_i$ $SIG.vk_i$), where M is the maximum sum of money intended to be played for the round and $walletaddress_i$ points to the transaction of a block in Roulettechain from where the money would be bet, if she wants to play.

Figure 6.3: Protocol $\pi_{Roulette}$

- Or else:
    - **Check-out:** (*Check-out, balance, σ*) to F_{SC}, where σ contains all signatures on balance, waits for confirmation from F_{SC} and stops execution, otherwise.
- RM, that is, $Player_0$ with id_0 publishes request[i] on the sidechain.

Calculating plyers phase:

- Each player verifies information in the previous block, that is the other player's request.
- $Player_0$ runs Algorithm *CalcPlayers* to determine the players of the round P_i and publishes them on the sidechain alongside her queue[0].

Betting phase:

- Each player verifies information in the previous block, that is players for the round and queue[0].
- Each $Player_i$ updates their queue[i].
- Each player chooses a randomness r and sends her $bets_i$[j]= $Enc_r(m_{ij})$ where m includes information about the number intended to be bet on alongside the respective sum of money.
- $Player_0$ publishes $bets_i$[j] on the sidechain.

Tallying result phase:

- Each player verifies information in the previous block, that is the $bets_i$[j].
- Each player sends his EGE.sk_i to $Player_0$.
- $Player_0$ decrypts $bets_i$[j] and calculates $result_k$[i] $= (SHA_{256}(\sum_{j=1}^{m}(\text{Dec}(Enc_r(m_{i,j})) \oplus (\text{Dec}(Enc_r(m_{0,j})))) \bmod 37$.
- $Player_0$ publishes $result_k$[i] and secretkey[i] = EGE.sk_i on the sidechain .

Recovery request: If a party P_i enters the recovery phase at any step of a given phase, it halts the execution of the protocol and sends a message (report, $Player_i$, sk_i, $Block_z$) to F_{SC} , where $Block_i$ is the block that contains information on which the verification test failed and thus contains forged or false data.

Figure 6.4: Protocol $\pi_{Roulette}$ (continuation)

Stateful contract functionality F_{SC}

Implementation of F_{SC}. It is important to emphasize that the F_{SC} functionality can be easily implemented via smart contracts over a blockchain, more formally, using a public available ledger. In addition, our construction (for protocol $\pi_{roulette}$) requires only simple operations, i.e., verification of signatures and discrete logarithm operations over cyclic groups. The regular operation of our protocol is performed entirely off-Roulettechain, without intervention of the contract. However, in the case that any participant in the game claim problems in the execution, any player can publish their complaint in the Roulettechain. This approach reduces the information stored in the parent-blockchain.

Assuming that the DDH problem is hard and that the digital signature scheme SIG is EUF-CMA secure, protocol $\pi_{Roulette}$ securely computes $F_{Roulette}$ in the F_{SC} -hybrid, random oracle model in the presence of malicious static adversaries.

In order to prove the above statement we construct a non-uniform expected probabilistic polynomial time simulator (ideal adversary) S and internal copies of c corrupted parties. S simulates both the actions of honest and corrupted parties and the functionality F_{SC} . Let H denote the set of honest parties and C denote the set of corrupted parties in the internal execution run by S. S is parameterized by a timeout limit τ . S proceeds as follows:

Functionality F_{SC}

The functionality is executed by n players with identities (id_0 ,. . . , id_n), parameterized by a timeout limit t'.

Player checking-in: Wait to receive from each player with id_i (checkin, M, $walletadress_i$ SIG.vk_i) containing the maximum balance intended to be played, and their signature verification key.

Player checking-out: Upon receiving (checkout, balance, σ) from $Player_i$, verify that σ contains valid signatures. If everything is correct, send (payout, balance) to $Player_i$. Then, send (checkedout, i, balance) to $Player_0$.

Recovery: Upon receiving a recovery request (report, $Player_i$, sk_i, $Block_z$) from $Player_i$ containing some unverifiable signatures in another block create a transaction with all of the above in Roulettechain and wait a timeout limit *t'*. Should the transaction be included within the timeout, then send (payout, balance[challenging] + balance[defending]) to $Player_{challenging}$ and (payout, 0) to to $Player_{defending}$, otherwise send (payout, balance[challenging] + balance[defending]) to $Player_{defending}$ and (payout, 0) to to $Player_{challenging}$.

Figure 6.5: Functionality F_{SC}

Security of $\pi_{roulette}$

Theorem 1. Assuming that the DDH problem is hard and that the digital signature scheme SIG is EUF-CMA secure, protocol $\pi_{Roulette}$ securely computes $F_{Roulette}$ in the F_{SC} -hybrid, random oracle model in the presence of malicious static adversaries and under the following assumptions:

- Roulettechain has liveliness and robustness.
- Honest majority on Roulettechain.
- There exists 1-honest player during the execution of the protocol.

Proof. In order to prove Theorem 1 we construct a non-uniform expected probabilistic polynomial time simulator (ideal adversary) S that interacts with the ideal functionality $F_{Roulette}$ and internal copies of at most one corrupted party. S simulates the actions of the other honest parties and the functionality F_{SC}. Let H denote the set of honest parties and C denote the set of corrupted parties in the internal execution run by S. S is parameterized by the timeout limit τ . S proceeds as follows:

Player Check-in: S simulates F_{SC} internally as well as the parties. Whenever a corrupted party $P_c \epsilon$ C sends a message (checkin, balance[i])) to F_SC, S acknowledges the check-in of that party with $F_{Roulette}$. Whenever an honest party $P_h \epsilon$ H checks-in with $F_{Roulette}$, S is informed and simulates $\pi_{Roulette}$'s check-in procedure for that party. If some party fails to check-in within the timeout limit, S allows the parties to dropout from $F_{Roulette}$ and reclaim their coins.

Player Check-out: S simulates F_{SC} internally as well as the honest parties. If a corrupted party P_c performs a check-out in the internal execution, S performs P_c's check-out on $F_{Roulette}$ and use the received coins to pay P_c . If an honest player P_h is able to check-out in the internal execution, then S allows P_h's check-out from $F_{Roulette}$ to proceed. S follows the same recovery triggers as the real protocol to activate the recovery phase.

Betting Phase: During the betting phase, S receives the bets of the honest parties from $F_{Roulette}$ and simulates the respective actions of the honest parties in the internal simulation. Whenever a corrupted party performs an action in the internal simulation, S forwards that action to $F_{Roulette}$. S follows the same recovery triggers as the real protocol to activate the recovery phase.

Result Tallying Phase: During the betting phase, S receives the secret keys of the honest parties from $F_{Roulette}$ and simulates the respective actions of the honest parties in the internal simulation. Whenever a corrupted party performs an action in the internal simulation, S forwards that action to $F_{Roulette}$. S follows the same recovery triggers as the real protocol to activate the recovery phase.

Recovery: S emulates F_{SC} and simulates the behavior of the honest parties according to the procedures described above for the respective part of the protocol. If a timeout occurs S perform the check-out for that player or if a misbehavior is detected S performs recovery: S aborts the execution in $F_{Roulette}$, thus publishing on the parent chain the evidence. Otherwise, S returns to the normal execution.

Simulator Analysis: Notice that the simulator S conducts a simulation with internal copies of the corrupted parties by emulating F_{SC} and executing the protocol exactly as an honest party would do for most of the protocol, except for phases where bets are placed and the keys are revealed. In these phases, S during the betting phase waits for information on the signed and encrypted bets in the form of commitments and in the next phase according to the keys received calculates the result which the round is supposed to have by $F_{Roulette}$ and then produces bogus secret keys that result into the ElGamal ciphertext that reprisents the bet of a player being decrypted to the value obtained from $F_{Roulette}$. S is able to do this because it can use the simulators for the signatures used in $\pi_{Roulette}$ to produce a valid signature showing that the bogus public key is valid even without knowing the respective secret one. As the secret key of the player is never revealed until the reveal phase, it is clear that the encrypted bets are indistinguishable from a random element of the same group. Hence, the execution with S could only be distinguished from the real execution if the signature generated by their respective simulators are distinguishable from an actual real world signature generated by a player. This is clearly not the case, since distinguishing the signatures generated by their simulator from those generated by real world parties who know the secret key would break the properties of the signature scheme.

In a more informal way, upon making a request(check-in or check-out), calculating players, betting and result tallying, S has total control over all communication and simulates all actions with timeout limit τ. The execution of S could only be distinguished from the real execution if S could forge signatures or decrypt ciphertexts without having firtsly be communicatted the respective secret key. This is clearly not the case, since by doing that the simulator would have broken the hard cryptographic properties or assumptions stated earlier either in the signature or in the encryption scheme.

Generalization to other games

In this section we discuss other games that may be executed by protocols constructed in the same setup, that is, an existing blockchain from a block of which an actor starts a pegged sidechain, hosts the game on an off-sidechain protocol and stores only vital information on it. By design the closest game to roulette is actually lottery because they both share the same mechanisms with their most different aspects being the payouts of the players. One could argue that roulette is a distinct case of lottery games and that would not have been a false statement. This means that by changing only the result-tallying function and the algorithm for calculating players, with respect to the numbers of choices players have and the maximum payout of each bet, one can construct the lottery protocol of their choice without needing to change anything else from our design.

Craps can also be considered a game close to roulette. In fact the only difference they have is that of the generation of randomness for the game. So as long as one modifies accordingly the payouts and the choices of the players they would be able to execute the game of craps as well.

Lastly, another application of this type of protocol is on specific card games. This derives from the fact that one.vs.one type of games are commonly found in card games and from the fact that by using a 256-bit string one can represent all possible shuffled decks of cards ($52! < 2^{256}$). Blackjack is a game that could be executed by our protocol with some tweaks, their major ones being, a setupcalculation phase would be needed after the calculating players phase and a hand execution phase after that. Of course the algorithm and the payouts should be corrected as well.

Chapter 7

Conclusions

In this thesis a new innovative protocol for the game of roulette was presented. In particular, the use of a pegged sidechain on which crucial information about the execution of the game would be stored alongside a recovery mechanism for penalizing deviations from the protocol was used. All of the results are based on the indistinguishably argument, in example, players are limited on what they can distinguish, or in a positive manner, players need enough information in order to be able to distinguish the scenario they are in.

A number of open questions arise from this work and are described below:

- Implementation of the protocol into a cryptoeconomic system via a smart contract.
- Generalizing the protocol to be able to execute generic one.vs.one lucky games.
- Optimize the mechanism for transferring assets from one blockhain to another.

Bibliography

[1] A. Kiayias, "Διάλεξη 8 - Κρυπτογραφία: Αρχές και πρωτόκολλα." Available online at: http://cgi.di.uoa.gr/~aggelos/crypto/page9/assets/8_publickey_handout_gr.pdf, 2008.

[2] Monax, "Blockchain explainer," 2016. Retrieved from https://monax.io/explainers/blockchains/.

[3] Juri Mattila, "The blockchain phenomenon: The disruptive potential of distributed consensus architectures." BERKELEY ROUNDTABLE ON THE INTERNATIONAL ECONOMY (BRIE), 2016.

[4] Panagiotis Grontas, Aris Pagourtzis, Alexandros Zacharakis and Bingsheng Zhang, "Towards everlasting privacy and efficient coercion resistance in remote electronic voting," 2018. In 3rd Workshop on Advances in Secure Electronic Voting, in assocation with Financial Cryptography 2018.

[5] Panagiotis Grontas, Aris Pagourtzis and Alexandros Zacharakis, "Coercion resistance in a practical secret voting scheme for large scale elections," 2017. In Proceedings of ISPAN-FCST-ISCC 2017, pp. 514–519, 2017.

[6] Vitalik Buterin, "Visions, part 1: The value of blockchain technology." Available online at: https://blog.ethereum.org/2015/04/13/visions-part-1-the-value-of-blockchain-technology/, 2015.

[7] IBM, "How does blockchain work?." Available online at: https://www.ibm.com/blockchain/what-is-blockchain, 2016.

[8] Adam Back, Matt Corallo, Luke Dashjr, Mark Friedenbach, Gregory Maxwell, Andrew Miller, Andrew Poelstra, Jorge Timón, and Pieter Wuille, "Enabling blockchain innovations with pegged sidechains." Available online at: https://blockstream.com/sidechains.pdf, 2014.

[9] A. Hill, "Introducing liquid: Bitcoin's first production sidechain." Available online at: https://blockstream.com/2015/10/12/introducing-liquid.html, 2015.

[10] Stathis Zachos, Aris Pagourtzis and Panagiotis Grontas, "Computational cryptography [ebook]." Available Online at: "http://hdl.handle.net/11419/5439", 2015.

[11] Berry Schoenmakers, "Lecture notes on cryptographic protocols." Available Online at: "http://www.win.tue.nl/~berry/2DMI00/", 2018.

[12] Barak, B.; and Sahai, A., “How to play almost any mental game over the net - concurrent composition using super-polynomial simulation..” In Proceedings of FOCS ’05, 2005.

[13] ACM Press, *The knowledge complexity of interactive proof-systems*, Proceedings of the seventeenth annual ACM symposium on Theory of computing, 1985.

[14] Barak, B.; and Mahmoody-Ghidary, M, “Lower bounds on signatures from symmetric primitives..” In Proceedings of FOCS ’07, 2007., 2007.

[15] Springer-Verlag New York, *A public key cryptosystem and a signature scheme based on discrete logarithms*, Proceedings of CRYPTO 84 on Advances in cryptology, 1985.

[16] Andreas V. Meier, “The elgamal cryptosystem.” Available online at: http://wwwmayr.in.tum.de/konferenzen/Jass05/courses/1/papers/meier_paper.pdf, 2005.

[17] S.M. van den Broek, “Digital signatures and the public Key infrastructure,” 1999.

[18] Alexandros Zacharakis, Panagiotis Grontas and Aris Pagourtzis, “Conditional blind signatures,” 2017. In 7th International Conference on Algebraic Informatics (short version). Full version available on: ”http://eprint.iacr.org/2017/682”.

[19] H. Okada, S. Yamasaki, and V. Bracamonte, “Proposed classification of blockchains based on authority and incentive dimensions,” *2017 19th International Conference on Advanced Communication Technology (ICACT)*, pp. 593–597, 2017.

[20] ISO/TC 307, “Blockchain and electronic distributed ledger technologies,” 2016.

[21] Satoshi Nakanoto, “Bitcoin: A peer-to-peer electronic cash system,” 2008.

[22] Jonatan H. Bergquist, “Blockchain technology and smart contracts, privacy-preserving tools,” 2017.

[23] David Siegel, “Understanding the dao attack.” Available online at: https://www.coindesk.com/understanding-dao-hack-journalists/, 2016.

[24] Ethereum community, “Ethereum homestead documentation.” Available online at: https://media.readthedocs.org/pdf/ethereum-homestead/latest/ethereum-homestead.pdf, 2017.

[25] Nick Szabo, “Smart contracts.” Available online at: http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_2.html, 1994.

[26] A. E. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," *IACR Cryptology ePrint Archive*, vol. 2015, p. 675, 2015.

[27] G. Zyskind, O. Nathan, and A. Pentland, "Decentralizing privacy: Using blockchain to protect personal data," in *2015 IEEE Symposium on Security and Privacy Workshops, SPW 2015, San Jose, CA, USA, May 21-22, 2015*, pp. 180–184, 2015.

[28] B. David, P. Gazi, A. Kiayias, and A. Russell, "Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain," in *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, pp. 66–98, 2018.

[29] Eric Wall, Gustaf Malm, "Using blockchain technology and smart contracts to create a distributed securities depository," 2016.

[30] R. M. Needham and M. D. Schroeder, "Using encryption for authentication in large networks of computers," *Commun. ACM*, vol. 21, no. 12, pp. 993–999, 1978.

[31] G. Lowe, "Breaking and fixing the needham-schroeder public-key protocol using FDR," *Software - Concepts and Tools*, vol. 17, no. 3, pp. 93–102, 1996.

[32] Yevgeniy Dodis, Aristeidis Tentes, "Lecture 14." Available online at: https://cs.nyu.edu/courses/fall09/G22.3220-001/scribe/lecture14.pdf, 2009.