ΚΡΥΠΤΟΓΡΑΦΙΑ ΕΛΛΕΙΠΤΙΚΩΝ ΚΑΜΠΥΛΩΝ ΚΑΙ Η ΕΦΑΡΜΟΓΗ ΤΗΣ ΣΤΟ BITCOIN

Επιβλέπων Καθηγητής:
Άγγελος Κιαγιάς

Γιώργος Καρυστιανός
ΜΠΛΑ

ΠΕΡΙΕΧΟΜΕΝΑ

3.4 Bitcoin Contracts

3.5 Secure Multiparty Computations στο Bitcoin

1 Εισαγωγή

Η Κρυπτογραφία Δημοσίου Κλειδιού αποτελεί σήμερα ένα πολύ έντονα αναπτυσσόμενο τομέα ασφάλειας και έχει αρχίσει να βρίσκει εφαρμογές σε αρκετές διαφορετικού τύπου συνθήκες, ειδικότερα ως μέθοδος ταυτοποίησης (authentication) και ψηφιακής υπογραφής. Αυτού του τύπου η κρυπτογραφία παρέχει μεγάλο βαθμό προστασίας των δεδομένων και θεωρείται ανώτερη από την κρυπτογραφία συμμετρικού κλειδιού αφού λύνει δύο βασικά προβλήματα της δεύτερης. Αυτά τα προβλήματα εντοπίζονται στη διανομή και στη διαχείριση κρυπτογραφικών κλειδιών. Παρόλα αυτά, η Κρυπτογραφία Δημοσίου Κλειδιού εμφανίζει, και αυτή με τη σειρά της, κάποια σημαντικά προβλήματα. Τα προβλήματα αυτά σχετίζονται με το πολύπλοκο μαθηματικό υπόβαθρο που χρησιμοποιεί αφού για την κρυπτογράφηση-αποκρυπτογράφηση και ψηφιακή υπογραφή δεδομένων απαιτούνται πολλές δαπανηρές, σε πόρους υλικού, αριθμητικές πράξεις (modulo πολλαπλασιασμός, αντιστροφή). Το πρόβλημα αυτό επιδεινώνεται συνυπολογίζοντας το γεγονός ότι το μήκος των κλειδιών σε αυτού του τύπου την κρυπτογραφία έχει πολύ μεγάλο μέγεθος έτσι ώστε να διασφαλιστεί ένα υψηλό επίπεδο ασφαλείας (π.χ. RSA: 1024 bit). Λύση στα παραπάνω προβλήματα δίνεται με βελτιστοποίηση σχεδιασμού των αριθμητικών πράξεων που απαιτούνται σε ένα κρυπτογράφημα δημοσίου κλειδιού και με τη χρήση ελλειπτικών καμπυλών αφού με αυτό τον τρόπο χρειαζόμαστε μικρότερου μήκους κλειδιά για την επίτευξη του ίδιου επιπέδου ασφαλείας.

Στα πλαίσια αυτής της Διπλωματικής εργασίας θα δούμε και θα αναλύσουμε την Κρυπτογραφία Ελλειπτικών Καμπυλών, τους αλγόριθμους κρυπτογράφησης της και τις ψηφιακές υπογραφές. Στην συνέχεια θα δούμε κάποιες βελτιώσεις στην απόδοση των αριθμητικών πράξεων χρησιμοποιώντας προβολικές συντεταγμένες και θα μελετήσουμε κάποιες μεθόδους που κάνουν τα Κρυπτοσυστήματα Ελλειπτικών Καμπυλών ανθεκτικά σε Side Channel Attacks, οι οποίες θεωρούνται αρκετά αποδοτικές κρυπταναλυτικές μεθόδους και αποτελούν παράμετρο σχεδιασμού ενός κρυπτογραφήματος μιας και εκμεταλλεύονται ιδιότητες του υλικού πάνω στο οποίο είναι σχεδιασμένο και υλοποιημένο ένα τέτοιο σύστημα.

Θα μιλήσουμε για το Bitcoin το οποίο είναι ένα Ψηφιακό νόμισμα που αξιοποιεί τεχνικές κρυπτογράφησης για την παραγωγή μονάδων αξίας και την επαλήθευση συναλλαγών, χωρίς τη διαμεσολάβηση κεντρικής τράπεζας. Το 2014, το διαδικτυακό λεξικό της Οξφόρδης συμπεριέλαβε για πρώτη φορά τον παραπάνω ορισμό της λέξης “κρυπτονόμισμα”. Ο ορισμός αυτός περιγράφει τα τρία βασικά χαρακτηριστικά ενός νομίσματος όπως το Bitcoin, χαρακτηριστικά που το κάνει να διαφέρει από τα παραδοσιακά flat currency όπως το ευρώ ή το δολάριο.

Το Bitcoin χρησιμοποιεί peer-to-peer τεχνολογία για να λειτουργεί χωρίς κάποια κεντρική αρχή, η διαχείριση των συναλλαγών και η έκδοση των χρημάτων διενεργείται συλλογικά από το ίδιο το δίκτυο. Θα δούμε το πρόβλημα της Malleability ιδιότητας στο Bitcoin και πως αυτό μπορεί να αντιμετωπιστεί. Ένα κρυπτογραφικό πρωτόκολλο είναι malleable αν η έξοδος του C μπορεί να μετασχηματιστεί (mauled) σε κάποια παρόμοια τιμή C' από κάποιον που δεν γνωρίζει τα μυστικά που χρησιμοποιήθηκαν για την παραγωγή του C. Τέλος θα δούμε πως μπορούμε να χρησιμοποιήσουμε το Bitcoin για Multiparty Computations πρωτόκολλα. Τα Multiparty Computation πρωτόκολλα (MPC) επιτρέπουν σε μια ομάδα από μέλη τα οποία δεν εμπιστεύεται το ένα το άλλο να υπολογίσουν μια κοινή συνάρτηση f με τις μυστικές τους εισόδους.

2 Κρυπτογραφία Ελλειπτικών Καμπυλών

2.1 Μαθηματικό Υπόβαθρο

2.1.1 Θεωρία Ομάδων

Ορίζουμε σαν προσθετική ομάδα $(G, +)$ ένα σύνολο στοιχείων G πάνω στα οποία μπορεί να εφαρμοστεί η πράξη της πρόσθεσης +. Αυτό σημαίνει ότι το $(G, +)$ έχει τις παρακάτω ιδιότητες**:**

- Προσεταιριστική ιδιότητα: Ισχύει $(a + b) + c = a + (b + c)$ για όλα τα $a, b, c \in G$
- Ταυτότητα: Υπάρχει ένα στοιχείο $0 \in G$ τέτοιο ώστε $a + 0 = 0 + a = a$ για κάθε $a \in G$
- Αντιστροφή: Για κάθε $a \in G$ υπάρχει ένα στοιχείο $-a \in G$ που ονομάζουμε προσθετικό αντίστροφο, τέτοιο ώστε $a + (-a) = -a + a = 0$

Ομοίως, μπορούμε να ορίσουμε και την πολλαπλασιαστική ομάδα $(G,\times)$ σαν εκείνη που έχει ένα σύνολο στοιχείων G πάνω στα οποία μπορεί να εφαρμοστεί η πράξη του πολλαπλασιασμού $\times$. Μια τέτοια ομάδα έχει τις παρακάτω ιδιότητες:

- Προσεταιριστική ιδιότητα: Ισχύει $a \times (b \times c) = (a \times b) \times c$ για όλα τα $a, b, c \in G$.
- Ταυτότητα: Υπάρχει ένα στοιχείο $1 \in G$ τέτοιο ώστε $a \times 1 = 1 \times a = a$ για όλα τα $a \in G$
- Αντιστροφή: Για κάθε $a \in G$ υπάρχει ένα στοιχείο $a^{-1} \in G$ που ονομάζουμε πολλαπλασιαστικό αντίστροφο, τέτοιο ώστε $a \times a^{-1} = a^{-1} \times a = 1$

Μία ομάδα λέγεται αβελιανή όταν ισχύει η αντιμεταθετικότητα δηλαδή όταν $a + b = b + a$ ή/και $a \times b = b \times a$ για όλα τα $a, b \in G$, ανάλογα με την αριθμητική πράξη που ορίζει την ομάδα αυτή.

Μια ομάδα λέγεται κυκλική όταν υπάρχει ένα στοιχείο $g \in G$, που λέγεται γεννήτορας της ομάδας, τέτοιο ώστε όλα τα στοιχεία της ομάδας να μπορούν να βρεθούν αν εφαρμοστεί επαναληπτικά σε αυτό το στοιχείο g η αριθμητική πράξη που ορίζει την ομάδα αυτή. Σε μια αθροιστική ομάδα ισχύει ότι τα στοιχεία της θα είναι $G = \{0, g, 2g, 3g, 4g, ..\}$ ενώ σε μια πολλαπλασιαστική ομάδα ισχύει ότι τα στοιχεία της θα είναι $G = \{1, g, g^2, g^3, g^4, \dots\}$.

Ένα σύνολο R ονομάζεται δακτύλιος όταν για όλα τα στοιχεία του ισχύουν ο πολλαπλασιασμός και η πρόσθεση και επίσης ισχύει:

- Προσεταιριστική ιδιότητα ως προς τον πολλαπλασιασμό: $a \times (b \times c) = (a \times b) \times c$ για όλα τα $a, b, c \in R$
- Το R είναι αβελιανό ως προς την πρόσθεση: $a + b = b + a$ για όλα τα $a, b \in R$
- Επιμεριστική ιδιότητα ως προς την πρόσθεση: $a \times (b + c) = (a \times b) + (a \times c)$ και $(b + c) \times a = (b \times a) + (c \times a)$ για όλα τα $a, b, c \in R$
- Επιμεριστική ιδιότητα ως προς τον πολλαπλασιασμό: $a + (b \times c) = (a + b) \times (a + c)$ και $(b \times c) + a = (b + a) \times (c + a)$ για όλα τα $a, b, c \in R$

Αξίζει να παρατηρηθεί ότι για ένα δακτύλιο δεν ισχύει η πράξη της διαίρεσης ή της αντιστροφής αφού δεν υπάρχει πάντα πολλαπλασιαστικό αντίστροφο για κάθε στοιχείο του δακτυλίου αυτού. Πιο συγκεκριμένα, ένας δακτύλιος μπορεί να έχει μηδενικούς διαιρέτες δηλαδή μη μηδενικά στοιχεία a, b για τα οποία ισχύει $ab = 0$. Το σύνολο των στοιχείων ενός δακτυλίου R για τα οποία υπάρχει ένα πολλαπλασιαστικό αντίστροφο συμβολίζονται με R^* και αποτελεί μια πολλαπλασιαστική ομάδα.

Ένας δακτύλιος λέγεται αντιμεταθετικός όταν ο πολλαπλασιασμός για τα στοιχεία αυτού του δακτυλίου είναι αντιμεταθετικός.

Μία ομάδα που έχει στοιχεία ορισμένα τόσο για άθροιση όσο και για πολλαπλασιασμό λέγεται σώμα $(F, +, \times)$ και έχει τις παρακάτω ιδιότητες:

- $(F, +)$ είναι μια αβελιανή ομάδα, δηλαδή ισχύει ότι $a + b = b + a$, με στοιχείο ταυτότητας το *0*.
- Η πράξη του πολλαπλασιασμού $\times$ είναι προσεταιριστική στο F, δηλαδή ισχύει $(a \times b) \times c = a \times (b \times c)$ για όλα τα $a, b, c \in F$
- Υπάρχει ένα στοιχείο ταυτότητας $1 \in F$ όπου $1 \neq 0$ τέτοιο ώστε $1 \times a = a \times 1 = a$ για όλα τα $a \in F$
- Η πράξη του πολλαπλασιασμού $\times$ είναι επιμεριστική πάνω στην πρόσθεση $+$, δηλαδή ισχύει ότι $a \times (b + c) = (a \times b) + (a \times c)$ και $(b + c) \times a = (b \times a) + (c \times a)$ για όλα τα $a, b, c \in F$.
- $(F, \times)$ είναι αβελιανή ομάδα, δηλαδή ισχύει ότι $a \times b = b \times a$, με στοιχείο ταυτότητας το *1*.
- Για κάθε $a \neq 0$, $a \in F$, υπάρχει ένα στοιχείο $a^{-1} \in F$ τέτοιο ώστε $a^{-1} \times a = a \times a^{-1} = 1$.

Η διαφορά μεταξύ ενός σώματος και ενός δακτυλίου είναι ότι το πρώτο έχει πάντα πολλαπλασιαστικό αντίστροφο και μη μηδενικούς διαιρέτες.

Ένα σώμα K λέγεται σώμα επέκτασης ενός σώματος F όταν $F \subset K$. Σε αυτήν την περίπτωση ονομάζουμε το F υποσώμα του K. Συνήθως, δηλώνουμε ότι το σώμα K είναι σώμα επέκτασης του F με K/F ή $[K: F]$. Επεκτάσεις K του σώματος F μπορούν να κατασκευαστούν αν προσθέσουμε σε αυτό ένα αλγεβρικό αριθμό x, έναν αριθμό δηλαδή που αποτελεί ρίζα ενός πολυωνύμου $F(x)$ με συντελεστές που ανήκουν στο F. Ένα στοιχείο της επέκτασης K ενός σώματος F έχει συνεπώς την παρακάτω μορφή:

$$a = \sum_{i=0}^{k-1} a_i x^i = a_0 + a_1 x^1 + a_2 x^2 + \ldots + a_{k-1} x^{k-1} \text{ όπου } a_i \in F.$$

Το k καλείται βαθμός του πολυωνύμου $F(x)$ αλλά και βαθμός του σώματος επέκτασης K. Οι αριθμοί $[1, x, x^2, \ldots, x^{k-1}]$ αποτελούν μια βάση για όλα τα στοιχεία του σώματος επέκτασης K. Κάθε στοιχείο του K μπορεί να προκύψει από την άθροιση κάποιων από τους αριθμούς της βάσης.

Υπάρχουν δύο είδη σωμάτων, τα πεπερασμένα ή κλειστά σώματα και τα άπειρα σώματα. Τα άπειρα σώματα έχουν έναν άπειρο αριθμό στοιχείων σε αντίθεση με τα πεπερασμένα σώματα που έχουν ένα συγκεκριμένο, πεπερασμένο αριθμό στοιχείων. Τα πεπερασμένα σώματα ονομάζονται επίσης και σώματα Galois ή $GF(p)$ προς τιμήν του μαθηματικού που έθεσε τις βάσεις τους (Everiste Galois 1811-1832). Τα πεπερασμένα σώματα είναι εξαιρετικά χρήσιμα σε μια πλειάδα από διαφορετικές υπολογιστικές εφαρμογές που περιλαμβάνουν την κρυπτογραφία και την ανίχνευση λαθών κωδικοποίησης. Γενικά τα σώματα αυτά έχουν την παρακάτω μορφή $GF(p) = \{0, 1, \ldots, p-1\}$.

Ένα πεπερασμένο σώμα $GF(p)$ υπάρχει όταν p είναι ένας πρώτος αριθμός που ονομάζεται χαρακτηριστική (characteristic) του πεπερασμένου σώματος και συμβολίζεται σαν $Char(GF(p))$. Μπορούμε να ορίσουμε πεπερασμένα σώματα επέκτασης $GF(q)$ πάνω στο $GF(p)$ εκείνα τα σώματα που έχουν $q = p^k$. Ο αριθμός k είναι ένα θετικό ακέραιο νούμερο και αποτελεί τον βαθμό του περασμένου σώματος επέκτασης. Αν $k = 1$ τότε το πεπερασμένο σώμα και το σώμα επέκτασης ταυτίζονται. Αυτά τα πεπερασμένα σώματα ονομάζονται πρώτα σώματα ή $GF(p)$. Όταν $p = 2$, τότε το πεπερασμένο σώμα επέκτασης, που συμβολίζεται σαν $GF(2^k)$, ονομάζεται πεπερασμένο σώμα δυαδικής επέκτασης και εμφανίζει χαρακτηριστικά πολύ χρήσιμα σε εφαρμογές υπολογιστών. Οι αριθμητικές πράξεις πάνω σε ένα τέτοιο σώμα ($GF(2^k)$) είναι απλοποιημένες σε σχέση με τις πράξεις που ορίζονται στα $GF(p)$ ή πρώτα σώματα. Τέλος οφείλουμε να ορίσουμε την τάξη ενός πεπερασμένου σώματος επέκτασης, $Order(GF(q))$, σαν τον συνολικό αριθμό των στοιχείων του σώματος αυτού.

Ένα στοιχείο $a \in GF(2^k)$ ορίζεται πάνω σε μια βάση B της μορφής $B = \{b_{k-1}, b_{k-2}, \ldots, b_1, b_0\}$ Κατά συνέπεια, το στοιχείο a μπορεί να γραφτεί σαν γραμμικός συνδυασμός των b_i της βάσης αυτής ως:

$$a = a_{k-1} b_{k-1} + a_{k-2} b_{k-2} + \cdots + a_1 b_1 + a_0 b_0$$

Αφού η χαρακτηριστική του $GF(2^k)$ $Char(GF(2^k)) = 2$, οι συντελεστές a_i του a που έχει αναπαρασταθεί από την βάση B, είναι ορισμένοι στο $GF(2)$ και έτσι παίρνουν τιμές μηδέν ή ένα. Το στοιχείο a μπορεί ακόμα να περιγραφεί σε διανυσματική μορφή σαν $a = \{a_{k-1}, a_{k-2}, \dots, a_1, a_0\}$.

Η βάση B για την αναπαράσταση στοιχείων του $GF(2^k)$ μπορεί να έχει πολλές διαφορετικές μορφές. Κάθε τέτοια μορφή βάσης καθορίζει την αλληλεπίδραση ενός στοιχείου του σώματος σε σχέσεις με τα υπόλοιπα στοιχεία του σώματος αυτού. Έτσι, η επιλογή της $GF(2^k)$ βάσης επηρεάζει δραστικά της μαθηματικές πράξεις μεταξύ στοιχείων του $GF(2^k)$. Για το λόγο αυτό, η επιλογή της βάσης αναπαράστασης στοιχείων του $GF(2^k)$ παίζει σημαντικό ρόλο στην απόδοση αρχιτεκτονικών υλικού για τις διάφορες μαθηματικές πράξεις του $GF(2^k)$.

Οι πιο υποσχόμενες βάσεις για αναπαράσταση των $GF(2^k)$ στοιχείων είναι η πολυωνυμική βάση, η κανονική βάση (Normal Basis) και η διπλή βάση αναπαράστασης. Από αυτές τις βάσεις αναπαράστασης, οι πιο ευρέως χρησιμοποιούμενες είναι η πολυωνυμική βάση και μια ειδική μορφή κανονικής βάσης που ονομάζεται βέλτιστη κανονική βάση (Optimal Normal Basis, ONB).

2.1.2 Σχέσεις Ισοδυναμίας και Αριθμητική Υπολοίπων (moduli)

Μια σχέση ισοδυναμίας πάνω σε ένα σύνολο S είναι ένα υποσύνολο $S\ x\ S$ του οποίου τα στοιχεία (a, b) γράφονται σαν $a \sim b$ και έχουν τις παρακάτω ιδιότητες:

- Ανακλαστική: $\alpha \sim \alpha$ για όλα τα $a \in S$
- Συμμετρική: $\alpha \sim b \Rightarrow b \sim \alpha$ για όλα τα $a, b \in S$
- Μεταβατική: $\alpha \sim b$ και $b \sim c$ τότε $a \sim c$ για όλα τα $a, b, c \in S$

Για ένα σύνολο S στο οποίο υπάρχει μια σχέση ισοδυναμίας μπορεί να οριστεί η έννοια της κλάσης ισοδυναμίας. Πιο συγκεκριμένα, η κλάση ισοδυναμίας του a, όπου $a \in S$, ορίζεται σαν $a = \{b \in S : b \sim a\}$. Ένα σύνολο S μπορεί να κατακερματιστεί σε κλάσεις ισοδυναμίας αφού αν ισχύει ότι $\alpha \sim b$ τότε $\bar{a} = \bar{b}$ ενώ αν δεν ισχύει ότι $\alpha \sim b$ τότε $\bar{a} \cap \bar{b} = 0$. Ένα οποιοδήποτε μέλος μιας κλάσης ισοδυναμίας ονομάζεται αντιπρόσωπος αυτής της κλάσης. Δύο αριθμοί $a, b \in Z$, όπου Z το σύνολο των ακεραίων, λέγονται ισοδύναμοι $modulo\ m$ (congruent) όταν η διαφορά τους $b - a$ διαιρείται απόλυτα με το m $(m \mid b - a)$. Αυτό συμβολίζεται σαν $a \equiv b\ (mod m)$.

Η σχέση των ισοδύναμων $modulo\ m$ είναι μια σχέση ισοδυναμίας με τις παρακάτω ιδιότητες:

- Ανακλαστική: $a \equiv a\ (mod m)$ αφού $m \mid \alpha - a = 0$
- Συμμετρική: Αν $a \equiv b\ (mod m) \Longrightarrow b \equiv a\ (mod m)$ αφού αν $m \mid b - a$ τότε $m \mid \alpha - b$
- Μεταβατική: Αν $a \equiv b\ (mod m)$ και $b \equiv c\ (mod m)$ τότε $a \equiv c\ (mod m)$ $a \sim c$ αφού αν $m \mid \alpha - b$ $m \mid b - c$ θα ισχύει ότι $m \mid (\alpha - b) + (b - c) \Longrightarrow m \mid b - c$.

Το σύνολο των κλάσεων ισοδυναμίας για τους ακεραίους (ισοδύναμες κλάσεις) είναι ακριβώς $\{\overline{0}, \overline{1}, \ldots, \overline{m-1}\}$. Αυτό το σύνολο συμβολίζεται σαν Z/mZ ή αλλιώς Z_{m}. Συνήθως οι μπάρες παραλείπονται και το σύνολο Z_{m} παρουσιάζεται σαν $\mathrm{Z_m} = \{0, 1, \ldots, m-1\}$

Μπορούμε να πραγματοποιήσουμε πράξεις μεταξύ ισοδύναμων με τον ίδιο τρόπο που πραγματοποιούμε πράξεις μεταξύ ακεραίων λαμβάνοντας υπόψη μας ότι λόγω της $modulus$ λογικής της πράξης το πολλαπλάσιο ενός αριθμού δεν αλλάζει την ισοδύναμη κλάση.

Λήμμα 2.1. Αν $a_1 \equiv a_2\ (modm)$ και $b_1 \equiv b_2\ (modm)$ τότε ισχύει $a_1 \pm b_1 \equiv a_2 \pm b_1\ (modm)$ και $a_1\, b_1 \equiv a_2\, b_1\ (modm)$.

Λήμμα 2.2. Υποθέτουμε ότι $ka \equiv kb\ (modm)$ και $gcd(k, m) = d$. Τότε $a \equiv b\ (mod\ m/d)$.

Λήμμα 2.3. Υποθέτουμε ότι $ka \equiv kb\ (modm)$ και $gcd(k, m) = 1$. Τότε $a \equiv b\ (modm)$.

Λήμμα 2.4. Το a έχει πολλαπλασιαστικό αντίστροφο $modulo\ m$ μόνο όταν $gcd(a, m) = 1$ (το α και το m είναι πρώτοι μεταξύ τους). Αυτό το αντίστροφο είναι μοναδικό.

Αν υπολογιστεί το πολλαπλασιαστικό αντίστροφο $modulo\ m$ τότε μπορούμε να ορίσουμε την διαίρεση $modulo\ m$ σαν πολλαπλασιασμό του διαιρετέου με το αντίστροφο του διαιρέτη.

Αξίζει να σημειωθεί ότι όταν το m είναι πρώτος αριθμός τότε το σύνολο Z_m είναι ένα πεπερασμένο σώμα και πιο συγκεκριμένα ένα πρώτο σώμα $(GF(p)\ όπου\ p = m)$ αφού σε αυτήν την περίπτωση πολλαπλασιαστικό αντίστροφο για το Z_m υπάρχει πάντα και είναι μοναδικό.

2.2 Βασικές Αρχές Θεωρίας Ελλειπτικών Καμπύλων

2.2.1 Ελλειπτικές Καμπύλες σε Πεπερασμένα Σώματα

Η χρήση των Ελλειπτικών Καμπυλών στην Κρυπτογραφία αποτελεί μόνο ένα πολύ μικρό μέρος της γενικότερης θεωρίας Ελλειπτικών Καμπυλών. Περιορίζεται σε μη υπεριδιάζουσες καμπύλες ορισμένες πάνω σε πεπερασμένα σώματα και πιο συγκεκριμένα σε πρώτα σώματα $(GF(p))$ ή αλλιώς F_p και σώματα δυαδικής επέκτασης $(GF(2^k))$ ή αλλιώς $F_{2\kappa}$.

Μια Ελλειπτική Καμπύλη ορισμένη πάνω σε ένα σώμα F είναι το σύνολο των λύσεων (x, y), της μακράς εξίσωσης $Weierstrass$ $E: y^2 + a_1xy + a_3\, y = x^3 + a_2x^2 + a_4x + a_5$ μαζί με το σημείο στο άπειρο που ορίζεται σαν ∂. Οι μεταβλητές a_1, a_2, a_3, a_4, a_5 ανήκουν στο F και $\Delta \neq 0$ όπου Δ είναι η διακρίνουσα της εξίσωσης E. Περισσότερα για την μακρά εξίσωση $Weierstrass$ καθώς και για τις Ελλειπτικές Καμπύλες γενικότερα μπορούν να βρεθούν στα [30], [52], [61], [62], [66].

Δύο Ελλειπτικές Καμπύλες E_1 και E_2 ορισμένες πάνω στο F, είναι ισομορφικές μεταξύ τους αν υπάρχει ένα μετασχηματισμός μεταξύ κάποιου x και y της E_1 και κάποιου x', y' της E_2 που είναι της μορφής $x = u^2x' + r, y = u^3y' + su^2x' + t$, όπου $u, s, r, t \in F$. Αυτός ο μετασχηματισμός, αναφέρεται σαν επιτρεπόμενη αλλαγή μεταβλητών και οδηγεί σε μια διαφορετική εξίσωση περιγραφής της Ελλειπτικής Καμπύλης. Αυτή η εξίσωση λέγεται μικρή εξίσωση Weierstrass και η μορφής της σχετίζεται με τον τύπο του σώματος πάνω στο οποίο είναι ορισμένη η Ελλειπτική Καμπύλη.

Πριν παρουσιαστούν οι διάφορες μορφές της μικρής εξίσωσης $Weierstrass$ μίας Ελλειπτικής Καμπύλης ορισμένης πάνω σε πεπερασμένα σώματα, οι υπεριδιάζουσες και μη υπεριδιάζουσες Ελλειπτικές Καμπύλες πρέπει να οριστούν. Μια Ελλειπτική Καμπύλη E ορισμένη πάνω σε ένα πεπερασμένο σώμα F είναι υπεριδιάζουσα (supersingular) όταν η χαρακτηριστική του πεπερασμένου σώματος F που την ορίζει διαιρείται με το t, όπου t είναι το ίχνος (trace) του F. Αν η χαρακτηριστική του πεπερασμένου σώματος F δεν διαιρείται με το t τότε η E λέγεται μη-υπεριδιάζουσα (non-supersingular). Υπάρχουν στοιχεία που αποδεικνύουν ότι οι υπεριδιάζουσες Ελλειπτικές Καμπύλες είναι ασθενείς για Κρυπτογραφικές εφαρμογές [49] και συνεπώς δεν θα αποτελέσουν αντικείμενο περαιτέρω ανάλυσης.

Για τις μη-υπεριδιάζουσες Ελλειπτικές Καμπύλες ορισμένες πάνω στο $GF(p)$ του οποίου η χαρακτηριστική είναι $Char(GF(p)) \neq 2$ ή 3, η μικρή εξίσωση $Weierstrass$ έχει την παρακάτω μορφή $E: y^2 = x^3 + ax + b$, όπου $a, b \in GF(p)$ και $\Delta = -16(4a^3 + 27b^2) \neq 0$.

Ομοίως, για τις μη-υπεριδιάζουσες Ελλειπτικές Καμπύλες ορισμένες πάνω στο $GF(2^k)$ του οποίου η χαρακτηριστική είναι $Char(GF(2^k)) = 2$, η μικρή εξίσωση $Weierstrass$ έχει την παρακάτω μορφή $E: y^2 + xy = x^3 + ax^2 + b$, όπου $a, b \in GF(2^k)$ και $\Delta = b \neq 0$.

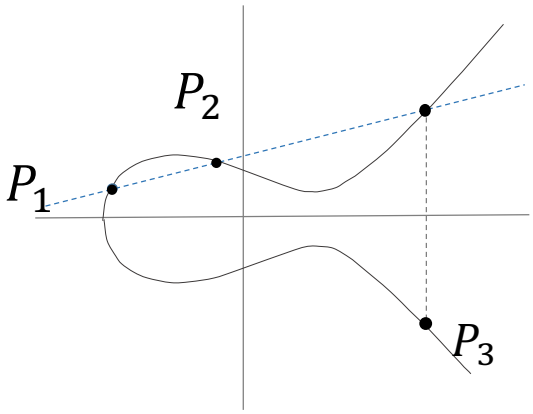
Υπάρχει ακόμα ένας ειδικός τύπος Ελλειπτικών καμπυλών ορισμένων πάνω σε $GF(2^k)$. Αυτές οι καμπύλες έχουν μικρή εξίσωση $Weierstrass$ της μορφής $E: y^2 + xy = x^3 + ax^2 + b$ όπου $a, b \in GF(2)$. Τέτοιες Ελλειπτικές Καμπύλες ονομάζονται Καμπύλες Koblitz και είναι ανώμαλες δυαδικές καμπύλες. Σε αυτές τις καμπύλες ο πολλαπλασιασμός μπορεί να πραγματοποιηθεί χωρίς τη χρήση κάποιας πράξης διπλασιασμού σημείων.

Ο αριθμός των πραγματικών σημείων πάνω σε μια Ελλειπτική Καμπύλη ορισμένη πάνω σε πεπερασμένο σώμα είναι και αυτός πεπερασμένος και συμβολίζεται σαν $\#E(GF(p))$ ή $\#E(GF(2^k))$ αντιστοίχως.

2.2.2 Πρόσθεση και Διπλασιασμός Σημείων Ελλειπτικής Καμπύλης (Νόμος Ομάδας)

Αν E είναι μια Ελλειπτική Καμπύλη ορισμένη πάνω σε ένα πεπερασμένο σώμα F, η πράξη της πρόσθεσης EK σημείων πραγματοποιείται όταν αθροίζουμε δύο σημεία της E έτσι ώστε να προκύψει ένα τρίτο σημείο πάνω στην E. Το σύνολο των σημείων Ελλειπτικής Καμπύλης μαζί με τον κανόνα της πρόσθεσης, σχηματίζουν μια αβελιανή ομάδα $E(F)$ του τύπου $(G, +)$ με στοιχείο ταυτότητας το σημείο στο άπειρο ∂. Ο διπλασιασμός EK σημείων ορίζεται σαν την άθροιση ενός σημείου με τον εαυτό του και μπορεί να θεωρηθεί ειδική περίπτωση της πρόσθεσης EK σημείων.

Οι δύο παραπάνω πράξεις πάνω στις Ελλειπτικές Καμπύλες πραγματοποιούνται χρησιμοποιώντας ένα γεωμετρικό κανόνα που ονομάζεται κανόνας χορδής και εφαπτομένης. Υποθέτουμε ότι έχουμε μια Ελλειπτική Καμπύλη E και δύο σημεία πάνω σε αυτήν, $P_1 = (x_1, y_1), P_2 = (x_2, y_2)$ και ότι θέλουμε να βρούμε ένα τρίτο σημείο πάνω στην Ελλειπτική Καμπύλη E που είναι το άθροισμα των P_1 και P_2, $P_3 = (x_3, y_3) = P_1 + P_2$. Σχεδιάζουμε μια γραμμή που ενώνει τα P_1 και P_2 και βρίσκουμε το σημείο που η ευθεία αυτή τέμνει την E. Το συμμετρικό αυτού του σημείου ως προς τον άξονα των χ είναι το ζητούμενο σημείο P_3. Ο διπλασιασμός ενός σημείου Ελλειπτικής Καμπύλης ακολουθεί ένα παρόμοιο κανόνα. Υποθέτουμε ότι έχουμε μια Ελλειπτική Καμπύλη E και ότι θέλουμε να βρούμε το διπλάσιο ενός σημείου $P_1 = (x_1, y_1)$ της E το οποίο ανήκει επίσης στην E και ορίζουμε σαν $P_3 = P_1 + P_2 = 2P_1 = (x_3, y_3)$. Σχεδιάζουμε την εφαπτομένη ευθεία στο P_1 και σημαδεύουμε το σημείο της E που αυτή η ευθεία τέμνει. Το συμμετρικό αυτού του σημείου ως προς τον άξονα των χ είναι το ζητούμενο σημείο P_3.



Σχήμα 1. Πρόσθεση Σημείων σε Ελλειπτική Καμπύλη

Ο παραπάνω γεωμετρικός κανόνας μπορεί να περιγραφεί με αλγεβρικές εξισώσεις. Τέτοιες εξισώσεις περιγράφουν την πρόσθεση και τον διπλασιασμό σημείων ακολουθώντας κανόνες αναλυτικής γεωμετρίας. Εξαρτώνται από την μορφή της εξίσωσης της Ελλειπτικής Καμπύλης και το πεπερασμένο σώμα πάνω στο οποίο ορίζεται η καμπύλη αυτή.

Για μη-υπεριδιάζουσες Ελλειπτικές Καμπύλες ορισμένες πάνω σε $GF(p)$, η χαρακτηριστική του σώματος είναι $Char(GF(p)) > 3$. Η μικρή εξίσωση $Weierstrass$ μιας τέτοιας Ελλειπτικής Καμπύλης θα είναι $E: y^2 = x^3 + ax + b$ και κατά συνέπεια μπορούμε να ορίσουμε τον διπλασιασμό ($P_3 = 2P_1$) και την άθροιση $P_3 = P_1 + P_1$ σημείων πάνω στην E ως ακολούθως:

Όταν $P_1 \neq P_2$(άθροιση σημείων) η κλήση λ της ευθείας μεταξύ P_1 και P_2 θα είναι $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$ για $x_1 \neq x_2$ και το σημείο $P_3 = P_1 + P_2 = (x_3, y_3)$ χαρακτηρίζεται από τις συνταταγμένες του ως εξής:

$$x_3 = \lambda^2 - x_1 - x_2$$

$$y_3 = \lambda(x_1 - x_3) - y_1 = \lambda(2x_1 + x_2 - \lambda^2) - y_1$$

Όταν $P_1 \neq P_2$αλλά $x_2 = x_1$ η κλήση είναι ∞, δηλαδή η ευθεία μεταξύ P_1 και P_2 είναι κάθετη στον άξονα των x και κατά συνέπεια τέμνει στην Ελλειπτική Καμπύλη στο σημείο στο άπειρο ∂. Σε αυτή την περίπτωση $P_3 = P_1 + P_2 = \partial$.

Όταν $P_1 = P_2$ (διπλασιασμός σημείων) και $y_1 \neq 0$, τότε η κλήση της εφαπτόμενης γραμμής στο P_1 θα είναι $\lambda = \frac{3{x_1}^2 + a}{2y_1}$ και το σημείο $P_3 = P_1 + P_2 = 2P_1 = (x_3, y_3)$ χαρακτηρίζεται από τις συντεταγμένες του ως εξής:

$$x_3 = \lambda^2 - x_1 - x_2 = \lambda^2 - 2x_1$$

$$y_3 = \lambda(x_1 - x_3) + y_1 = \lambda(3x_1 - \lambda^2) - y_1$$

Όταν $P_1 = P_2$ (διπλασιασμός σημείων) και $y_1 = 0$, η εφαπτομένη γραμμή είναι κάθετη στον άξονα των x και συνεπώς, $P_3 = 2P_1 = \partial$.

Αφού το σημείο στο άπειρο είναι στοιχείο ταυτότητας για την ομάδα $E(GF(p))$ ισχύει $P_3 = P_1 + \partial = P_1$. Η αφαίρεση μεταξύ σημείων πάνω στην Ελλειπτική Καμπύλη μπορεί να πραγματοποιηθεί κάνοντας πρόσθεση σημείων χρησιμοποιώντας το σημείο $-P_2$ αντί για P_2, όπου $-P_2 = (x_2, -y_2)$.

2.2.3 Πολλαπλασιασμός Σημείου με Ακέραιο (Scalar Πολλαπλασιασμός)

Αν E είναι μια Ελλειπτική Καμπύλη ορισμένη πάνω σε ένα πεπερασμένο σώμα τότε ο πολλαπλασιασμός μεταξύ ενός ακεραίου s και ενός σημείου P έχει ως αποτέλεσμα ένα νέο σημείο πάνω στην Ελλειπτική Καμπύλη $Q = sP$. Η πράξη αυτή ονομάζεται πολλαπλασιασμός σημείου με ακέραιο ή $scalar$ πολλαπλασιασμός. Ο $scalar$ πολλαπλασιασμός είναι μια επαναληπτική ακολουθία πράξεων πρόσθεσης και διπλασιασμού EK σημείων που ακολουθεί

τον παρακάτω αλγόριθμο.

Scalar Πολλαπλασιασμός

Input: P, s

Output: $Q = sP$

1. $Q = \partial$

2. $While\ s \neq 0\ do$

   2.1. $If\ s\ is\ even\ then\ s = s/2\ and\ P = 2P$

   2.2. $If\ s\ is\ odd\ then\ s = s - 1\ and\ Q = Q + P$

3. $Return\ Q$

2.3 Κρυπτογραφία Ελλειπτικών Καμπύλων

2.3.1 Παράμετροι Τομέα

Το πρώτο βήμα σε ένα σύστημα που χρησιμοποιεί κρυπτογραφία Ελλειπτικών Καμπυλών είναι η γέννηση μιας κρυπτογραφικά ασφαλούς Ελλειπτικής Καμπύλης και ο σαφής ορισμός της με συγκεκριμένες παραμέτρους. Αυτές οι παράμετροι ονομάζονται παράμετροι τομέα (Domain parameters). Έστω ότι υπάρχει ένα πεπερασμένο σώμα F_p και μια δημιουργημένη Ελλειπτική Καμπύλη $E(F_p)$ που χρησιμοποιεί το σώμα αυτό. Οι παράμετροι τομέα για την συγκεκριμένη καμπύλη αποτελούνται από τα εξής:

- την τάξη του πεπερασμένου σώματος F_p: $Order(\ F_p\ )$
- το σημείο $P\ =\ (x, y)$ πρώτης τάξεως, που ονομάζεται σημείο βάσης
- τους συντελεστές a, b που εμφανίζονται στην μικρή εξίσωση $Weierstrass\ E$ της Ελλειπτικής Καμπύλης
- την τάξη n του σημείου P
- την τιμή $h\ =\ \#E(F_p)/n$ που ονομάζεται συμπαράγοντας (cofactor).

Η γέννηση των παραμέτρων τομέα εμπεριέχει τον υπολογισμό του αριθμού των σημείων της Ελλειπτικής Καμπύλης $\#E(F_p)$, την επιβεβαίωση ότι το $\#E(F_p)$ διαιρείται ακριβώς με το n $(n\ \neq\ Order(F_p))$ και τον υπολογισμό του h αλλά και του σημείου $P\ \neq\ \infty$.

2.3.2 Αλγόριθμοι Κρυπτογράφησης

Όπως και στα υπόλοιπα Κρυπτοσυστήματα Δημοσίου Κλειδιού ορίζονται 3 αλγόριθμοι:

Παραγωγή Κλειδιών

INPUT: Elliptic curve domain parameters (p, a, b, P, n).

OUTPUT: Public Key Q and Private Key d.

1. Select $d\ \in\ [1, n-1]$.

2. Compute $Q\ =\ dP$.

3. Return (Q, d).

Το δημόσιο κλειδί είναι το σημείο Q, ενώ το ιδιωτικό κλειδί είναι η τιμή d. Το πρόβλημα υπολογισμού του d έχοντας τους παραμέτρους τομέα και το Q είναι το πρόβλημα του Διακριτού Λογαρίθμου στις Ελλειπτικές Καμπύλες ECDLP(Elliptic Curve Discrete Logarithm Problem). Το δημιουργημένο ζεύγος κλειδιών χρησιμοποιείται σε ECDH (Elliptic Curve Diffie Helman) πρωτόκολλα ανταλλαγής και αποκατάστασης κλειδιών.

ECDH (Elliptic Curve Diffie Helman)

1. Οι Alice και Bob συμφωνούν σε μία ελλειπτική καμπύλη E πάνω σε ένα πεπερασμένο σώμα F_q έτσι ώστε το πρόβλημα του διακριτού λογαρίθμου να είναι δύσκολα επιλύσιμο στο $E(F_q)$. Συμφωνούν επίσης σε ένα σημείο $P \in E(F_q)$ έτσι ώστε η υποομάδα που παράγεται από το P, να έχει μεγάλη τάξη (συνήθως, η καμπύλη και το σημείο επιλέγονται ώστε η τάξη να είναι ένας μεγάλος πρώτος).

2. Η Alice διαλέγει έναν ακέραιο d_A, τον οποίο γνωρίζει μόνο αυτή, υπολογίζει $Q_A = d_A P$ και στέλνει το Q_A στον Bob.

3. Ο Bob διαλέγει έναν ακέραιο d_B, τον οποίο γνωρίζει μόνο αυτός, υπολογίζει $Q_B = d_B P$ και στέλνει το Q_B στην Alice.

4. Η Alice υπολογίζει $Q_{AB} = d_A Q_B = d_A d_B P$

5. Ο Bob υπολογίζει $Q_{AB} = d_B Q_A = d_A d_B P$

6. Οι Alice και Bob χρησιμοποιούν μια μέθοδο που έχουν συμφωνήσει δημόσια για να εξάγουν το κλειδί Q_{AB}. Για παράδειγμα θα μπορούσαν να χρησιμοποιήσουν τα τελευταία 256 bits της $x - συντεταγμένης$ του $d_A d_B P$ ως κλειδί. Διαφορετικά θα μπορούσαν να χρησιμοποιήσουν ως συνάρτηση κατακερματισμού (Hash Function) την x-συντεταγμένη. Και στις δύο περιπτώσεις το κλειδί που εξάγεται θα ανήκει στο F_q.

Η μόνη πληροφορία που η Eve(επιτιθέμενος) δύναται να γνωρίζει είναι η ελλειπτική καμπύλη E, το πεπερασμένο σώμα F_q και τα σημεία P, Q_A και Q_B. Συνεπώς, για να αποσπάσει το μυστικό κλειδί Q_{AB} θα πρέπει να λύσει το πρόβλημα ECDHP.

Υπολογιστικό Diffie-Helman Πρόβλημα στις Ελλειπτικές Καμπύλες ECDHP (Elliptic Curve Diffie-Helman Problem)

Δίνεται: Μια ελλειπτική καμπύλη E, ορισμένη πάνω στο F_q, ένα σημείο $P \in E(F_q)$ (η βάση του διακριτού λογαρίθμου) και άλλα δύο σημεία $P_a, P_b \in E(F_q)$ πολλαπλάσια του P. Δηλαδή είναι $P_a = aP$ και $P_b = bP$ για κάποιους $a, b \in Z$.

Ζητείται: Να βρεθεί το σημείο $P_{ab} = abP$

Πρόβλημα του διακριτού λογαρίθμου DLP (Discrete Logarithm Problem)

Δίνεται: Ένας πρώτος αριθμός p, ένας γεννήτορας g του Z_p^* και ένα στοιχείο $b \in Z_p^*$

Ζητείται: Να βρεθεί ακέραιος x, με $0 \leq x \leq p-2$, τέτοιος ώστε $g^x \equiv b(mod p)$.

Πρόβλημα του διακριτού λογαρίθμου στις ελλειπτικές καμπύλες ECDLP (Elliptic Curve Discrete Logarithm Problem)

Δίνεται: Μια ελλειπτική καμπύλη E, ορισμένη πάνω στο F_q, ένα σημείο $P \in E(F_q)$ (η βάση του διακριτού λογαρίθμου και ένα σημείο $Q \in E(F_q)$.

Ζητείται: Να βρεθεί, αν υπάρχει, ακέραιος k τέτοιος ώστε : $kP = Q$.

Παρατήρηση:

Το ECDLP φαίνεται να είναι δυσκολότερο από το DLP(Discrete Logarithm Problem). Μάλιστα, για το δεύτερο υπάρχει αλγόριθμος υπό εκθετικού χρόνου, ενώ για το πρώτο όχι. Έτσι, αν και δεν υπάρχει απόδειξη, που να μας εξασφαλίζει ότι το ECDLP είναι πράγματι δυσκολότερο από το DLP, με τα σημερινά δεδομένα έχουμε εξίσου καλή ασφάλεια με αρκετά μικρότερο μήκος κλειδιού όταν χρησιμοποιούμε Κρυπτοσυστήματα ελλειπτικών καμπυλών.

Ας δούμε τώρα τους αλγόριθμους Κρυπτογράφησης και Αποκρυπτογράφησης:

Κρυπτογράφηση

INPUT: Elliptic curve domain parameters (p, a, b, P, n), public key Q, plaintext m.

OUTPUT: Ciphertext (C_1, C_2).

1. Represent the message m as a point M in $E(Fp)$.

2. Select $k \in [1, n-1]$.

3. Compute $C_1 = kP$.

4. Compute $C_2 = M + kQ$.

5. Return (C_1, C_2).

Αποκρυπτογράφηση

INPUT: Domain parameters (p, a, b, P, n), private key d, ciphertext (C_1, C_2).

OUTPUT: Plaintext m.

1. Compute $M = C_2 - dC_1$, and extract m from M.

2. Return (m).

Οι διαδικασίες Κρυπτογράφησης και Αποκρυπτογράφησης των Ελλειπτικών Καμπυλών είναι ανάλογες της βασικής El Gamal Κρυπτογράφησης. Ένα μήνυμα m εκφράζεται πρώτα ως σημείο M, έπειτα κρυπτογραφείται προσθέτοντας το στο kQ όπου το k είναι ένας τυχαία επιλεγμένος ακέραιος και το Q είναι το δημόσιο κλειδί του παραλήπτη. Ο αποστολέας στέλνει τα σημεία $C_1 = kP$ και $C_2 = M + kQ$ στον παραλήπτη ο οποίος χρησιμοποιεί το δικό του ιδιωτικό κλειδί d και υπολογίζει $dC_1 = d(kP) = k(dP) = kQ$ και έτσι βρίσκει το μήνυμα $M = C_2 - kQ$. Ένας επιτιθέμενος που θέλει να βρει το M πρέπει να υπολογίσει kQ. Το να υπολογίσεις το kQ από τους παραμέτρους τομέα, Q και $C_1 = kP$, είναι ανάλογο με το Diffie-Helman πρόβλημα και υπολογιστικά δύσκολο.

2.3.3 Ψηφιακές Υπογραφές

Η κρυπτογραφία Ελλειπτικών Καμπυλών βρίσκει χρήση σε πρωτόκολλα πιστοποίησης τα οποία βασίζονται σε σχήματα ψηφιακών υπογραφών. Ένα σχήμα ψηφιακής υπογραφής χρησιμοποιεί της παραμέτρους τομέα και το ζεύγος κλειδιών για την διαδικασία της δημιουργίας ψηφιακών υπογραφών και για της επιβεβαίωση ψηφιακών υπογραφών. Και οι δύο διαδικασίες χρησιμοποιούν συναρτήσεις κατακερματισμού $H(x)$(Hash Functions).

Το πιο ευρέως χρησιμοποιούμενο σχήμα ψηφιακών υπογραφών βασισμένο στο ECDLP είναι το ECDSA (Elliptic Curve Digital Signature Algorithm) σχήμα και παρουσιάζεται παρακάτω:

ECDSA signature generation

INPUT: Domain parameters $D = (p, a, b, P, n, h)$, private key d, message m.

OUTPUT: Signature (r, s).

1. Select $k \in [1, n-1]$.

2. Compute $kP = (x_1, y_1)$ and convert x_1 to an integer.

3. Compute $r = x_1 \, mod \, n$. If $r = 0$ then go to step 1.

4. Compute $e = H(m)$.

5. Compute $s = k^{-1}(e + dr)\ mod\ n$. If $s = 0$ then go to step 1.

6. Return (r, s).

ECDSA signature verification

INPUT: Domain parameters $D = (p, a, b, P, n, h)$, public key Q, message m, signature (r, s).

OUTPUT: Acceptance or rejection of the signature.

1. Verify that r and s are integers in the interval $[1, n - 1]$. If any verification fails then return ("Reject the signature").

2. Compute $e = H(m)$.

3. Compute $w = s^{-1}\ mod\ n$.

4. Compute $u_1 = ew\ mod\ n$ and $u_2 = rw\ mod\ n$.

5. Compute $X = u_1P + u_2Q$.

6. If $X = \infty$ then return ("Reject the signature");

7. Convert the x-coordinate x_1 of X to an integer and compute $v = x_1\ mod\ n$.

8. If $v = r$ then return ("Accept the signature");
Else return ("Reject the signature").

Αν η υπογραφή (r, s) σε ένα μήνυμα m έχει όντος παραχθεί από τον νόμιμο χρήστη, τότε $s \equiv k^{-1}(e + dr)\ mod\ n$, έτσι έχουμε:

$$k \equiv s^{-1}(e + dr) \equiv s^{-1}e + s^{-1}rd \equiv we + wrd \equiv u_1 + u_2d\ (modn)$$

Οπότε $X = u_1P + u_2Q = (u_1 + u_2d)P = kP$ και $v = r$ όπως απαιτείται.

Παρατηρώντας τους παραπάνω αλγορίθμους μπορεί να διαπιστωθεί εύκολα ότι οι χρησιμοποιούμενες σε αυτούς αριθμητικές πράξεις είναι δύο. Η πράξη του πολλαπλασιασμού $modulo\ k$ και η πράξη του $scalar$ πολλαπλασιασμού ακεραίου με σημείο της Ελλειπτικής Καμπύλης. Η πράξη του πολλαπλασιασμού $modulo\ k$ έχει αμελητέα μαθηματική πολυπλοκότητα, όμως ο $scalar$ πολλαπλασιασμός έχει σαφώς μεγαλύτερη πολυπλοκότητα αφού εμπεριέχει τις πράξεις του διπλασιασμού και της άθροισης σημείων της Ελλειπτικής Καμπύλης. Για αυτό το λόγο, η βελτιστοποίηση της απόδοσης ενός κρυπτοσυστήματος Ελλειπτικών Καμπυλών εστιάζεται σε αυτή την αριθμητική πράξη και σε όλες εκείνες τις πράξεις που χρειάζονται ώστε να πραγματοποιηθεί αυτή η πράξη καλύτερα.

Σημειώσεις Ασφάλειας

1. (Απαιτήσεις Ασφάλειας για τις Hash Functions)

Αν η H δεν είναι μη αντιστρέψιμη συνάρτηση(pre-image resistant), τότε ο επιτιθέμενος E μπορεί να πλαστογραφεί υπογραφές με τον ακόλουθο τρόπο:

1. Ο E επιλέγει έναν αυθαίρετο ακέραιο l και υπολογίζει το r από την x-συντεταγμένη του $Q + lP\ (modn)$.

2. Ο E ορίζει $s = r$ και υπολογίζει $e = rl\ (modn)$. Αν ο E μπορεί να βρει μήνυμα m τέτοιο ώστε $e = H(m)$, τότε το ζεύγος (r, s) είναι μια έγκυρη υπογραφή για το m.

Αν η H δεν είναι ανθεκτική στις συγκρούσεις(collision resistant), τότε ο E πλαστογραφεί με τον ακόλουθο τρόπο:

1. Πρώτα βρίσκει 2 μηνύματα m και m' τέτοια ώστε να ισχύει $H(m) = H(m')$

2. Στην συνέχεια ζητάει από τον A να υπογράψει στο m και έτσι η υπογραφή αυτή θα είναι έγκυρη και για το m'.

2. (Αρχές για τον έλεγχο του r και s στην Επιβεβαίωση Υπογραφής)

Στο πρώτο βήμα του Αλγόριθμου Επιβεβαίωσης γίνεται ο έλεγχος ότι τα r και s είναι ακέραιοι στο διάστημα $[1, n-1]$. Αυτοί οι έλεγχοι μπορούν να γίνουν πολύ αποτελεσματικά και έχουν παρατηρηθεί επιθέσεις σε παρόμοια συστήματα υπογραφών(El Gamal) τα οποία δεν έκανα τους ελέγχους αυτούς. Στην συνέχεια ακολουθεί επίθεση στην ECDSA στην περίπτωση που δεν γίνεται ο έλεγχος αν $r \not\equiv 0\ (modn)$.

Έστω ότι ο A χρησιμοποιεί την Ελλειπτική Καμπύλη $E: y^2 = x^3 + ax + b$ στο σώμα F_p όπου p πρώτος και το b είναι τετραγωνικό υπόλοιπο $modulo\ p$ και έχει για σημείο βάσης το $P = (0, \sqrt{b})$ πρώτης τάξης n.(Είναι πιθανό όλοι οι χρήστες να έχουν επιλέξει σημείο βάσης με την x-συντεταγμένη να είναι 0 έτσι ώστε να έχουν παραμέτρους τομέα με μικρό μέγεθος.) Ο επιτιθέμενος τώρα μπορεί να πλαστογραφήσει την υπογραφή του A σε οποιοδήποτε μήνυμα m της επιλογής του υπολογίζοντας $e = H(m)$. Μπορούμε να ελέγξουμε ότι το ζεύγος $(r = 0, s = e)$ είναι μια έγκυρη υπογραφή για το m.

3. (Απαιτήσεις Ασφάλειας για μυστικά ανά-μήνυμα)

Ο μυστικός αριθμός k που επιλέγεται ανά μήνυμα στον αλγόριθμο της παραγωγής της υπογραφής ECDSA έχει τις ίδιες απαιτήσεις ασφάλειας όπως και το ιδιωτικό κλειδί d.
Αν ο επιτιθέμενος E μάθει έστω και ένα k το οποίο ο A χρησιμοποίησε για την παραγωγή

της υπογραφής (r, s) σε κάποιο μήνυμα m, τότε ο E μπορεί να βρει το ιδιωτικό κλειδί του A αφού

$$d = r^{-1}(ks - e) mod n$$

όπου $e = H(m)$. Επίσης ο Howgrave-Graham και Smart έχουν δείξει ότι αν ένας επιτιθέμενος μάθει με κάποιον τρόπο μερικά από τα bits των μυστικών k ανά-μήνυμα που αντιστοιχούν σε αρκετά υπογεγραμμένα μηνύματα, τότε ο επιτιθέμενος μπορεί εύκολα να υπολογίσει το ιδιωτικό κλειδί. Αυτές οι παρατηρήσεις δείχνουν ότι τα μυστικά ανά-μηνύματα πρέπει να παράγονται με ασφάλεια, να αποθηκεύονται και τέλος να καταστρέφονται με ασφάλεια μετά την χρήση τους.

4. (Επαναλαμβανόμενη χρήση μυστικών ανά-μήνυμα)

Τα μυστικά ανά-μήνυμα k πρέπει να παράγονται τυχαία. Πιο συγκεκριμένα, δεν πρέπει ποτέ το ίδιο μυστικό να χρησιμοποιηθεί σε άλλο μήνυμα, αλλιώς το ιδιωτικό κλειδί d μπορεί να βρεθεί.

Έστω λοιπόν ότι το ίδιο μυστικό k έχει χρησιμοποιηθεί για να παραχθούν οι υπογραφές (r, s_1) και (r, s_2) σε 2 μηνύματα m_1 και m_2. Τότε $s_1 \equiv k^{-1}(e_1 + dr)\ mod\ n$ και $s_2 \equiv k^{-1}(e_2 + dr)\ mod\ n$, όπου $e_1 = H(m_1)$ και $e_2 = H(m_2)$. Έτσι έχουμε $ks_1 \equiv (e_1 + dr)\ mod\ n$ και $ks_2 \equiv (e_2 + dr)\ mod\ n$. Αφαιρώντας τις 2 σχέσεις παίρνουμε $k(s_1 - s_2) \equiv (e_1 - e_2)\ mod\ n$ και αν $s_1 \not\equiv s_2\ (mod n)$, το οποίο συμβαίνει με μεγάλη πιθανότητα, τότε

$$k \equiv (s_1 - s_2)^{-1}\ (e_1 - e_2)\ mod\ n$$

Αφού ο επιτιθέμενος μάθει το k τότε μπορεί να βρει και το ιδιωτικό κλειδί d.

Ντετερμινιστική ECDSA

Το 2010 τα ιδιωτικά κλειδιά της ECDSA του Sony PlayStation 3 διέρρευσαν λόγο μιας κακιάς γεννήτριας τυχαίων αριθμών που χρησιμοποιήθηκε [33]. Το τυχαίο μυστικό $scalar$ που παραγόταν στην διαδικασία της υπογραφής αποδείχτηκε αργότερα ότι ήταν ένας σταθερός αριθμός. Και αργότερα κάποια ελαττώματα βρέθηκαν σε συσκευές Android όταν παραγόντουσαν τυχαίες τιμές, τα οποία οδήγησαν στο να χαθούν Bitcoins σε συσκευές που λειτουργούσαν με Bitcoin Software [65].

Αυτά τα δύο περιστατικά ασφάλειας της ECDSA ήταν λόγο κακών γεννητριών τυχαίων αριθμών. Για να αποφύγουμε τέτοιες κακές γεννήτριες, μια ντετερμινιστική ECDSA θα αύξανε την ασφάλεια της εφαρμογής της υπογραφής. Η ντετερμινιστική ECDSA προτάθηκε από τον Pornin στο RFC 6979 [57]. Ντετερμινιστική εδώ σημαίνει ότι αντί να επιλέξουμε έναν τυχαίο $scalar\ k$ στην διαδικασία της υπογραφής, το k είναι σταθερό με το ίδιο μήνυμα και ιδιωτικό κλειδί κατά την διάρκεια της παραγωγής της υπογραφής αλλά είναι δυσδιάκριτο σε σχέση με τα τυχαία παραγόμενα.

Η παραγωγή του k χρησιμοποιεί το Hash του μηνύματος $h(m)$ και το ιδιωτικό κλειδί ως είσοδο σε μια ντετερμινιστική γεννήτρια ψευδοτυχαίων αριθμών την HMAC-DRBG και η έξοδος της χρησιμοποιείται για το k. Ο αναλυτικός αλγόριθμος περιγράφεται στο [57]. Η ασφάλεια είναι σίγουρη εφόσον η HMAC συμπεριφέρεται ως ψευδοτυχαία συνάρτηση.

2.4 Βελτιώσεις Απόδοσης της ECC/ECDSA

Τώρα θα δούμε πως θα μπορούσαμε να βελτιώσουμε την απόδοση της ψηφιακής υπογραφής ECDSA, αφού ο πολλαπλασιασμός σημείου είναι η πιο χρονοβόρα διαδικασία.

Για να υπολογίσουμε τον πολλαπλασιασμό σημείου η πιο απλή μέθοδος ονομάζεται double-and-add σε δυαδική αναπαράσταση. Γενικεύσεις αυτής της μεθόδου όπως η $k-ary$ μέθοδος, η μέθοδος του παραθύρου μπορούν να χρησιμοποιηθούν για τον υπολογισμό του πολλαπλασιασμού σημείου στις Ελλειπτικές Καμπύλες σε πεπερασμένα σώματα[50]. Αυτοί οι αλγόριθμοι επεξεργάζονται ένα block από ψηφία ταυτόχρονα και χρειάζονται μια προ επεξεργασία υπολογισμών βασισμένη στο μέγεθος του block. Ο Gordon περιγράφει κάποιες γρήγορες εκθετικές μεθόδους που μπορούν να χρησιμοποιηθούν στην ECC στο [38]. Ο Itoh στο [43] προτείνει κάποιες μεθόδους για τον υπολογισμό επαναλαμβανόμενων διπλασιασμών σε προβολικές συντεταγμένες οι οποίες είναι συμβατές με την μέθοδο του παραθύρου, έτσι ώστε ο αριθμός των πολλαπλασιασμών και προσθέσεων να μειωθεί. Η Emilia Kasper στο [44] παρουσιάζει μια μέθοδο για να βελτιωθεί η απόδοση της βιβλιοθήκης Ελλειπτικών Καμπυλών στο openSSl. Ο συγγραφέας επέλεξε την καμπύλη $secp224r1$ για την χρήση της σε 64-bit εκτέλεση. Για να μειώσει το κόστος των υπολογισμών του σημείου, χρησιμοποιείται η τεχνική των προ-υπολογισμών για να υπολογίσει πολλαπλάσια ενός σημείου πριν ο $scalar$ πολλαπλασιασμός σημείου ξεκινήσει και έτσι ο τελικός αριθμός διπλασιασμών σημείου και πρόσθεσης μειώνεται.

2.4.1 Πρόσθεση και Διπλασιασμός Σημείου Ελλειπτικής Καμπύλης Σε Προβολικές Συντεταγμένες

Έστω ότι έχουμε μια ελλειπτική καμπύλη E ορισμένη πάνω σε ένα πεπερασμένο σώμα F. Κάθε σημείο της ελλειπτικής καμπύλης (EK) περιγράφεται από δύο συντεταγμένες $x, y \in F$. Σε αυτή την περίπτωση λέμε ότι το EK σημείο ανήκει στο δύο διαστάσεων συγγενές επίπεδο $A_F = \{(x, y) \in F \times F\}$. Βέβαια, υπάρχει μια αντιστοιχία μεταξύ του συγγενές επιπέδου A_F^2 και του δύο διαστάσεων προβολικό επίπεδο $P_F = \{(X:Y:Z) \in F \times F \times F\}$.

Η κλάση ισοδυναμίας στο προβολικό επίπεδο είναι $\{(X:Y:Z) = (\theta^c X, \theta^d Y, \theta^e Z) : X, Y, Z, \theta \in F\}$ παρόλο που συνήθως επιλέγεται $e = 0$ έτσι ώστε $\{(X:Y:Z) = (\theta^c X, \theta^d Y, Z) : X, Y, Z, \theta \in F$. Οι τιμές c, d είναι ακέραιοι. Αν $Z = 0$ στο προβολικό επίπεδο τότε $(X:Y:0)$ είναι η γραμμή στο άπειρο η οποία ταυτίζεται με το σημείο στο άπειρο του συγγενούς επιπέδου. Σε οποιαδήποτε άλλη περίπτωση $(Z \neq 0)$ μπορούμε να αντιστοιχίσουμε τις συντεταγμένες (x, y) του συγγενούς επιπέδου με τις συντεταγμένες του προβολικό επιπέδου ως $(X, Y, Z) = (xZ^c, yZ^d, 1)$ ή αλλιώς $x = X/Z^c$ και $y = Y/Z^d$.

Αν E είναι η εξίσωση της ελλειπτικής καμπύλης στο συγγενές επίπεδο, η ισοδύναμη εξίσωση E στο προβολικό επίπεδο μπορεί να βρεθεί αν αντικαταστήσουμε τα x, y με τις ισοδύναμες προβολικό συντεταγμένες X/Z^cκαι Y/Z^d αντιστοίχως. Ανάλογα με τις τιμές των c και d, διάφοροι τύποι προβολικών συντεταγμένων μπορούν να προσδιοριστούν. Μεταξύ αυτών οι πιο σημαντικές (και ευρέως χρησιμοποιούμενες) είναι οι τυπικές προβολικές συντεταγμένες $(c = 1 \, και \, d = 1)$, οι Jacobian προβολικές συντεταγμένες $(c = 2 \, και \, d = 1)$, οι Chudnovsky προβολικές συντεταγμένες $((X:Y:Z:Z^2:Z^3)$ αναπαράσταση), οι Lopez-Dahab

προβολικές συντεταγμένες ($c = 1 \kappa\alpha\iota\, d = 2$) και αρκετά άλλες που αποτελούν ανάμιξη συγγενών και προβολικών συντεταγμένων (mixed affine – projective συντεταγμένες).

Έστω ότι έχουμε μια ελλειπτική καμπύλη E ορισμένη πάνω σε $GF(p)$. Τότε, η μικρή εξίσωση Weierstrass της ελλειπτικής καμπύλης αυτής στο συγγενές επίπεδο έχει την μορφή $E: y^2 = x^3 + ax + b$ και αφού εφαρμόσουμε την μετατροπή σε προβολικό επίπεδο θα γίνει $E: Y^2 = X^3 Z^{2d-3c} + aXZ^{2d-c} + bZ^{2d}$ όταν 2d > 3c και $E: Y^2 Z^{3c-2d} = X^3 + aXZ^{2c} + bZ^{3c}$ όταν $2d < 3c$. Για $c = 1$ και $d = 1$ (τυπικές προβολικές συντεταγμένες) η εξίσωση της ελλειπτικής καμπύλης γίνεται

$$E: Y^2 Z = X^3 + aXZ^2 + bZ^3$$

Έστω ότι έχουμε δύο ΕΚ σημεία στο προβολικό επίπεδο $P_1 = (X_1, Y_1, Z_1)$ και $P_2 = (X_2, Y_2, Z_2)$. Τότε η πρόσθεση ΕΚ σημείων ($P_3 = (X_3, Y_3, Z_3) = P_1 + P_2$) και ο διπλασιασμός EK σημείων ($P_3 = 2P_1$) μπορεί να περιγραφεί στους παρακάτω αλγόριθμους:

Πρόσθεση Σημείων στο Προβολικό Επίπεδο

$U = U_1 - U_2 \; where \; U_1 = Y_2 Z_1, U_2 = Y_1 Z_2$

$V = V_1 - V_2 \; where \; V_1 = X_2 Z_1, V_2 = X_1 Z_2$

$W = Z_1 Z_2$

$A = U^2 W - V^3 - 2V^2 V_2$

$Then$

$X_3 = VA$

$Y_3 = U(V^2 V_2 - A) - V^3 U_2$

$Z_3 = V^3 W$

Διπλασιασμός Σημείων στο Προβολικό Επίπεδο

$W = aZ^2 + 3X^2$

$S = YZ$

$B = XYS$

$H = W^2 + 8B$

$Then$

$X_3 = 2HS$

$Y_3 = W(4B - H) - 8Y^2S^2$

$Z_3 = 8S^3$

2.4.2 Scalar Πολλαπλασιασμός Ελλειπτικών Καμπυλών

Ο $scalar$ πολλαπλασιασμός είναι μαθηματικά η πιο πολύπλοκη πράξη σε ελλειπτικές καμπύλες. Ένας $scalar$ πολλαπλασιασμός $Q = sP$, απαιτεί πολλούς διπλασιασμούς και προσθέσεις EK σημείων ο αριθμός των οποίων εξαρτάται από τον ακέραιο s. Εκφράζοντας το s σε δυαδική μορφή, η εξάρτηση αυτή καταδεικνύεται ακόμα πιο πολύ. Ο αριθμός τον μηδενικών και μη μηδενικών bits του s, η θέση τους μέσα στο δυαδικό διάνυσμα s και το μέγεθος bit του s μπορούν να οδηγήσουν κάθε φορά σε διαφορετικό αριθμό προσθέσεων και διπλασιασμών EK σημείων στον $scalar$ πολλαπλασιασμό. Αυτό φαίνεται χαρακτηριστικά, στη δυαδική έκδοση του αλγορίθμου $scalar$ πολλαπλασιασμού.

Δυαδικός scalar πολλαπλασιασμός (Double-and-Add Method)

INPUT: $s = (s_{t-1}, \ldots, s_1, s_0)_2, P \in E(F)$.

OUTPUT: $Q = sP$

1. $Q = \partial$

2. $For\ i\ from\ 0\ to\ t - 1\ do$

   2.1 $If\ s_i = 1\ then\ Q = Q + P.$

   2.2 $P = 2P.$

3. $Return\ Q.$

Το Hamming Weight του s $(HW(s))$ καθορίζει τον αριθμό των πράξεων πρόσθεσης EK σημείων και το μέγεθος bit του s καθορίζει τον αριθμό των πράξεων διπλασιασμού EK σημείων καθώς και τον αριθμό των γύρων του αλγορίθμου Double-and-Add. Συνεπώς, στον δυαδικό αλγόριθμο $scalar$ πολλαπλασιασμού υπάρχουν $HW(s)$ προσθέσεις EK σημείων (PA) και t διπλασιασμοί σημείων (PD). Υποθέτοντας ότι το μέγεθος bit του s είναι κοντά στην τάξη k του πεπερασμένου σώματος και ότι $HW(s) = k/2$ κατά μέσο όρο (ο αριθμός των μηδενικών bit του s είναι ίδιος με τον μη μηδενικών bits του s), ο συνολικός αριθμός των πράξεων ελλειπτικών καμπυλών για $Q = sP$ θα είναι $k/2\ PA + k\ PD$.

Μια πιο γενική προσέγγιση στον $scalar$ πολλαπλασιασμό που σχετίζεται με την μορφή του ακεραίου s είναι η επεξεργασία w bits του s ανά χρονική στιγμή. Μια τέτοια μέθοδος ονομάζεται μέθοδος παραθύρου και απαντάται σε διάφορες μορφές [41] όπως η μέθοδος σταθερού παραθύρου ή μεταβαλλόμενου παραθύρου.

Η βασική ιδέα των μεθόδων παραθύρου για τον scalar πολλαπλασιασμό σχετίζεται με την χρήση w bits του πολλαπλασιαστή s σε κάθε κύκλο ρολογιού με στόχο την μείωση του συνολικού αριθμού πράξεων πρόσθεσης και διπλασιασμού EK σημείων. Αυτά τα w bits ονομάζονται παράθυρο του s. Οι μέθοδοι παραθύρου χρησιμοποιούν μια διαδικασία προ επεξεργασίας για τον υπολογισμό του $P_j = jP$ όπου j παίρνει περιττές τιμές από 1 σε $2^{w-1} - 1$. Αυτές οι τιμές αποθηκεύονται έτσι ώστε να χρησιμοποιηθούν στην κυρίως διαδικασία του πολλαπλασιασμού η οποία μπορεί να έχει διάφορες μορφές ανάλογα με το είδος της μεθόδου παραθύρου που εφαρμόζεται. Στην μέθοδο σταθερού παραθύρου, ο πολλαπλασιαστής s χωρίζεται σε σταθερού w bit μήκους παράθυρα με αυστηρά καθορισμένο bit αρχής και τέλους. Στη μέθοδο μεταβλητού παραθύρου χρησιμοποιείται παράθυρο με μήκος w bit το πολύ και μεταβλητό bit αρχής και τέλους. Αυτό το μεταβλητό παράθυρο «κυλά» από δεξιά προς αριστερά στο δυαδικό διάνυσμα του s, παραλείποντας συνεχόμενα μηδενικά s_i bits μετά από την επεξεργασία ενός μη μηδενικό s_i bit. Η μέθοδος μεταβλητού παραθύρου είναι ταχύτερη από αυτήν σταθερού παραθύρου ξεπερνώντας προβλήματα που παρουσιάζει η δεύτερη και παρουσιάζεται στον παρακάτω αλγόριθμο.

Μέθοδος Μεταβλητού Παραθύρου για Scalar Πολλαπλασιασμό (Window Method)

Είσοδος: $s = \sum_{i=0}^{t-1} s_i 2^i : (s_{t-1}, \ldots, s_1, s_0)_2, P \in E(F)$.

Έξοδος: $Q = sP$

Φάση Προ επεξεργασίας (υπολογισμός του $P_e = eP$, όπου $e \leq 2^w - 1$ είναι ένας περιττός ακέραιος)

1. $P_1 = P,\ Q = \partial,\ r = t - 1.$

2. $For\ j = 1\ to\ 2^{w-1} - 1$

   2.1 $P_{2j+1} = P_{2j-1} + 2P$

Φάση Κυρίως Υπολογισμών

1. $While\ r \geq 0\ do$

   1.1 $If\ s_r = 0\ then$

      1.1.2 $Q = 2Q$

      1.1.2 $r = r - 1$

Else

1.1.3 $v = w$

1.1.4 *While* $s_{r-v+1} = 0$ *do* $v = v - 1$ (ο μεγαλύτερος ακέραιος v για περιττό u)

1.1.5 $u = \{s_r, s_{r-1} \ldots s_{r-v+1}\}$ (το u είναι περιττός ακέραιος)

1.1.6 $Q = 2^v Q + P_u$

1.1.7 $r = r - v$

2. *Return Q*

2.5 Side Channel Επιθέσεις σε ECC/ECDSA

Οι επιθέσεις side channel (SCA) θεωρούνται αρκετά αποδοτικές κρυπταναλυτικές μεθόδους και αποτελούν παράμετρο σχεδιασμού ενός κρυπτογραφήματος μιας και εκμεταλλεύονται ιδιότητες του υλικού πάνω στο οποίο είναι σχεδιασμένο και υλοποιημένο ένα τέτοιο σύστημα. Πιο συγκεκριμένα, οι επιθέσεις αυτές εκμεταλλεύονται πληροφορία που μπορεί να «διαφύγει» από την υλοποίηση ενός κρυπτογραφικού αλγορίθμου κατά την διάρκεια της εκτέλεσης του αλγορίθμου αυτού. Τέτοιες πληροφορίες μπορεί να σχετίζονται με τον χρόνο υπολογισμού, με την εκπεμπόμενη ηλεκτρομαγνητική ακτινοβολία, με τα μηνύματα λάθους ή με ίχνη της κατανάλωση ισχύος ενός συστήματος που πραγματοποιεί κάποια κρυπτογραφική πράξη. Οι πληροφορίες αυτές χρησιμοποιούνται έτσι ώστε να χαρακτηρίσουν συγκεκριμένους υπολογισμούς που πραγματοποιούνται σε δεδομένο χρονικό διάστημα της εκτέλεσης της κρυπτογραφικής διεργασίας με απώτερο σκοπό την εύρεση του ιδιωτικού κλειδιού.

SCA κρυπταναλυτικές μέθοδοι έχουν προταθεί και για την κρυπτογραφία ελλειπτικών καμπυλών [26], συμπεριλαμβανομένων μοντέλων επιθέσεων χρονισμού (timing attacks), απλής ανάλυσης ισχύος (Simple Power Analysis, SPA) και διαφορικής ανάλυσης ισχύος (Differential Power Analysis, DPA) [45],[46] που θεωρούνται σημαντικές κρυπτογραφικές απειλές. Σε αυτές τις επιθέσεις, οι πράξεις πρόσθεσης και διπλασιασμού ΕΚ σημείων αναγνωρίζονται κατά την διάρκεια ενός $scalar$ πολλαπλασιασμού ανιχνεύοντας την κατανάλωση ισχύος (SPA, DPA) ή τον απαραίτητο χρόνο υπολογισμού αποτελέσματος από το σύστημα (timing attacks) και στη συνέχεια αναλύοντας στατιστικά μια συγκεκριμένη τιμή στην διεργασία $scalar$ πολλαπλασιασμού. Αυτές οι πληροφορίες μαζί με την γνώση του χρησιμοποιούμενου αλγόριθμου $scalar$ πολλαπλασιασμού (ο οποίος δεν θεωρείται κρυφός) είναι αρκετές για τον προσδιορισμό του ακεραίου s και κατά συνέπεια της εύκολης επίλυσης του ECDLP προβλήματος εφόσον ο κρυπτογραφικός αλγόριθμος το επιτρέπει.

Πρέπει να σημειωθεί, βέβαια, ότι οι παραπάνω SCA δεν είναι εφαρμόσιμες σε κάθε δυνατό κρυπτογραφικό σύστημα αφού απαιτείται να υπάρχει κάποιος εύκολος τρόπος πραγματοποίησης μετρήσεων χρόνου και/ή κατανάλωσης ισχύος. Όταν μια συσκευή λειτουργεί σε πιθανό εχθρικό, ανασφαλές, περιβάλλον, τα SCA μπορεί να αποτελέσουν σοβαρή απειλή.

Εκτός από τις παθητικές SCA, υπάρχουν και ενεργές SCA και ένα παράδειγμα είναι η ανάλυση λαθών(Fault Analysis). Η Fault Analysis (FA) βασίζεται στην παραγωγή λαθών στο σύστημα με στόχο να εξάγει κλειδιά. Λάθη μπορούν να δημιουργηθούν αλλάζοντας την ένταση του ρεύματος, πειράζοντας το ρολόι και πολλά άλλα. Για να αναγνωρίσεις τέτοια λάθη και να φέρεις εις πέρας μια επίθεση, το ίδιο μήνυμα κρυπτογραφείτε δύο φορές και τα αποτελέσματα συγκρίνονται.

Παρόλο που υπάρχουν πολλοί τρόποι να διεξαχθούν τέτοιου είδους επιθέσεις, έχουν προταθεί κάποια βασικά μέτρα αντιμετώπισης τους. Τα μέτρα αυτά μπορούμε να τα χωρίσουμε σε δύο κατηγορίες. Στην μία είναι η αποφυγή ή η μείωση της πληροφορίας που διαφεύγει και στην άλλη είναι να αποσυνδέσουμε τελείως την σχέση μεταξύ πληροφορίας και μυστικών δεδομένων.

Για της timing attacks, η πρόσθεση καθυστερήσεων στις λειτουργίες του συστήματος φαίνεται να είναι η πιο απλή πρόταση αλλά η υλοποίηση της έχει δυσκολίες. Αν όλες οι λειτουργίες του συστήματος χρειαζόντουσαν τον ίδιο χρόνο τότε θα μειωνόταν η αποδοτικότητα του συστήματος. Η πρόσθεση τυχαίων καθυστερήσεων στις λειτουργίες θα μείωνε τις πιθανότητες να σπάσει το σύστημα αλλά θα ήταν πάλι εφικτό συλλέγοντας περισσότερα δεδομένα.

Όσο αφορά της επιθέσεις κατανάλωσης ισχύος, μια μέθοδος για να αποτρέψεις SPA και DPA είναι η ισορροπία της κατανάλωσης ισχύος. Η μέθοδος αυτή απαιτεί την εγκατάσταση ψεύτικων μετρητών(Dummy Registers) έτσι ώστε η κατανάλωση ισχύος να ισορροπείτε. Η μέθοδος αλλάζει την κατανάλωση σε μια σταθερά και αφαιρεί την εξάρτηση των δεδομένων μεταξύ εισόδων και ιδιωτικών κλειδιών. Άλλη μέθοδος είναι η πρόσθεση θορύβου στις μετρήσεις της κατανάλωσης ισχύος. Παρόμοια με την πρόσθεση καθυστερήσεων στις timing attacks, η πρόσθεση θορύβου απλά θα μεγαλώσει τον αριθμό των δειγμάτων που χρειάζονται για να σπάσει το σύστημα.

Η μέθοδος Montgomery παρουσιάστηκε για πρώτη φορά το 1987 [53] και λέγεται πως είναι ανθεκτική στις timing attacks και στην simple power analysis [42, 55], αλλά ο Brumley και Tuveri πρότειναν μια απομακρυσμένη timing attack στην υλοποίηση της μεθόδου Montgomery ladder στο OpenSSl για ECDSA χρησιμοποιώντας Ελλειπτικές Καμπύλες σε δυαδικά σώματα. Ο Luther Martin δεν θεώρησε πως ήταν σοβαρή επίθεση αφού είναι μια σπάνια περίπτωση που δουλεύει μόνο σε Ψηφιακές Υπογραφές ECDSA που χρησιμοποιούνται στο OpenSSl' TLS πάνω σε δυαδικά σώματα[48]. Όμως, μετά την πρόταση της επίθεσης FLUSH + RELOAD, ο Yarom και Falkner ανάκτησαν το μυστικό $scalar\ s$ στην ECDSA στο OpenSSl με την υλοποίηση της μεθόδου Montgomery, εφαρμόζοντας αυτήν την επίθεση. Οι συγγραφείς πρότειναν να αποφεύγεται η μέθοδος Montgomery ladder στην υλοποίηση των πρωτοκόλλων Ελλειπτικών Καμπυλών στο OpenSSl αφού δεν είναι πλέον ασφαλής.

Ένας άλλος τρόπος προστασίας εναντίων SCA παρόμοιος με την μέθοδο Montgomery που ονομάζεται double-and-add-always προτείνεται από τον Coron το 1999 [26]. Ο Coron πρότεινε 3 μέτρα ενάντια σε Differential Power Attacks, τυχαιοποίηση του μυστικού $scalar$ στον πολλαπλασιασμό, τύφλωση του σημείου P και τυχαίες προβολικές συντεταγμένες. Αλλά όπως φαίνεται στο [34], η μέθοδος της τυχαιοποίησης του μυστικού $scalar$ είναι ευάλωτη στις carry-based επιθέσεις και στο [35] βλέπουμε ότι ούτε η τυχαιοποίηση του μυστικού $scalar$ ούτε η τύφλωση σημείου είναι ανθεκτικές στην Doubling Attack. Αλλά ευτυχώς μια μέθοδος που ονομάζεται random key splitting [21] κάνει τις Ελλειπτικές Καμπύλες ανθεκτικές στην Doubling Attack χωρίζοντας το μυστικό $scalar$ σε 2 τυχαίες τιμές.

Όσο αφορά την ανάλυση λαθών(Fault Analysis), το coherence check που περιγράφει ο Dominguez Oviedo A. [56] μπορεί να χρησιμοποιηθεί στην μέθοδο Montgomery Ladder για να αντέξει επιθέσεις διαφορικής ανάλυσης λαθών (Differential Fault Analysis). Επίσης η μέθοδος Montgomery Ladder είναι ανθεκτική στην C safe-error Attack η οποία είναι ένα είδος FA. Η μέθοδος combined curve check που προτείνεται από τον Blomer J. [13] μπορεί να αντιμετωπίσει τις Sign Change Fault Attacks εισάγοντας μια σχετική καμπύλη ανίχνευσης λαθών. Ο Ciet και Joye [22] παρουσιάζουν δύο μεθόδους, την point validation και την curve integrity check. Η point validation ελέγχει αν ένα συγκεκριμένο σημείο είναι στην καμπύλη προστατεύοντας έτσι το σύστημα από invalid points attacks. Ενώ η μέθοδος curve integrity check ανιχνεύει λάθη που μπορεί να έχουν εισαχθεί στις παραμέτρους της καμπύλης όταν ο πολλαπλασιασμός σημείου υπολογίζεται.

Τέλος ο Benger [9] πρότεινε μια επίθεση για να εξάγεις το ιδιωτικό κλειδί από τον αλγόριθμο της ECDSA στο OpenSSl. Η ανάλυση είναι βασισμένη στην καμπύλη $secp256k1$ που χρησιμοποιείται στο Bitcoin. Ο συγγραφέας συνδύασε την FLUSH+RELOAD attack και κάποιες lattice τεχνικές. Μπόρεσαν να εξάγουν ιδιωτικά κλειδιά από Ελλειπτικές Καμπύλες με μεγάλη πιθανότητα χρησιμοποιώντας λιγότερες από 256 υπογραφές.

Η επίθεση αυτή απαιτεί την εξασφάλιση μερικών υπογραφών για ένα καθορισμένο ιδιωτικό κλειδί, οπότε θα μπορούσαμε να μειώσουμε τον αριθμό που ένα ιδιωτικό κλειδί χρησιμοποιείται. Το bitcoin σύστημα χρησιμοποιεί επίσης την καμπύλη $secp256k1$ και προτείνει ένα καινούργιο κλειδί να χρησιμοποιείται σε κάθε συναλλαγή. Όμως η πρόταση αυτή δεν ακολουθείται πάντα [58].

2.5.1 Απλή Ανάλυση Ισχύος (Simple Power Analysis)

Για την SPA, υπάρχουν 3 μέθοδοι κατάλληλοι για υλοποίηση. Η double-and-add-always, η Montgomery Ladder και η Unified Operation.

Double-and-add-always

Η μέθοδος αυτή είναι μια βελτίωση της μεθόδου Double-and-add η οποία προσθέτει σε κάθε γύρο ψεύτικες(dummy) προσθέσεις σημείων. Για κάθε bit του μυστικού $scalar$, οι λειτουργίες της πρόσθεσης και του διπλασιασμού εκτελούνται έτσι ώστε οι υπολογιστικές διεργασίες να είναι ανεξάρτητες από την τιμή του μυστικού $scalar$.

Double-and-add-always Algorithm

$INPUT: d, P$

$OUTPUT: Q = dP$

1: $Q[0] = \partial$

2: $Q[1] = \partial$

3: $if\ d_1 = 1\ then$

    4: $Q[0] := P$

5: $end\ if$

6: $for\ i = 1\ to\ n\ do$

    7: $Q[0] := 2Q[0]$

8: $Q[1] := Q[0] + P$

9: $Q[0] := 2Q[d_i]$

10: $end\ for$

11: $return\ Q[0]$

Montgomery Ladder

Η μέθοδος αυτή κάνει τον πολλαπλασιασμό σημείου εφαρμόζοντας την double-and-add μέθοδο, αλλά υπολογίζει σταθερά τον ίδιο αριθμό διπλασιασμού και πρόσθεσης σημείων σε μια σταθερή πατέντα ανεξάρτητα από το μυστικό $scalar$ και χωρίς ψεύτικους(dummy) υπολογισμούς.

Montgomery Ladder Algorithm

$INPUT: d, P$

$OUTPUT: Q = dP$

1: $Q[0] = \partial$

2: $Q[1] = \partial$

3: $if\ d_1 = 1\ then$

4: $Q[0] := P$

5: $Q[1] := 2P$

6: $end\ if$

7: $for\ i = 1\ to\ n\ do$

8: $Q[1 - d_i] := Q[0] + Q[1]$

9: $Q[d_i] := 2Q[d_i]$

10: $end\ for$

11: $return\ Q[0]$

Ο παραπάνω αλγόριθμος παρουσιάζει κάποια ενδιαφέροντα πλεονεκτήματα. Η διαφορά $P - Q$ σε κάθε αλγοριθμική επανάληψη έχει γνωστή τιμή και ισοδυναμεί με το αρχικό P σημείο. Ο P. Montgomery παρατήρησε ακόμα, ότι η x συντεταγμένη του αθροίσματος δύο σημείων με σταθερή διαφορά, μπορεί να υπολογιστεί χρησιμοποιώντας μόνο τις x συντεταγμένες αυτών των δύο εμπλεκόμενων σημείων.

Unified Operation

Η μέθοδος αυτή ενώνει τις πράξεις της πρόσθεσης και του διπλασιασμού σε μία λειτουργεία έτσι ώστε να εξαλείψει τις διαφορές μεταξύ των δύο πράξεων. Δοσμένων δύο σημείων $P_1 = (x_1, y_1)$ και $P_2 = (x_2, y_2)$ και είτε $P_1 = P_2$ είτε $P_1 \neq P_2$ η πράξη της πρόσθεσης και του διπλασιασμού μπορούν να υπολογιστούν.

Unified Operation Algorithm

$INPUT: P_1 = (x_1, y_1), P_2 = (x_2, y_2)$

$OUTPUT: P_3 = (x_3, y_3)$

$\mu = y_1 - \lambda x_1$

$\lambda = \dfrac{x_1^2 + x_1 x_2 + x_2^2 + a}{y_1 + y_2}$ $\quad \text{όπου } y_1 + y_2 \neq 0$

$x_3 = \lambda^2 - x_1 - x_2$

$y_3 = -\lambda x_3 - \mu$

2.5.2 Διαφορική Ανάλυση Ισχύος (Differential Power Analysis)

Για να αποφύγουμε τεχνικές που αναλύουν στατιστικά το μυστικό scalar, έχουν προταθεί διάφορα μέτρα για την αντιμετώπιση τους, όπως το Scalar Blinding, Base Point Blinding και το Random Scalar Splitting.

Scalar Blinding

Η μέθοδος αυτή επιλέγει έναν τυχαίο αριθμό k με n bits ενώ υπολογίζει το $Q = dP$, στην συνέχεια υπολογίζει $d' = d + k(\#N)$ όπου N είναι ο αριθμός των σημείων που έχει η καμπύλη και τέλος $Q = d'P = (d + k(\#N))P = dP$ $\alpha\varphi o\acute{u}$ $\#NP = \partial$.

Base Point Blinding

Η μέθοδος αυτή είναι παρόμοια με την Scalar Blinding και τυφλώνει το σημείο της βάσης P υπολογίζοντας $Q' = d(P + R)$ ενώ η τιμή της dR αποθηκεύεται μυστικά.

Random Scalar Splitting

Η μέθοδος αυτή διαχωρίζει το μυστικό scalar σε δύο τυχαίες τιμές k_1 και k_2 όπου $k = k_1 + k_2$ έτσι ώστε σε κάθε εκτέλεση το μυστικό $scalar$ είναι τυχαίο. Μια άλλη διαχώριση είναι να επιλέξουμε ένα τυχαίο r, έτσι ώστε $k = \lfloor k/r \rfloor + (k \ mod \ r)$.

2.5.3 Ανάλυση Λαθών (Fault Analysis)

Στην περίπτωση της Ανάλυσης Λαθών η μέθοδος Double-And-Add-Always είναι ευάλωτη στο C safe-error που προτάθηκε από τους Yen και Kim [63] γιατί όταν επιφέρουμε λάθη στην λειτουργία $Q[1] = Q[0] + P$ του αλγορίθμου δεν έχει διαφορά αν το b_i bit του μυστικού $scalar$ είναι 0. Με τέτοιο τρόπο το μυστικό $scalar$ μπορεί να διαρρεύσει. Όμως, η μέθοδος Montgomery Ladder είναι ανθεκτική σε αυτήν την επίθεση γιατί δεν περιλαμβάνει ψεύτικες(dummy) λειτουργίες και έτσι ότι λάθη και να επιφέρει η επίθεση θα προξενήσουν λανθασμένο αποτέλεσμα.

Η παρακάτω μέθοδος είναι ανθεκτική στην διαφορική ανάλυση λαθών (Differential Fault Analysis).

Coherence Check Method

Η μέθοδος αυτή όπως φαίνεται στο [56] είναι αποτελεσματική μόνο όταν χρησιμοποιείται η Montgomery Ladder. Αυτό συμβαίνει γιατί η διαφορά μεταξύ $Q[0]$ και $Q[1]$ είναι πάντα P στην μέθοδο Montgomery. Αν η διαφορά της τιμής ελέγχεται κατά την διάρκεια και μετά του πολλαπλασιασμού σημείου, οποιαδήποτε λάθη έχουν προκληθεί θα μπορούν να εντοπιστούν.

2.5.4 Περιπτώσεις Υλοποίησης της ECC/ECDSA

Τώρα θα δούμε 3 περιπτώσεις υλοποίησης της ECDSA οι οποίες χρησιμοποιούν τις μεθόδους που έχουν προαναφερθεί. Σε όλες τις περιπτώσεις χρησιμοποιούμε Ελλειπτικές Καμπύλες με προβολικές συντεταγμένες.

Περίπτωση 1

Η πρώτη περίπτωση εφαρμόζει τις δύο μεθόδους του Coron, την Double-And-Add-Always μαζί με την Scalar Blinding. Η Double-And-Add-Always μέθοδος αντικαθιστά την Window μέθοδο σε προβολικές συντεταγμένες. Η υλοποίηση εφαρμόζει τον Double-and-add-always Algorithm. Επίσης αντί να εκτελέσουμε τον πολλαπλασιασμό σημείου κατευθείαν στο μυστικό $scalar$, επιλέγουμε τυχαία μια τιμή k για να τυφλώσουμε το $scalar$ και εφαρμόζουμε την μέθοδο Scalar Blinding.

Στην περίπτωση αυτή ο χρόνος για να υπογράψουμε μεγαλώνει λόγο των παραπάνω πράξεων που χρησιμοποιεί η Double-And-Add-Always και λόγο της παραγωγής του τυχαίου αριθμού της μεθόδου Scalar Blinding. Όσο αφορά την ασφάλεια, η Double-And-Add-Always μπορεί να μας προστατέψει ενάντια σε απλές Side Channel επιθέσεις, αλλά αποτυγχάνει ενάντια στην Doubling Attack [35] και στην C safe-error. Ενώ η Scalar Blinding είναι και αυτή ευάλωτη στις Carry-based και Doubling επιθέσεις.

Περίπτωση 2

Η δεύτερη περίπτωση εφαρμόζει τις μεθόδους Double-And-Add-Always σε συνδυασμό με την Window έτσι ώστε να είναι δυνατό να εκτελέσουμε ψεύτικες(dummy) λειτουργίες πρόσθεσης. Όπως και στην προηγούμενη περίπτωση, έτσι και εδώ θα τυφλώσουμε το μυστικό $scalar$ με την μέθοδο Scalar Blinding.

Η περίπτωση αυτή είναι η πιο αποδοτική και είναι ανθεκτική σε SPA και DPA. Στην μέθοδο του Window, ενώ τα δυαδικά $scalar$ bits έχουν δύο μόνο τιμές, εδώ έχουμε 16 επιλογές στην περίπτωση που το παράθυρο είναι μεγέθους 4 γιατί 4 bits αναπαριστούν μία τιμή. Για να κάνουμε τις πράξεις της τιμής 0 μη διακριτές από τις άλλες 15, ο συνδυασμός των μεθόδων Double-And-Add-Always και Window χρησιμοποιείται για να σιγουρέψει ότι η επιπλέον πράξη της πρόσθεσης εκτελείτε όταν η τιμή είναι 0.

Περίπτωση 3

Η τρίτη περίπτωση υλοποίησης συνδυάζει την μέθοδο Montgomery Ladder και την μέθοδο Random Scalar Splitting. Η Montgomery δεν χρησιμοποιεί ψεύτικες(dummy) πράξεις αλλά για κάθε bit του μυστικού $scalar$, πρότυπες σταθερές πράξεις εφαρμόζονται. Όσο αφορά το Random Scalar Splitting, επιλέγουμε τυχαία τον αριθμό kk για να εκτελέσουμε τον Scalar πολλαπλασιασμό, πρώτα έχουμε $kk \cdot P$, έπειτα υπολογίζουμε $(s - kk) \cdot P$ ετσι ώστε στο τέλος να έχουμε $Q = kk \cdot P + (s - kk) \cdot P = s \cdot P$.

Ο συνδυασμός των δύο αυτών μεθόδων μας κοστίζει αρκετά σε χρόνο αλλά από την άποψη της ασφάλειας είναι η πιο ανθεκτική από τις υπόλοιπες. Ο διαχωρισμός του μυστικού $scalar$ διπλασιάζει τον χρόνο στον πολλαπλασιασμό σημείου, αλλά είναι ασφαλείς ενάντια σε DPA και στην Doubling Attack. Ενώ η Montgomery Ladder μας κάνει την υλοποίηση ανθεκτική σε SPA αφού έχει την ίδια συμπεριφορά σε κάθε bit του $scalar$ και δεν είναι ευάλωτη στην επίθεση C safe-error η οποία είναι ένα είδος Fault Analysis.

3 Bitcoin

3.1 Επισκόπηση του Bitcoin

Το Bitcoin είναι ένα αποκεντρωμένο ψηφιακό νόμισμα το οποίο επιτρέπει άμεσες πληρωμές σε οποιονδήποτε χρήστη σε όλο τον κόσμο. Το Bitcoin χρησιμοποιεί peer-to-peer τεχνολογία για να λειτουργεί χωρίς κάποια κεντρική αρχή, η διαχείριση των συναλλαγών και η έκδοση των χρημάτων διενεργείται συλλογικά από το ίδιο το δίκτυο.

Το λογισμικό του Bitcoin δημοσιεύτηκε από τον Satoshi Nakamoto το 2008[54] με άδεια από το MIT και έγινε πλήρως λειτουργικό το 2009. Το Bitcoin είναι η πρώτη επιτυχημένη υλοποίηση στην οποία διανέμεται ένα κρυπτογραφικό νόμισμα. Βασισμένοι στο γεγονός ότι τα χρήματα είναι οποιοδήποτε αντικείμενο ή κάποια καταγραφή, τα οποία χρησιμοποιούνται για πληρωμές αγαθών και υπηρεσιών και αποπληρωμή των χρεών σε μια χώρα ή σε ένα κοινωνικό-οικονομικό πλαίσιο, το Bitcoin είναι σχεδιασμένο γύρω από την ιδέα ότι γίνεται χρήση της κρυπτογραφίας για τον έλεγχο της δημιουργίας και της μεταφοράς των χρημάτων, αντί να βασιζόμαστε σε κεντρικές αρχές που μπορεί να μην εμπιστευόμαστε.

Το Bitcoin έχει αρκετές καλές ιδιότητες [7, 25] και έτσι έχει γίνει το μεγαλύτερο κρυπτο-νόμισμα παγκοσμίως:

- **Μερικώς ανώνυμο**: παρόλο που όλες οι συναλλαγές είναι δημόσια γνωστές, οι χρήστες μπορούν να χρησιμοποιούν τις διευθύνσεις των Bitcoin ως ψευδώνυμα. Επίσης για έναν χρήστη σε διαφορετική συναλλαγή, διαφορετική διεύθυνση Bitcoin πρέπει να χρησιμοποιηθεί. Με αυτόν τον τρόπο διατηρείται η ανωνυμία σε κάποιο βαθμό.

- **Μεταβιβάσιμο**: οι συναλλαγές δεν ελέγχονται από κάποια κεντρική αρχή, έτσι δημοσιεύονται όλες στο κοινό για να επικυρωθούν. Αφού δεν υπάρχει κάποιο οικονομικό δίκτυο, το Bitcoin είναι μεταβιβάσιμο.

- **Πληρωμές μη αναστρέψιμες**: από την στιγμή που ο ιδιοκτήτης ξοδέψει τα Bitcoins του δεν θα μπορεί να τα ζητήσει πίσω χωρίς την συγκατάθεση του αποδέκτη των Bitcoins.

- **Φτηνό και γρήγορο**: αντίθετα με τις τραπεζικές συναλλαγές, οι Bitcoin συναλλαγές δεν ζητάνε αμοιβή εκτός και αν προκύπτει από τον πληρωτή. Επίσης, μια συναλλαγή στο Bitcoin θα επικυρωθεί μέσα σε λίγα λεπτά, το οποίο είναι πιο γρήγορο από τις μεταφορές χρημάτων μέσω τράπεζας.

- **Ασφαλές**: στα συστήματα ψηφιακών χρημάτων η διπλή χρήση του ίδιου νομίσματος είναι μια συνηθισμένη επίθεση. Όμως, στο Bitcoin αυτό μπορεί να αποφευχθεί επικυρώνοντας δημόσια την συναλλαγή. Επιπροσθέτως, το Bitcoin μπορεί να αποτρέψει αυτήν την επίθεση αν η Hash function και ο αλγόριθμος

ψηφιακής υπογραφής είναι ασφαλής. Συγκεκριμένα, στο Bitcoin οι συναλλαγές πιστοποιούνται με κρυπτογραφική απόδειξη, μέσω της υπολογιστικής ισχύς των χρηστών του δικτύου, οι οποίοι αναφέρονται με τον όρο miners.

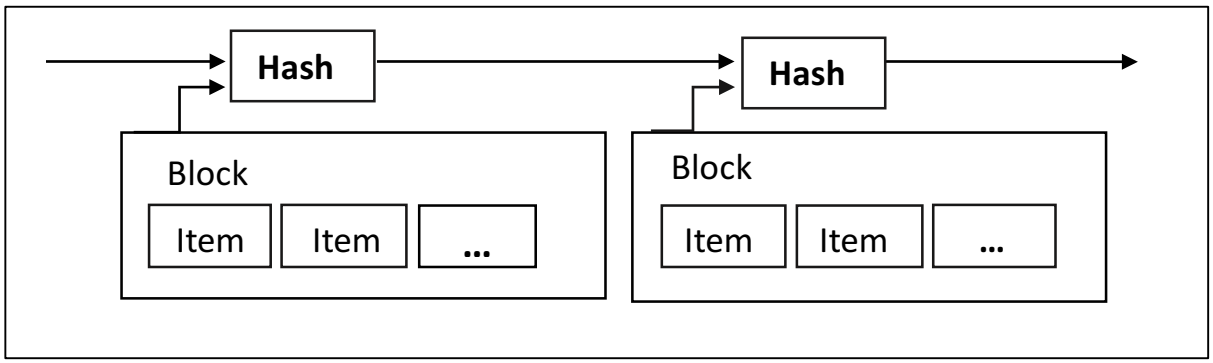
- **Όχι πληθωριστικό**: αντίθετα με τα κανονικά νομίσματα, υπάρχει ένας περιορισμός στον αριθμό των συνολικών Bitcoin που μπορούν να δημιουργηθούν. Σύμφωνα με την αρχική εκτίμηση περίπου 21 εκατομμύρια είναι ο αριθμός αυτός. Οπότε ο πληθωρισμός δεν είναι πρόβλημα για τον κόσμο των Bitcoins.

3.1.1 Αρχιτεκτονική του Bitcoin

Time stamping

Το Time stamping είναι η χρήση μιας ηλεκτρονικής σφραγίδας χρόνου που αποδεικνύει την χρονική αλληλουχία των γεγονότων. Προτάθηκε από τους Haber και Stornetta το 1991 [40] και πιστοποιούσε πότε ένα ηλεκτρονικό έγγραφο δημιουργείται ή τροποποιείται.

Αφού οι Bitcoin συναλλαγές είναι δημόσια γνωστές, το σύστημα του Time stamping χρειάζεται έτσι ώστε οι συμμετέχοντες να συμφωνούν ως προς την σειρά των συναλλαγών. Η βασική αρχή του timestamp είναι ότι η πληροφορία που αποθηκεύεται στα blocks του Bitcoin είναι διατεταγμένη σε μια αλυσίδα και το hash του block διάφορων αντικειμένων σφραγίζεται χρονικά ενώ κάθε χρονική σφραγίδα(time stamp) συμπεριλαμβάνει προηγούμενες χρονικές σφραγίδες στην hash τιμή της. Το timestamp αποδεικνύει ότι τα δεδομένα υπήρχαν εκείνη την στιγμή.

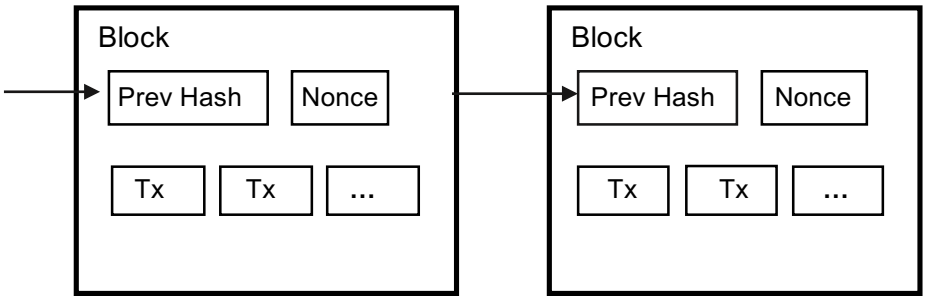


Εικόνα 3.1: Timestamp block στο Bitcoin

Proof of work

Για να επικυρώσουμε μια συναλλαγή στο Bitcoin, η μέθοδος proof-of-work χρησιμοποιείται η οποία έχει προταθεί από τον Adam Back's Hashcash [6]. Η proof-of-work ψάχνει για το hash μιας τιμής το οποίο ξεκινάει με μηδενικά bits. Ο Hash αλγόριθμος που χρησιμοποιείται είναι ο SHA-256. Ο μέσος όρος της δουλειάς που χρειάζεται είναι εκθετικός σε σχέση με τον αριθμό των μηδενικών bits που απαιτούνται και μπορεί να επαληθευτεί εκτελώντας ένα μόνο Hash.

Για το timestamp δίκτυο, εφαρμόζουμε την proof-of-work προσαυξάνοντας μια nonce τιμή σε ένα block μέχρι να βρεθεί η τιμή που μας δίνει στο hash του block τα απαραίτητα μηδενικά bits. Όταν στην συνέχεια η proof-of-work ικανοποιείται, το block αυτό δεν μπορεί να αλλάξει εκτός και αν ξαναγίνει όλη η υπολογιστική δουλειά από την αρχή. Αφού τα επόμενα blocks είναι συνδεδεμένα μεταξύ τους σαν αλυσίδα, η δουλειά που χρειάζεται για να αλλάξουμε ένα block συμπεριλαμβάνει και την αλλαγή όλων των επομένων block επίσης.

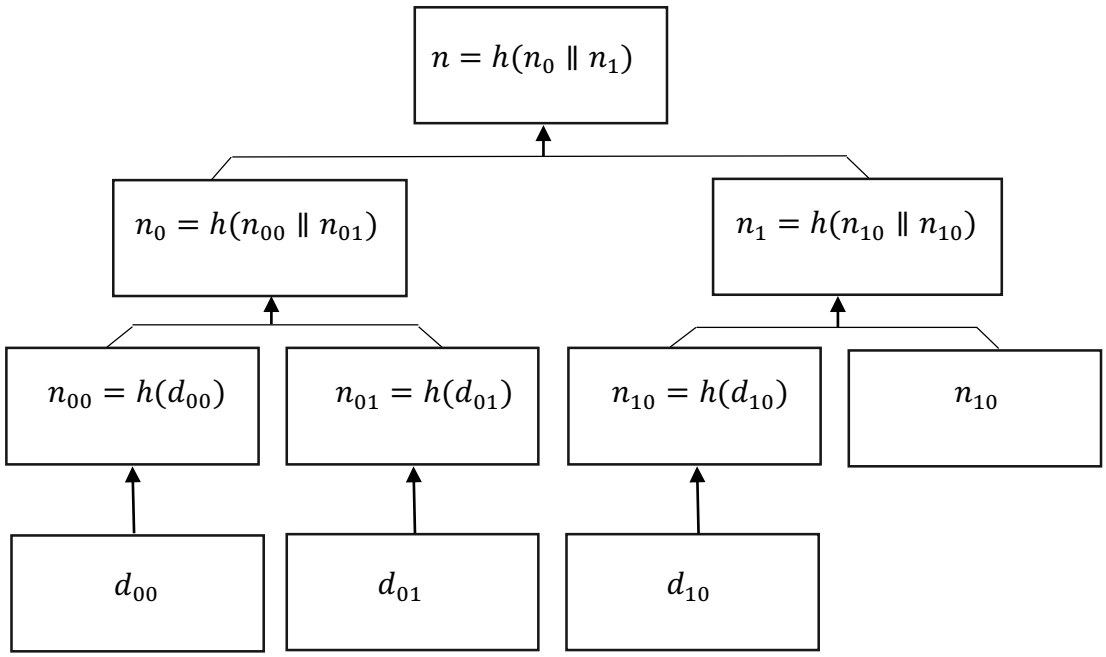


Εικόνα 3.2: Proof-of-work block στο Bitcoin

Η proof-of-work λύνει επίσης και το πρόβλημα προσδιορισμού της αναπαράστασης στην λήψη αποφάσεων με πλειοψηφία. Αν η πλειοψηφία βασιζόταν σε μία ψήφο ανά IP-address τότε θα ήταν ανατρεπόμενη από κάποιον που θα είχε πολλές IP. Η proof-of-work χρησιμοποιεί μία ψήφο ανά κόμβο ο οποίος έχει κάνει την υπολογιστική δουλειά για να ικανοποιήσει το block της proof-of-work. Η πλειοψηφική απόφαση εκπροσωπείται από την πιο μακριά αλυσίδα από blocks, στην οποία έχει γίνει η περισσότερη υπολογιστική δουλειά. Αν η πλειοψηφία της υπολογιστικής δύναμης αποτελείται από τίμιους κόμβους τότε η τίμια αλυσίδα των blocks θα μεγαλώσει πιο γρήγορα από τις άλλες. Για να αλλάξουμε ένα προηγούμενο block, ο επιτιθέμενος θα πρέπει να ξανακάνει την υπολογιστική δουλειά που χρειάζεται για εκείνο το block καθώς και για όλα τα επόμενα συνδεδεμένα block μετά από αυτό και να καταφέρει να ξεπεράσει την δουλεία των τίμιων κόμβων. Η δυσκολία της proof-of-work προσδιορίζεται από την μέση τιμή των blocks ανά ώρα. Όσο πιο γρήγορα παράγονται τα blocks τόσο αυξάνεται και η δυσκολία.

Merkle trees

Τα Merkle trees είναι ένα σημαντικό στοιχείο στο Bitcoin. Είναι ένα δυαδικό δένδρο που αποτελείται από τα hashes και χρησιμοποιείται για την επαλήθευση της ακεραιότητας των δεδομένων αποτελεσματικά και με ασφάλεια. Τα Merkle trees προτάθηκαν από τον Ralph Merkle [51].



Εικόνα 3.3: Merkle Tree

Στην εικόνα 3.3 βλέπουμε ένα παράδειγμα ενός Merkle tree. Σε αυτό το δένδρο κάθε εσωτερικός κόμβος είναι το αποτέλεσμα της Hash function με είσοδο το padding των δύο παιδιών του και τα φύλα απαρτίζονται από δεδομένα. Η τελική τιμή hash n ονομάζεται Merkle root και αποθηκεύεται στο block header. Αφού κάθε εσωτερικός κόμβος αναμένεται να έχει δύο παιδιά, στην ειδική περίπτωση που θα έχουμε μονό αριθμό φύλων (όπως ο κόμβος n_1) τότε το Bitcoin αντιγράφει τον τελευταίο κόμβο.

Μια καλή ιδιότητα των Merkle trees είναι ότι δεν χρειάζεται να ξανά υπολογίσουμε τα hash όλων των δεδομένων σε περίπτωση που ένα data block αλλάξει. Για παράδειγμα, αν το data block d_{00} αλλάξει μόνο τα n_{00}, n_0 και το root n θα ξανά υπολογιστούν.

Block Chain

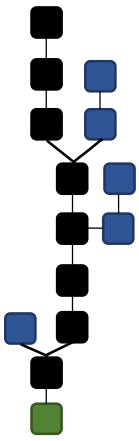
Τα blocks στο Bitcoin περιέχουν πληροφορίες για την συναλλαγή, την proof-of-work και το Hash του προηγούμενου block. Η δομή του block header φαίνεται στην εικόνα 3.4.

Description	Version	Previous Hash	Merkle Root	Timestamp	Bits	Nonce
Size (Bytes)	4	32	32	4	4	4

Εικόνα 3.4: Block Header

Το Previous Hash είναι η τιμή του hash από το προηγούμενο block header. Η Merkle Root χρησιμοποιείται για την επαλήθευση της ακεραιότητας της συναλλαγής, η Timestamp είναι η χρονική στιγμή της παραγωγής του block, τα Bits μας δείχνουν την δυσκολία της proof-of-work και διάφορα nonce δοκιμάζονται για τις απαιτήσεις της Bitcoin proof-of-work.

Η σειρά αυτή των blocks σχηματίζει την αλυσίδα του Bitcoin που ονομάζουμε Block Chain, η οποία κοινοποιείται δημόσια. Η ακολουθία των blocks στην αλυσίδα βασίζεται στον χρόνο που θα επιβεβαιωθεί η συναλλαγή και κάθε καινούργιο block φτιάχνεται πάνω από το προηγούμενό του.



Εικόνα 3.5: Longest Chain Rule

Για την δημιουργία ενός block, οι miners διαλέγουν τα blocks στα οποία περιέχονται συναλλαγές, για επέκταση. Μπορεί όμως κάποιοι άλλοι miners να επιλέξουν διαφορετικά blocks και έτσι η αλυσίδα θα έχει διακλαδώσεις, όπως και φαίνεται στην εικόνα 3.5.

Για να τακτοποιηθούν αυτές οι διακλαδώσεις, το Bitcoin έχει τον κανόνα της πιο μακριάς αλυσίδας (longest chain rule), στον οποίο μόνο η πιο μακριά αλυσίδα γίνεται αποδεκτή, ενώ οι άλλες απορρίπτονται. Στην εικόνα, τα blocks με το μαύρο χρώμα σχηματίζουν την πιο μακριά αλυσίδα και με μωβ χρώμα είναι εκείνα τα blocks με πιο κοντή αλυσίδα που θα απορριφθούν.

Mining

Η εξόρυξη των Bitcoins (mining) έχει δύο σκοπούς, ο ένας είναι η δημιουργία καινούργιων Bitcoins και ο άλλος είναι να σιγουρέψει ότι κάθε συμμετέχων θα έχει μια εικόνα συνέπειας από την καταγραφή των συναλλαγών (transaction log). Αφού η ασυνέπεια των καταγραφών των συναλλαγών μπορεί να δώσει ευκαιρία σε επιτιθέμενους να ξοδέψουν τα Bitcoins τους δύο φορές, το mining επιτρέπει στις επιβεβαιωμένες συναλλαγές να προστίθενται στο block με τις συναλλαγές και να έρχονται σε αντίθεση με αποτέλεσμα οι άκυρες συναλλαγές να απορρίπτονται για την αποφυγή της Double Spending Attack.

Ένας από τους κύριους λόγους που εισήχθηκε αυτό το αποκεντρωτικό σύστημα πληρωμών είναι και το double spending, δηλαδή να μην μπορεί κάποιος να πληρώνει με το ίδιο χρηματικό ποσό δυο οι και περισσότερους χρήστες. Αυτό λύνεται με την καταγραφή τις χρηματικής δραστηριότητας κάθε χρήστη η οποία γίνεται μέσω των transaction που υπάρχουν στα blocks της κύριας αλυσίδας. Έτσι για παράδειγμα αν ο Α στείλει 10 BTC στον Β και 10 BTC στον Γ όπου σαν reference transaction χρησιμοποίει το ίδιο και στις δυο περιπτώσεις ,τότε οι verifying node του δικτύου:

- Στην περίπτωση που το transaction από τον Α στον Β υπάρχει ήδη σε block,τότε θα απορρίψουν το transaction από τον Α στον Γ.

- Στην περίπτωση που έχουν γίνει broadcast την ίδια στιγμή, τότε θα κάνουν ένα από τα δυο δεκτά και θα ενημερώνουν παράλληλα το δυο, έτσι ένα από τα δυο θα επικρατήσει ως δεκτό και το άλλο ως double spending πριν μπουν στο block.

Για να εξορύξουμε τώρα ένα block, οι miners θα πρέπει να υπολογίσουν την proof-of-work με την χρήση της Hash Function SHA256. Το block header γίνεται hash δύο φορές απο την SHA256 για να παραχθεί η τιμή hash. Αν η τιμή hash ξεκινάει με τον απαιτούμενο αριθμό απο μηδενικά, τότε το block έχει εξορυχθεί επιτυχώς και θα προστεθεί στο επίσημο Blockchain. Απο την στιγμή που η proof-of-work είναι δύσκολη να εκτελεστεί, το να δοκιμάσουμε διαφορετικές nonce για να φτάσουμε τις απαραίτητες απαιτήσεις είναι σπάνιο γεγονός και περίπου κάθε 10 λεπτά ένα καινούργιο block έχει εξορυχθεί [59].

3.1.2 Bitcoin συναλλαγές (Transactions)

Συναλλαγή(transaction) στο Bitcoin είναι η διαδικασία μεταφοράς ιδιοκτησίας Bitcoin από μια διεύθυνση Bitcoin σε μια άλλη. Μια Bitcoin διεύθυνση είναι το 160-bit hash του ECDSA δημοσίου κλειδιού και αποθηκεύεται σε Bitcoin πορτοφόλι μαζί με το αντίστοιχο ιδιωτικό κλειδί. Τα Bitcoin πορτοφόλια μπορούν να αποθηκεύσουν μια ή και περισσότερες διευθύνσεις Bitcoin και η κάθε μια μπορεί να χρησιμοποιηθεί μία φορά. Το ιδιωτικό κλειδί χρησιμοποιείται για να υπογράψουμε την συναλλαγή και το δημόσιο κλειδί για επιβεβαίωση των υπογραφών. Θα χωρίσουμε τις συναλλαγές σε δύο μορφές, την απλοποιημένη μορφή και την πιο αναλυτική τις οποίες και θα περιγράψουμε παρακάτω.

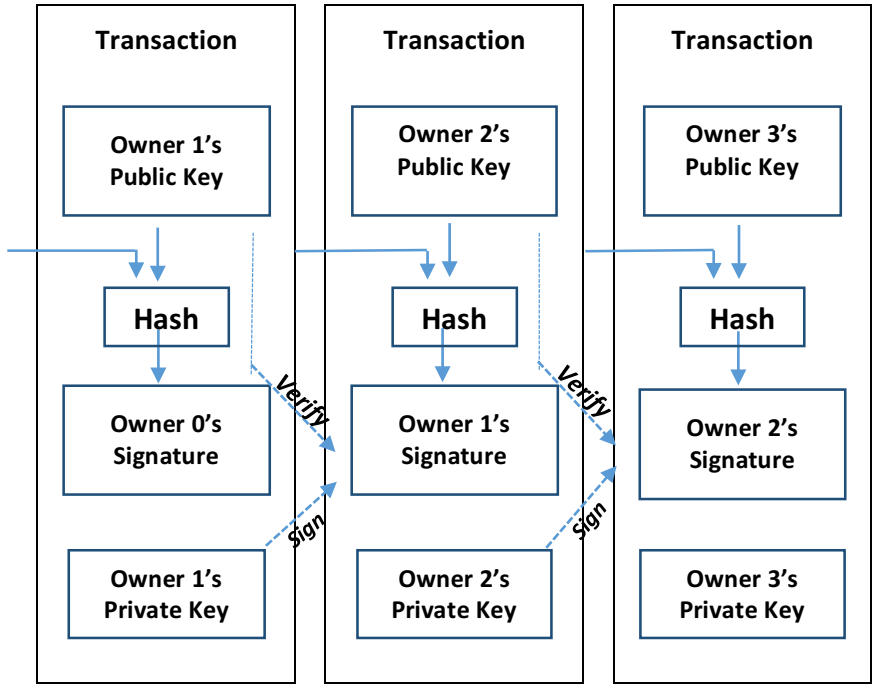
1. **Απλοποιημένη μορφή**: Αρχικά θα περιγράψουμε μια πιο απλοποιημένη μορφή συναλλαγής και στην συνέχεια θα την επεκτείνουμε για να πάρουμε την πιο αναλυτική. Έστω $A = (A_{sk}, A_{pk})$ ένα ζεύγος κλειδιών όπου A_{sk} είναι το ιδιωτικό κλειδί του Α και A_{pk} το δημόσιο. Μια συναλλαγή η οποία μεταφέρει ένα ποσό v (το οποίο ονομάζεται η τιμή της συναλλαγής), από μια διεύθυνση Α σε μια διεύθυνση Β έχει την ακόλουθη μορφή:

$$T_x = (y, B_{pk}, v, sig_A(y, B_{pk}, v))$$

όπου y είναι το index της προηγούμενης συναλλαγής T_y και sig_A η υπογραφή του Α. Λέμε ότι ο Β είναι ο παραλήπτης της T_x και ότι η συναλλαγή T_y είναι η είσοδος της συναλλαγής T_x ή ότι ο Β εξαργυρώνει την συναλλαγή. Πιο συγκεκριμένα, το ποσό των χρημάτων που μεταφέρθηκαν στον Α στην συναλλαγή T_y μεταφέρονται τώρα στον Β. Η συναλλαγή είναι έγκυρη μόνο αν:

   1. Ο Α ήταν ο παραλήπτης της συναλλαγής T_y.
   2. Η τιμή της συναλλαγής T_y ήταν το λιγότερο v, η διαφορά μεταξύ της v και της τιμής της T_y ονομάζεται τέλος συναλλαγής (transaction fee).
   3. Η συναλλαγή T_y δεν έχει εξαργυρωθεί νωρίτερα.
   4. Η υπογραφή του Α είναι σωστή.

Ο Satoshi έδωσε μια απλοποιημένη περιγραφή για το πως λειτουργούν οι συναλλαγές στο Bitcoin, όπως φαίνεται και στην εικόνα 3.6.



Εικόνα 3.6

Παρατηρώντας την μεσαία συναλλαγή από τον Owner 1 στον Owner 2, ο Owner 1 χρησιμοποιεί το ιδιωτικό του κλειδί για να υπογράψει πάνω στην hash τιμή της προηγούμενης συναλλαγής και μαζί με το δημόσιο κλειδί του Owner 2 σχηματίζεται η υπογραφή του Owner 1. Η υπογραφή επαληθεύεται χρησιμοποιώντας το δημόσιο κλειδί του Owner 1. Αφού επαληθευτεί, η συναλλαγή γίνεται έγκυρη και προστίθεται μέσα στο Block.

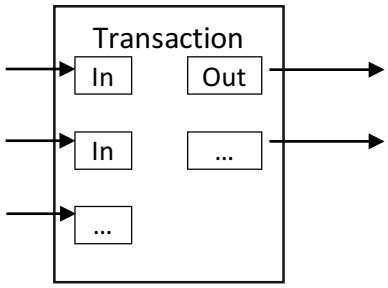
Η πρώτη σημαντική γενίκευση αυτού του απλοποιημένου συστήματος είναι ότι μια συναλλαγή μπορεί να έχει πάνω απο μια είσοδο, δηλαδή μπορεί να έχει εισόδους από αρκετές προηγούμενες συναλλαγές $T_{y_1}, \dots, T_{y_l}$. Έστω $A_1, \dots, A_l$ το σχετικό ζεύγος των κλειδιών των παραληπτών εκείνων των συναλλαγών. Τότε η συναλλαγή με τις πολλαπλές εισόδους έχει την ακόλουθη μορφή:

$$T_x = (y_1, \dots, y_l, B_{pk}, v, sig_{A_1}(y_1, B_{pk}, v), \dots, sig_{A_l}(y_l, B_{pk}, v))$$

έτσι παίρνουμε ως έξοδο της συναλλαγής ότι ο B είναι ο παραλήπτης του ποσού v, υπό την προϋπόθεση ότι είναι το v ίσο με το άθροισμα των τιμών των συναλλαγών $T_{y_1}, \dots, T_{y_l}$. Για να είναι έγκυρη η συναλλαγή θα πρέπει καμία απο τις συναλλαγές να μην έχει εξαργυρωθεί νωρίτερα και όλες οι υπογραφές να είναι έγκυρες. Επίσης κάθε συναλλαγή μπορεί να έχει έναν μετρητή που ονομάζουμε lock-time t ο οποίος μας προσδιορίζει τον χρόνο τον οποίο η συναλλαγή οριστικοποιήθηκε (το t μπορεί να αναφέρεται είτε στο block index είτε στον πραγματικό χρόνο). Στην περίπτωση αυτή έχουμε:

$$T_x = (y_1, \dots, y_l, B_{pk}, v, t, sig_{A_1}(y_1, B_{pk}, v, t), \dots, sig_{A_l}(y_l, B_{pk}, v, t))$$

Μια τέτοια συναλλαγή γίνεται έγκυρη μόνο όταν ο χρόνος t έχει επιτευχθεί και αν καμία από τις προηγούμενες συναλλαγές $T_{y_1}, \dots, T_{y_l}$ έχει εξαργυρωθεί μέχρι την στιγμή t, αλλιώς απορρίπτεται. Κάθε συναλλαγή μπορεί επίσης να έχει πολλές εξόδους, με τον τρόπο αυτό μπορούμε να δώσουμε ρέστα ή να διαμοιράσουμε τα χρήματα σε πολλούς χρήστες.



2. **Αναλυτική μορφή**: θα δούμε τώρα την πραγματική και πιο αναλυτική μορφή συναλλαγής του Bitcoin. Αρχικά έχουμε ότι κάθε συναλλαγή T_x αναγνωρίζεται όχι από το index αλλά από το hash του $H(T_x)$.

   Η βασική διαφορά είναι ότι στην πραγματικότητα το Bitcoin μας δίνει περισσότερη ελευθερία ως προς τον ορισμό της εξαργύρωσης της συναλλαγής. Ας σκεφτούμε την πιο απλή συναλλαγή όπου υπάρχει μόνο μία είσοδος και δεν έχουμε time-locks t, στην απλοποιημένη μορφή του συστήματος που έχουμε περιγράψει πιο πάνω, αν ο χρήστης Α θέλει να εξαργυρώσει την συναλλαγή θα πρέπει να παράγει μια άλλη συναλλαγή T_x υπογεγραμμένη με το ιδιωτικό του κλειδί. Ενώ στο πραγματικό Bitcoin κάθε συναλλαγή T_y έχει την περιγραφή της συνάρτησης output-script π_y της οποίας η έξοδος είναι Boolean. Η συναλλαγή T_x που εξαργυρώνει την συναλλαγή T_y είναι έγκυρη αν και μόνο αν η συνάρτηση π_y έχει ως έξοδο την τιμή true με είσοδο T_x. Ένα παράδειγμα της συνάρτησης π_y είναι αυτή που βλέπει την T_x ως ζεύγος των τιμών m_x (μήνυμα του x) και σ_x (υπογραφή του x) και ελέγχει αν η σ_x είναι έγκυρη υπογραφή του m_x βάσει του δημόσιου κλειδιού του A_{pk}.

   Μια συναλλαγή λοιπόν έχει την ακόλουθη μορφή:

$$T_x = (y, \pi_x, v, \sigma_x)$$

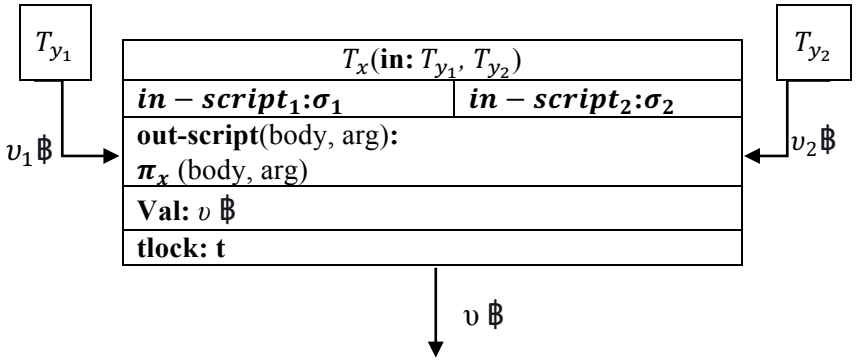
   όπου το $[T_x] = (y, \pi_x, v)$ είναι το σώμα (body) της συναλλαγής T_x και το σ_x είναι η υπογραφή του x η οποία χρησιμοποιείται για να κάνει το output-script π_y να βγάλει ως έξοδο την τιμή true στην T_x. Τα scripts είναι γραμμένα στην γλώσσα προγραμματισμού του Bitcoin. Η γενίκευση των συναλλαγών με πολλαπλές εισόδους και με time-locks είναι της ακόλουθης μορφής:

$$T_x = (y_1, \dots, y_l, \pi_x, v, t, \sigma_1, \dots, \sigma_l)$$

όπου το σώμα (body) της συναλλαγής $[T_x]$ είναι το $(y_1, \dots, y_l, \pi_x, v, t)$ και είναι έγκυρη αν οι επόμενες συνθήκες ικανοποιούνται:

1. Ο χρόνος t έχει επιτευχθεί
2. Κάθε συνάρτηση $\pi_i([T_x], \sigma_i)$ παίρνει την τιμή true, όπου κάθε π_i είναι το output-script της συναλλαγής T_{y_i}
3. Καμία από της συναλλαγές δεν έχει εξαργυρωθεί νωρίτερα

Στην εικόνα 3.7 βλέπουμε μια τέτοια συναλλαγή. Η εξαργύρωση των συναλλαγών συμβολίζεται με βέλη τα οποία θα έχουν πάνω τους την αξία της συναλλαγής. Έστω λοιπόν η συναλλαγή $T_x = (y_1, y_2, \pi_x, v, t, \sigma_1, \sigma_2)$:



Εικόνα 3.7

Οι συναλλαγές όπου έχουν ως είσοδο υπογραφές και ως έξοδο συνάρτηση αλγόριθμου επαλήθευσης είναι το πιο συνηθισμένο είδος συναλλαγών. Θα τις ονομάσουμε standard transactions και την Bitcoin διεύθυνση στην οποία θα γίνεται επαλήθευση θα την ονομάσουμε αποδέκτη (recipient) της συναλλαγής. Κάποιοι miners αποδέχονται μόνο τέτοιου είδους συναλλαγές. Υπάρχουν όμως και κάποιοι που αποδέχονται τις non-standard (ή strange) transactions, στις οποίες αυτό που αλλάζει είναι ότι μπορούμε να ορίσουμε το πως οι συναλλαγές αυτές μπορούν να εξαργυρωθούν και ένα παράδειγμα είναι το mining pool που ονομάζεται Eligius, το οποίο βγάζει ένα καινούργιο block περίπου κάθε μία ώρα. Το mining pool είναι μια συνεργασία από miners στην οποία μοιράζονται τα κέρδη.

3.1.3 Εναλλακτική Κρυπτογραφία Δημοσίου Κλειδιού στο Bitcoin

Το 2013 ο Vitalik Buterin υιοθέτησε τις Lamport υπογραφές για την αντικατάσταση της Κρυπτογραφίας Ελλειπτικών Καμπυλών στο Bitcoin [18]. Η ιδέα αυτή εμπνεύστηκε από την υπόθεση της ύπαρξης κβαντικών υπολογιστών. Ο αλγόριθμος του Shor είναι χρήσιμος στην παραγοντοποίηση αριθμών σε κβαντικούς υπολογιστές και τροποποιημένη εκδοχή του μπορεί να μειώσει σημαντικά τον χρόνο που χρειάζεται κάποιος για να σπάσει την Κρυπτογραφία Ελλειπτικών Καμπυλών. Σε μια Bitcoin συναλλαγή το δημόσιο κλειδί είναι εκτεθειμένο σε μια χρησιμοποιημένη Bitcoin διεύθυνση, το οποίο κάνει την Κρυπτογραφία Ελλειπτικών Καμπυλών ευάλωτη. Οπότε η Lamport one time signature μπορεί να αντικαταστήσει την Κρυπτογραφία Ελλειπτικών Καμπυλών στην κατασκευή του δημοσίου κλειδιού. Η κατασκευή των Bitcoin διευθύνσεων δεν θα αλλάξει, αλλά το δημόσιο κλειδί θα αποτελείται από 320 RIPEMD-160 hashes αντί από σημείο Ελλειπτικής Καμπύλης.

Ο Joseph Binneau και ο Andrew Miller πρότειναν ένα άλλο ψηφιακό νόμισμα που ονομάζεται Fawkescoin το 2014 [15]. Χρησιμοποιούνται μηχανισμοί παρόμοιοι με το Bitcoin αλλά στις συναλλαγές έχει αντικατασταθεί η υπογραφή ECDSA με την hash-based Guy Fawkes υπογραφή, η οποία είναι παρόμοια με την μέθοδο του Buterin. Όμως μια τέτοια μέθοδος, όπως η Lamport υπογραφή θα έβγαζε υπογραφές με χιλιάδες bits και αυτό δεν είναι επιθυμητό στο Bitcoin.

Η πρόταση του Binneau και του Miller με της Guy Fawkes υπογραφές κατασκευάζει τις υπογραφές χρησιμοποιώντας μόνο μία hash function και ένα ασφαλές χρονοδιάγραμμα το οποίο κάνει το Fawkescoin αρκετά αποδοτικό. Η ασφάλεια της συναλλαγής βασίζεται στο γεγονός ότι το μήνυμα που μεταφέρεται γίνεται δημόσια γνωστό πριν την δημοσίευση του x, όπου το x είναι μια τυχαία και ασφαλής τιμή γνωστή μόνο στον ιδιοκτήτη και χρησιμοποιείται για τον υπολογισμό της τιμής (H_x) ως διεύθυνση.

Η αντικατάσταση της Κρυπτογραφίας Δημοσίου Κλειδιού μας δείχνει ότι είναι εφικτό να κατασκευαστούν ψηφιακά νομίσματα σαν το Bitcoin ακόμα και πριν την ανακάλυψη της Κρυπτογραφίας Δημοσίου Κλειδιού. Ακόμα και αν ποτέ οι αλγόριθμοι δημοσίου κλειδιού στο πρωτόκολλο του Bitcoin κινδυνέψουν, τότε τέτοιου είδους εφεδρικές λύσεις μπορούν να χρησιμοποιηθούν.

3.2 Transaction Malleability

Ο όρος Malleability εισάχθηκε πρώτη φορά στην κρυπτογραφία από τον Dolev [31]. Ένα κρυπτογραφικό πρωτόκολλο είναι malleable αν η έξοδος του C μπορεί να μετασχηματιστεί (mauled) σε κάποια παρόμοια τιμή C' από κάποιον που δεν γνωρίζει τα μυστικά που χρησιμοποιήθηκαν για την παραγωγή του C. Για παράδειγμα, ένα σχήμα συμμετρικής κρυπτογραφίας (Enc, Dec) είναι malleable αν η γνώση του Ciphertext $C = Enc(K, M)$ αρκεί για τον υπολογισμό του C' τέτοιου ώστε $M' = Dec(K, C')$ δεν είναι ίσο με το M, αλλά είναι παρόμοιο με αυτό, μπορεί να αλλάζει μόνο το πρώτο bit.

Γενικά οι ορισμοί ασφάλειας στην κρυπτογραφία δεν συνεπάγονται ότι είναι non-malleable με αποτέλεσμα η non-malleability ιδιότητα να θεωρείται ως επιπλέον χαρακτηριστικό των κρυπτοσυστημάτων. Η έννοια της malleability έχει μελετηθεί αρκετά κυρίως από την θεωρητική κοινότητα σε αρκετά διαφορετικά πλαίσια όπως σε encryption και commitment σχήματα, σε zero knowledge πρωτόκολλα, σε multiparty computation πρωτόκολλα, σε hash functions και one-way functions και σε πολλά άλλα.

Τον Φεβρουάριο του 2014, μια από τις μεγαλύτερες εταιρίες ανταλλαγής Bitcoin η MtGox σταμάτησε τις συναλλαγές της λόγο επίθεσης που εκμεταλλεύτηκε την malleability των Bitcoin συναλλαγών και ανακοίνωσε ότι χάθηκαν περίπου 850.000 bitcoins [67]. Υπάρχουν βέβαια υποψίες ότι το MtGox χρησιμοποίησε ως δικαιολογία την malleability όπως υποστηρίζεται στο [28]. Το γεγονός αυτό πάντως αύξησε τις προσπάθειες που έγιναν από κακόβουλους χρήστες για να εκμεταλλευτούν αυτήν την αδυναμία.

Το γεγονός ότι οι Bitcoin συναλλαγές είναι malleable ήταν γνωστό πολύ πριν το περιστατικό στο MtGox [11]. Ο όρος Malleability στην περίπτωση αυτή σημαίνει ότι έχοντας μια συναλλαγή T η οποία μεταφέρει x bitcoins από την διεύθυνση A στην B, μπορούμε να κατασκευάσουμε μια άλλη συναλλαγή T' η οποία είναι συντακτικά διαφορετική από την T αλλά είναι σημασιολογικά ίδιες, δηλαδή η T' μεταφέρει κ αυτή x bitcoins από τον A στον B. Η επίθεση αυτή μπορεί να γίνει απο οποιονδήποτε και χωρίς την γνώση του ιδιωτικού κλειδιού του A.

3.2.1 Πηγές της Malleability ιδιότητας

Στο Bitcoin κάθε συναλλαγή έχει ένα μοναδικό ID, το οποίο ονομάζουμε και transaction ID (TXID) και είναι το hash όλων των περιεχομένων της συναλλαγής. Όλες οι πληροφορίες που εμπλέκονται στην συναλλαγή προστατεύονται από την υπογραφή του χρήστη και δεν μπορούν να πειραχτούν από κακόβουλους χρήστες. Όμως λόγο της γλώσσας προγραμματισμού που χρησιμοποιεί το Bitcoin, οι υπογραφές μπορούν να τροποποιηθούν και να θεωρηθούν έγκυρες. Οι αλλαγές στην υπογραφή θα οδηγήσουν στην αλλαγή της hash τιμής, η οποία είναι το ID της συναλλαγής και έτσι μια συναλλαγή T' θα θεωρηθεί διαφορετική από την T.

Υπάρχουν αρκετοί τρόποι να κατασκευάσουμε την T' από την T. Ένας είναι να εκμεταλλευτείς την malleability που υπάρχει στα σχήματα των υπογραφών στο Bitcoin. Το γεγονός ότι έχοντας μια υπογραφή σ σε κάποιο μήνυμα M, με ιδιωτικό κλειδί s_k είναι εύκολο να κατασκευάσουμε

μια άλλη έγκυρη υπογραφή σ' στο M. Αυτό συμβαίνει λόγο του ECDSA αλγόριθμου που χρησιμοποιείται ο οποίος έχει την ακόλουθη ιδιότητα:

- Για κάθε υπογραφή $\sigma = (r, s) \in \{1, \dots, N-1\}^2$ η τιμή $\sigma' = (r, N - s)$ είναι επίσης μια έγκυρη υπογραφή στο M και με το ίδιο κλειδί s_k.

Αφού λοιπόν οι συναλλαγές στο Bitcoin έχουν την μορφή $T = (\mu\acute{\eta}\nu\upsilon\mu\alpha\ M, \upsilon\pi\sigma\gamma\rho\alpha\phi\acute{\eta}\ \sigma\ \sigma\tau\sigma\ M)$, τότε η $T' = (M, \sigma')$ είναι μια έγκυρη συναλλαγή εννοιολογικά ίδια με την T αλλά συντακτικά διαφορετική. Ένας άλλος τρόπος βασίζεται στο γεγονός ότι το Bitcoin επιτρέπει πιο περίπλοκες συναλλαγές και συγκεκριμένα αυτές που ονομάζουμε non-standard transactions, στις οποίες η υπογραφή σ είναι ένα script της γλώσσας του Bitcoin.

Ο Ken Shirriff περιγράφει λεπτομερώς πως μπορούμε να τροποποιήσουμε το hash της συναλλαγής στο blog του [60]. Τα προγράμματα scriptSig και scriptPubKey είναι γραμμένα στην γλώσσα του Bitcoin, η οποία είναι stack-based, non-Turning complete και χρησιμοποιεί single byte opcodes [29]. Αν έχουμε τα hashes από την αυθεντική και από την τροποποιημένη συναλλαγή, τα αντίστοιχα input και output scripts είναι ίδια αλλά στο ολόκληρο scriptSig υπάρχουν κάποιες διαφορές. Οι διαφορές αυτές φαίνονται με κόκκινο στην εικόνα 3.8 και 3.9.

PUSHDATA 48		48
signature (DER)	sequence	30
	length	45
	integer	02
	length	20
	X	539901ea7d6840eea8826c1f3d0d1fca7827e491deabcf17889e7a2e5a39f5a1
	integer	02
	length	21
	Y	00fe745667e444978c51fdba6981505f0a68619f0289e5ff2352acbd31b3d23d87
SIGHASH_ALL		01
PUSHDATA 41		41
public key	type	04
	X	6c4ea0005563c20336d170e35ae2f168e890da34e63da7fff1cc8f2a54f60dc4
	Y	02b47574d6ce5c6c5d66db0845c7dabcb5d90d0d6ca9b703dc4d02f4501b6e44

Εικόνα 3.8: Αυθεντικό scriptSig

OP_PUSHDATA2 0048		4d 48 00
signature (DER)	sequence	30
	length	45
	integer	02
	length	20
	X	539901ea7d6840eea8826c1f3d0d1fca7827e491deabcf17889e7a2e5a39f5a1
	integer	02
	length	21
	Y	00fe745667e444978c51fdba6981505f0a68619f0289e5ff2352acbd31b3d23d87
SIGHASH_ALL		01
OP_PUSHDATA2 0041		4d 41 00
public key	type	04
	X	6c4ea0005563c20336d170e35ae2f168e890da34e63da7fff1cc8f2a54f60dc4
	Y	02b47574d6ce5c6c5d66db0845c7dabcb5d90d0d6ca9b703dc4d02f4501b6e44

Εικόνα 3.9: Τροποποιημένο scriptSig

Το αυθεντικό scriptSig της συναλλαγής καθορίζει ότι 48 bytes δεδομένων προωθούνται στο stack χρησιμοποιώντας τον opcode PUSHDATA και το τροποποιημένο scriptSig χρησιμοποιεί το OP_PUSHDATA2 (0x4d) για να καθορίσει τον αριθμό των bytes που θα προωθηθούν στα επόμενα δύο bytes 48 00. Αυτό σημαίνει ότι και τα δύο κάνουν το ίδιο πράγμα, προωθούν 48 bytes της υπογραφής στο stack για εκτέλεση. Οπότε και τα δύο scriptSig θεωρούνται έγκυρα ενώ παράγουν διαφορετικές τιμές στο hash της συναλλαγής και έτσι μια malleability επίθεση είναι πιθανή.

Η τροποποίηση των push operations στο scriptSig είναι ο πιο συνηθισμένος τρόπος για την εκτέλεση της malleability επίθεσης. Εκτός απο αυτήν την μέθοδο, ο Pieter Wuille μας δείχνει και κάποιες άλλες πηγές της malleability ιδιότητας στο Bitcoin [68], οι οποίες είναι οι ακόλουθες:

1. *Malleability στις ECDSA υπογραφές*: οι υπογραφές περιγράφουν σημεία πάνω σε ελλειπτική καμπύλη. Αρχίζοντας από την υπογραφή είναι εφικτό να βρεις ένα δεύτερο σύνολο παραμέτρων το οποίο κωδικοποιεί το ίδιο σημείο στην ελλειπτική καμπύλη.

2. *Non-DER encoded ECDSA υπογραφές*: η κρυπτογραφική βιβλιοθήκη που χρησιμοποιείται απο το Bitcoin Core, η OpenSSl, αποδέχεται διάφορα format εκτός από το standardized DER (Distinguished Encoding Rules) encoding.

3. *Επιπλέον Data Pushes*: το scriptPubKey μπορεί να προωθεί επιπλέον δεδομένα στο ξεκίνημα του script τα οποία δεν καταναλώνονται και μένουν στο stack μετά τη λήξη του προγράμματος.

4. *Non-Push λειτουργίες στο scriptSig*: οποιαδήποτε ακολουθία script λειτουργιών στο scriptSig η οποία καταλήγει στο να προωθεί εκείνα τα δεδομένα που προορίζονταν, χωρίς όμως να είναι μια απλή προώθηση δεδομένων, καταλήγει σε μια διαφορετική συναλλαγή με την ίδια εγκυρότητα.

5. *Push λειτουργίες στο scriptSig με μη-κανονικό μέγεθος*: η γλώσσα προγραμματισμού του Bitcoin έχει διάφορους push operators οι οποίοι κάνουν όλοι την ίδια δουλεία.

6. *Zero-padded number pushes*: σε περιπτώσεις που το scriptPubKey opcodes χρησιμοποιεί εισόδους που ερμηνεύονται σε αριθμούς, μπορούν σε αυτούς να προστεθούν μηδενικά.

7. *Δεδομένα που αγνοούνται από τα scripts*: αν το scriptPubKey περιέχει π.χ. το OP_DROP τότε η τελευταία προώθηση δεδομένων στο αντίστοιχο scriptSig θα αγνοείται πάντα.

8. *Sighash flags*: τα Sighash flags μπορούν να χρησιμοποιηθούν για να αγνοηθούν κάποια σημεία του script κατά την διαδικασία της υπογραφής.

9. *Καινούργιες υπογραφές από τον αποστολέα*: ο αποστολέας (ή οποιοσδήποτε με πρόσβαση στο ιδιωτικό του κλειδί) μπορεί πάντα να δημιουργεί καινούργιες υπογραφές.

3.2.2 Malleability Επιθέσεις

Έστω ότι θέλουμε να βάλουμε μια Bitcoin συναλλαγή T στο Blockchain τότε απλά θα μεταδώσουμε την T στο δίκτυο. Η διαδικασία αυτή επιτρέπει στον επιτιθέμενο A να μάθει την T πριν αυτή συμπεριληφθεί στο Blockchain και να παράγει τη σημασιολογικά ισοδύναμη T'. Αν ο A είναι τυχερός τότε οι miners θα συμπεριλάβουν στην Blockchain την T', αντί την T. Αρχικά δεν μοιάζει να είναι μεγάλο το πρόβλημα αφού η T' είναι ισοδύναμη με την T, δηλαδή αν η T μεταφέρει x bitcoins από τον Bob στην Alice τότε και η T' θα κάνει το ίδιο. Ο λόγος που η Malleability μας δημιουργεί πρόβλημα είναι γιατί στο Bitcoin όπως έχουμε πει οι συναλλαγές αναγνωρίζονται από τα hash τους. Έστω μια συναλλαγή $T = (M, \sigma)$, τότε το hash του $H(M, \sigma)$ είναι είναι ο identifier (TXID) της συναλλαγής. Προφανώς τα TXID των δύο συναλλαγών θα είναι διαφορετικά.

Υπάρχουν τρείς περιπτώσεις όπου δημιουργείται πρόβλημα:

Περίπτωση 1: Bitcoin Software Operators

Υπάρχουν κάποιοι Bitcoin Operators οι οποίοι δεν λαμβάνουν υπόψη τους την malleability των συναλλαγών και το MtGox είναι σε αυτήν την κατηγορία. Πιο συγκεκριμένα, η επίθεση που έγινε στο MtGox είναι η ακόλουθη:

1. Ένας κακόβουλος χρήστης P καταθέτει x νομίσματα στο λογαριασμό του στο MtGox.

2. Ο P στην συνέχεια ζητάει από το MtGox να του μεταφέρει τα χρήματα πίσω.

3. Το MtGox εκδίδει μια συναλλαγή T η οποία μεταφέρει τα x νομίσματα στον P.

4. Ο χρήστης P εφαρμόζει την Malleability επίθεση και δημιουργεί την ισοδύναμη με την T αλλά με διαφορετικό TXID T'. Υποθέτουμε ότι miners συμπεριλαμβάνουν την T' στο Blockchain.

5. Ο P λέει στο MtGox ότι η συναλλαγή δεν έγινε ποτέ.

6. Το MtGox ελέγχει και δεν βλέπει καμία συναλλαγή με το TXID της T και βγάζει συμπέρασμα ότι ο χρήστης έχει δίκιο. Οπότε το MtGox δίνει ξανά τα χρήματα πίσω στον P.

Με τον τρόπο αυτό ο P κατάφερε να αποσύρει τα χρήματα του από την τράπεζα δύο φορές. Το πρόβλημα όπως φαίνεται δημιουργείται στο βήμα 6 όπου γίνεται ο έλεγχος από το MtGox, το οποίο ψάχνει για το TXID της συναλλαγής ενώ θα έπρεπε να κοιτάει για συναλλαγές σημασιολογικά ισοδύναμες με την T.

Περίπτωση 2: η διαδικασία "ρέστα"

Η περίπτωση αυτή σχετίζεται με την πρώτη υπό την έννοια ότι δεν δημιουργεί πρόβλημα αν οι χρήστες γνωρίζουν το πρόβλημα της malleability. Το Bitcoin δίνει την δυνατότητα στους χρήστες να δίνουν 'ρέστα'.

Έστω ότι ο χρήστης Α έχει 3 Bitcoins σε μια έξοδο μιας προηγούμενης συναλλαγής T_0 στο Blockchain και θέλει να μεταφέρει το 1 Bitcoin στον χρήστη Β. Η διαδικασία που ακολουθεί ο Α είναι η εξής:

1. Δημιουργεί την συναλλαγή T_1 με είσοδο την T_0 και δύο εξόδους. Η πρώτη έξοδος είναι για τον Β με τιμή συναλλαγής 1 bitcoin και η δεύτερη είναι για τον εαυτό του με τιμή 2 bitcoins.

2. Στην συνέχεια μεταδίδει την T_1 στην Blockchain.

3. Δημιουργεί την συναλλαγή T_2 με είσοδο την T_1 με σκοπό να πάρει τα 2 Bitcoins.

Αν τώρα ο Α στο βήμα 2 δεν περιμένει να εμφανιστεί η T_1 στο Blockchain και κάποιος την έχει τροποποιήσει και οι miners εμφάνισαν την αλλαγμένη συναλλαγή T_1' στο Blockchain, τότε η δεύτερη συναλλαγή T_2 γίνεται άκυρη.

Περίπτωση 3: Bitcoin Contracts

Η τρίτη περίπτωση αφορά τα Bitcoins Contracts. Τα πρωτόκολλα αυτά σχηματίζουν οικονομικές συμφωνίες ανάμεσα σε αμοιβαία μη έμπιστους χρήστες. Τα συμβόλαια αυτά χρησιμοποιούν τις non-standard συναλλαγές. Μια από τις τεχνικές που χρησιμοποιείται είναι ότι αφήνει τους χρήστες να υπογράψουν μια συναλλαγή T_1 πριν προλάβει η είσοδος της T_0 να εμφανιστεί στην Blockchain. Το πρόβλημα είναι ότι η T_1 χρησιμοποιεί το TXID $H(T_0)$, οπότε αν ο επιτιθέμενος εφαρμόσει την malleability επίθεση στο T_0 τότε η T_1 ακυρώνεται. Σε αντίθεση με τις άλλες δύο περιπτώσεις που το πρόβλημα είναι στην εφαρμογή του software, στα Bitcoin contracts είναι πρόβλημα πρωτοκόλλου.

3.2.3 Λύσεις του malleability προβλήματος

Αρκετές προτάσεις έχουν γίνει προς την αλλαγή του πρωτοκόλλου του Bitcoin με σκοπό να εξαλειφθεί η Malleability ιδιότητα από αυτό. Δυστυχώς όμως το να αλλαχθεί το Bitcoin είναι γενικά δύσκολο λόγο του ότι δεν ελέγχεται από κάποια κεντρική αρχή και έτσι οποιαδήποτε τροποποίηση μπορεί να επιφέρει προβλήματα, όπως καταστάσεις όπου υπάρχουν διαφωνίες ανάμεσα στους τίμιους χρήστες ως προς την κατάσταση των συναλλαγών. Ένα τέτοιο παράδειγμα είναι το bug που υπήρχε σε ένα update ενός δημοφιλούς mining τον Μάρτιο του 2013 [19]. Η Blockchain χωρίστηκε στα δύο και για 6 ώρες υπήρχαν δύο δίκτυα Bitcoin το καθένα με την δική του ιστορία συναλλαγών. Ευτυχώς μετά απο 24 blocks το πρόβλημα επιλύθηκε.

Στην συνέχεια θα δούμε τρείς διαφορετικές προτάσεις που επιλύουν το Malleability πρόβλημα:

Λύση 1

Η πρώτη πρόταση είναι του Pieter Wuille [68], ο οποίος παρουσιάζει κάποιους καινούργιους κανόνες για την αντιμετώπιση του Malleability προβλήματος περιορίζοντας την σύνταξη των Bitcoin συναλλαγών. Οι κανόνες αυτοί είναι οι ακόλουθοι:

1. *Κανονική κωδικοποίηση ECDSA υπογραφών*: μια ECDSA υπογραφή στο OP_CHECHSIG, OP_CHECKSIGVERIFY, OP_CHECKMYLTISIG ή στο OP_CHECKMYLTISIGVERIFY πρέπει να χρησιμοποιεί αυστηρά DER-encoding. Αν κάνουμε επαλήθευση σε υπογραφή χωρίς DER-encoding τότε αυτό κάνει το script να βγάζει False.

2. *Non-Push λειτουργίες στο scriptSig:* στο scriptSig επιτρέπονται μόνο data pushes. Η αξιολόγηση οποιασδήποτε άλλης λειτουργίας κάνει το script να βγάζει False.

3. *Push λειτουργίες στο scriptSig με μη-κανονικό μέγεθος:* η μικρότερη δυνατή push λειτουργία πρέπει να χρησιμοποιείται όποτε αυτό είναι δυνατό. Αν χρησιμοποιήσουμε μια λειτουργία για ένα push data η οποία μπορούσε να κωδικοποιηθεί με συντομότερο τρόπο τότε το script βγάζει False.

4. *Zero-padded number pushes:* όποτε ένα script opcode καταναλώνει μια stack τιμή ερμηνευμένη ως αριθμό, η κωδικοποίηση πρέπει να γίνεται με τον συντομότερο τρόπο. Η πρόσθεση μηδενικών δεν επιτρέπεται.

5. *Malleability στις ECDSA υπογραφές:* Για κάθε υπογραφή $\sigma = (r, s) \in \{1, \dots, N - 1\}^2$ η τιμή $\sigma' = (r, N - s)$ είναι επίσης μια έγκυρη υπογραφή στο M και με το ίδιο κλειδί s_k. Απαιτούμε λοιπόν στην τιμή s μέσα στις υπογραφές να είναι $s < \frac{\#order}{2}$ όπου $\#order$ είναι η τάξη της καμπύλης.

6. *Superfluous λειτουργίες στο scriptSig:* αξιολόγηση του scriptPubKey θα απαιτείται με αποτέλεσμα μίας μη μηδενικής τιμής. Αν υπάρχουν παραπάνω στοιχεία δεδομένων στο stack τότε το script βγάζει False.

7. *Δεδομένα που αγνοούνται από τα scripts:* τα παραπάνω μη χρήσιμα στοιχεία στο stack τα οποία καταναλώνονται από την OP_CHECKMULTISIG και από την OP_CHECKMULTISIGVERIFY πρέπει να είναι το empty byte array (αποτέλεσμα της OP_0). Οτιδήποτε άλλο θα κάνει το script να βγάζει False.

Λύση 2

Η δεύτερη πρόταση είναι του Peter Todd [10] ο οποίος μας παρουσιάζει μια καινούργια λειτουργία στην γλώσσα προγραμματισμού του Bitcoin την OP_CHECKLOCKTIMEVERIFY, η οποία επιτρέπει στις εξόδους των συναλλαγών να μην μπορούν να ξοδευτούν μέχρι κάποια στιγμή στο μέλλον.

Η CHECKLOCKTIMEVERIFY επαναπροσδιορίζει την λειτουργία του NOP2 (No Operation) opcode. Όταν εκτελείται το πρόγραμμα, αν κάποια από τις ακόλουθες συνθήκες είναι αληθής, τότε το script θα τερματίσει με λάθος:

1. Το stack είναι άδειο

2. Το πρώτο στοιχείο της λίστας στο stack είναι μικρότερο από το 0

3. Το lock-time του πρώτου στοιχείου της λίστας στο stack και το πεδίο nLockTime δεν είναι τα ίδια

4. Το πρώτο στοιχείο της λίστας στο stack είναι μεγαλύτερο από το πεδίο nLockTime της συναλλαγής

5. Το πεδίο nSequence της txin είναι 0xffffffff

Αν οι παραπάνω συνθήκες δεν είναι αληθής τότε η εκτέλεση του script θα συνεχίσει σαν να είχε εκτελεστεί η λειτουργία NOP.

Το πεδίο nLockTime σε μια συναλλαγή αποτρέπει την εξόρυξη αυτής μέχρι να φτάσει σε ένα συγκεκριμένο ύψος το block ή να επιτευχθεί ο χρόνος του block. Συγκρίνοντας το επιχείρημα της CHECKLOCKTIMEVERIFY με το πεδίο της nLockTime, βλέπουμε ότι έμμεσα γίνεται επιβεβαίωση για το αν έχει φτάσει το block στο επιθυμητό ύψος ή αν ο χρόνος του block έχει επιτευχθεί και μέχρι να ισχύσει μια από τις δύο παραπάνω συνθήκες η έξοδος της συναλλαγής δεν μπορεί να ξοδευτεί.

Η πρόταση αυτή μπορεί να μην αντιμετωπίζει άμεσα το πρόβλημα της malleability αλλά χρησιμοποιώντας αυτήν την λειτουργία μπορούμε να κατασκευάσουμε συναλλαγές που είναι ανθεκτικές στην Malleability ιδιότητα.

Λύση 3

Η Τρίτη πρόταση είναι των Marcin, Stefan, Daniel και Lukasz [1] παρουσιάζει αλλαγές στο πρωτόκολλο του Bitcoin οι οποίες δημιουργούν ανθεκτικές στην Malleability συναλλαγές.

Στην τωρινή εκδοχή του πρωτοκόλλου του Bitcoin κάθε συναλλαγή περιέχει το hash της συναλλαγής που ξοδεύει. Το hash υπολογίζεται πάνω από όλη την συναλλαγή. Η αλλαγή που προτείνεται είναι ότι ο υπολογισμός του hash να γίνεται πάνω από την συναλλαγή αλλά χωρίς τα input scripts, δηλαδή μόνο πάνω από το body της συναλλαγής, όπως πραγματοποιούνται και τα hash των υπογραφών στις συναλλαγές. Αυτό σημαίνει ότι η τιμή του hash της συναλλαγής θα είναι ίδια ανεξαρτήτως από τα input scripts.

Προφανώς ένας επιτιθέμενος θα μπορεί ακόμα να αλλάξει το input script της συναλλαγής και να αναμεταδώσει την τροποποιημένη συναλλαγή στο δίκτυο αλλά τα hash των δύο συναλλαγών θα είναι ίδια και έτσι δεν θα έχει σημασία πια από τις δύο συναλλαγές θα συμπεριληφθεί στην Blockchain.

Επίσης με αυτήν την τροποποίηση του πρωτοκόλλου θα είμαστε σε θέση να υπογράφουμε μια αλυσίδα από συναλλαγές ακόμα και αν δεν ξέρουμε τα input scripts για κάποιες από αυτές. Το μόνο απαραίτητο στοιχείο για να υπογράψουμε θα είναι τα output scripts, οι τιμές των συναλλαγών και τα hashes των συναλλαγών που εξαργυρώνονται από την πρώτη συναλλαγή στην αλυσίδα.

Με την τροποποίηση αυτή τα input scripts χρησιμοποιούνται μόνο για να δείξουν ότι μια συναλλαγή εξουσιοδοτείται για να εξαργυρώσει άλλες συναλλαγές. Οπότε δύο συναλλαγές που διαφέρουν μόνο στα input scripts είναι ισοδύναμες. Η Blockchain δεν θα περιέχει ποτέ δυο τέτοιες συναλλαγές και έτσι το hash θα αναγνωρίζει μοναδικά την συναλλαγή που είναι να εξαργυρωθεί.

3.2.4 Πειράματα και αποτελέσματα της Malleability επίθεσης

Τώρα θα δούμε κάποια πειράματα τα οποία έκαναν οι Marcin, Stefan, Daniel και Lukasz [2] με σκοπό να δούνε και να μετρήσουν την αποτελεσματικότητα της επίθεσης αυτής. Οι επιθέσεις αυτές έγιναν σε συναλλαγές που είχαν εκδοθεί από τους ίδιους.

Υλοποίηση της Malleability επίθεσης

Για να εκτελέσουν οι συγγραφείς του [2] μια Malleability επίθεση εφάρμοσαν ένα πρόγραμμα που ονομάζεται Adversary και χρησιμοποίησαν την bitcoinj βιβλιοθήκη. Το πρόγραμμα αυτό είναι συνδεδεμένο σαν ένα peer στο δίκτυο του Bitcoin και βλέπει συναλλαγές οι οποίες στέλνουν Bitcoin σε μια συγκεκριμένη διεύθυνση την Addr η οποία ανήκει στους ίδιους. Όταν το πρόγραμμα Adversary βλέπει μια τέτοια συναλλαγή, τότε την τροποποιεί αλλάζοντας την ECDSA υπογραφή από (r, s) σε (r, N-s) και στην συνέχεια μεταδίδει την τροποποιημένη συναλλαγή στο δίκτυο.

Η αποτελεσματικότητα της επίθεσης μετριέται σε ποσοστό των περιπτώσεων που η τροποποιημένη συναλλαγή μπαίνει στην Blockchain και η αυθεντική απορρίπτεται. Αυτό εξαρτάται απο την δομή του δικτύου και από τους miners, οι οποίοι λαμβάνουν την τροποποιημένη συναλλαγή πριν την αυθεντική. Αν θέλουμε η επίθεση να είναι όσο πιο αποτελεσματική γίνεται θα πρέπει να προωθήσουμε την τροποποιημένη συναλλαγή σε όλο το peer-to-peer δίκτυο όσο πιο γρήγορα γίνεται. Οπότε, ο Adversary συνδέεται σε πολύ περισσότερα peers από έναν τυπικό Bitcoin client και διατηρεί σε μέσο όρο περίπου 1000 συνδέσεις, ενώ ένας τυπικός Bitcoin client διατηρεί περίπου 8. Επιπλέον συνδέεται απευθείας σε κάποιους κόμβους οι οποίοι διατηρούνται από mining pools και στέλνει τις τροποποιημένες συναλλαγές πρώτα σε αυτούς.

Ανάλυση αποτελεσματικότητας της Malleability επίθεσης

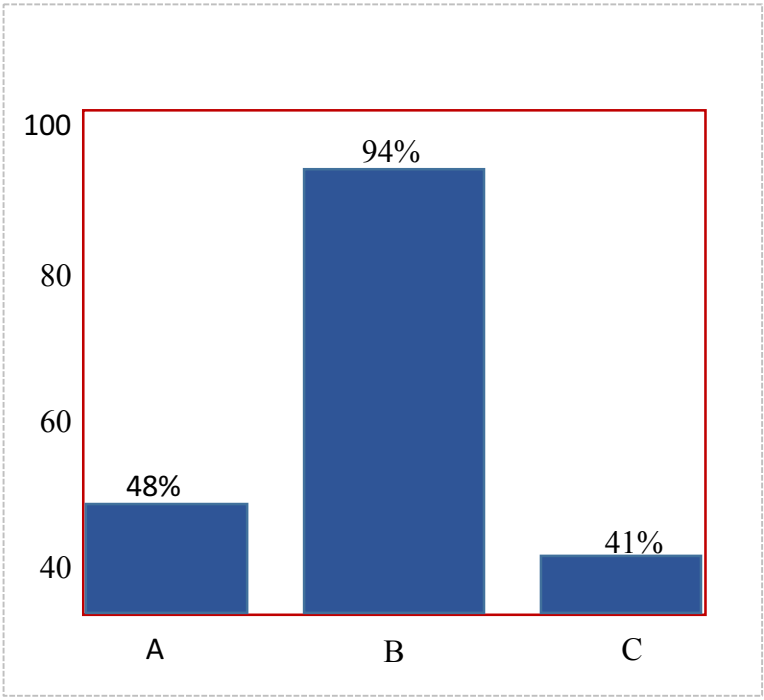
Για να μετρηθεί η αποτελεσματικότητα της επίθεσης δημιουργήσαμε ένα άλλο πρόγραμμα που ονομάζεται Victim, το οποίο κάνει πολλές συναλλαγές στέλνοντας Bitcoins στην διεύθυνση Addr. Πιο συγκεκριμένα μετρήθηκε η αποτελεσματικότητα της επίθεσης σε τρείς διαφορετικές περιπτώσεις:

A. Η IP διεύθυνση του Victim δεν είναι γνωστή στον Adversary.

B. Η IP διεύθυνση του Victim είναι γνωστή στον Adversary. Στην περίπτωση αυτή ο Adversary μπορεί να συνδεθεί απευθείας στο Victim και έτσι βλέπει πιο γρήγορα τις συναλλαγές που μεταδίδει. Στα πειράματα που έγιναν και τα δύο προγράμματα Adversary και Victim ήταν συνδεδεμένα στο ίδιο τοπικό δίκτυο και αυτά έδειξαν ότι με αυτόν τον τρόπο αυξάνεται η αποτελεσματικότητα της επίθεσης.

C. Το Victim γνωρίζει για το Malleability πρόβλημα και προσπαθεί να μεταδίδει τις συναλλαγές του σε περισσότερους κόμβους. Συγκεκριμένα το Victim είναι συνδεδεμένο σε 100 peers στο δίκτυο και η IP διεύθυνση δεν ήταν γνωστή στον Adversary.

Τα αποτελέσματα φαίνονται στην εικόνα 3.10.



Εικόνα 3.10

Η αποτελεσματικότητα εξαρτάται από τους κόμβους που είναι συνδεδεμένο το Victim. Μετά από κάθε συναλλαγή ακυρώνει τις συνδέσεις που είχε δημιουργήσει και φτιάχνει καινούργιες. Επιπλέον η αποτελεσματικότητα εξαρτάται από την κατανομή των blocks που έχουν εξορυχθεί μεταξύ των miners στην περίοδο δοκιμών, οπότε τα πειράματα έγιναν σε μεγάλες χρονικές περιόδους (π.χ. 24 ώρες). Με σκοπό να αποκλειστούν παράγοντες όπως η φυσική εγγύτητα των Adversary και Victim λόγο του γεγονότος ότι βρίσκονται στο ίδιο δίκτυο κάποιες από τις επιθέσεις έγιναν με τον Adversary και Victim να λειτουργούν σε διαφορετικά δίκτυα.

Δοκιμές Bitcoin client

Με σκοπό να προσδιορίσουμε την σημασία του Malleability προβλήματος για τους ανεξάρτητους Bitcoin χρήστες θα δούμε κάποια πειράματα που έγιναν στα πιο γνωστά Bitcoin πορτοφόλια όπου βλέπουμε πως αυτά συμπεριφέρονται όταν μια συναλλαγή τροποποιηθεί. Οι δοκιμές έγιναν σε 14 Bitcoin πορτοφόλια. Σε κάθε δοκιμή αρκετές Bitcoin μεταφορές έγιναν από το πορτοφόλι στην διεύθυνση Addr. Κατά την διάρκεια των δοκιμών ο Adversary προσπαθούσε να τροποποιήσει κάθε συναλλαγή προς αυτήν την διεύθυνση.

Στα πειράματα αυτά παρατηρήθηκε ότι:

1. Οι περισσότεροι Bitcoin clients προσδιορίζουν αν η συναλλαγή έχει εγκριθεί κοιτάζοντας στην Blockchain για μια συναλλαγή με το ίδιο hash.

2. Οι Bitcoin clients είναι πιθανό να λάβουν από το δίκτυο την τροποποιημένη συναλλαγή και έτσι να συμπεριλάβουν στο ιστορικό των συναλλαγών είτε την τροποποιημένη είτε την αυθεντική είτε και τις δύο.

Η τελευταία παρατήρηση μπορεί να οδηγήσει σε απρόβλεπτες συμπεριφορές στο πορτοφόλι του χρήστη. Οι συμπεριφορές που παρατηρήθηκαν είναι οι ακόλουθες:

a. Το πορτοφόλι εσφαλμένα υπολογίζει την συναλλαγή.

b. Το πορτοφόλι δεν μπορεί να κάνει καμία συναλλαγή γιατί υποθέτει ότι κάποια συναλλαγή θα επιβεβαιωθεί στο μέλλον, κάτι που δεν θα γίνει ποτέ αφού η συναλλαγή έχει τροποποιηθεί.

c. Η εφαρμογή σταματάει να λειτουργεί.

Αν ο client δεν μπορεί πλέον να κάνει κάποια συναλλαγή και η αφαίρεση της προβληματικής συναλλαγής από την ιστορία των συναλλαγών δεν είναι δυνατή, τότε ο χρήστης δεν θα μπορεί να ξοδέψει τα Bitcoins που έχει στο πορτοφόλι του για πάντα. Επιπλέον, λόγο της διαδικασίας ‘ρέστα’, μια επίθεση σε μία μόνο συναλλαγή θα μπορούσε να κάνει όλα τα χρήματα από το πορτοφόλι να μην μπορούν να ξοδευτούν.

Στην εικόνα 3.11 βλέπουμε τα αποτελέσματα των δοκιμών. Όπου (a), (b) και (c) είναι οι συμπεριφορές των πορτοφολιών που παρατηρήθηκαν πιο πάνω.

Wallet name	Type	(a)	(b)	(c)	When the problem disappears
Bitcoin Core	Desktop				-
Xapo	Web				-
Armory	Desktop	✗			never
Green Address	Desktop	✗			never
Blockchain.info	Web	✗			after six blocks without confirmation
Coinkite	Web	✗	✗		after several blocks without confirmation
Coinbase	Web	✗	✗		after several hours
Electrum	Desktop	✗	✗		after application reset
MultiBit	Desktop	✗	✗		after “Reset block chain and transactions” procedure
Bitcoin Wallet	Mobile	✗	✗		after “Reset block chain” procedure
KnC Wallet	Mobile	✗	✗	✗	after “Wallet reset” procedure
Hive	Desktop	✗	✗	✗	after restoring the wallet from backup
BitGo	Web	✗	✗		never
Mycelium	mobile	✗	✗		never

Εικόνα 3.11

Συγκεκριμένα το Bitcoin Core και Xapo φαίνεται ότι αντιμετωπίζουν το Malleability πρόβλημα σωστά. Το Bitcoin Core εντοπίζει την Malleability επίθεση και την σημειώνει ως “double spending”. Το Green Address και Armory παρουσιάζουν λάθος υπόλοιπα λογαριασμών αλλά όταν επιτρέπουν στον χρήστη να κάνει μια καινούργια συναλλαγή τότε λαμβάνουν υπόψιν τους το πραγματικό υπόλοιπο. Τα Blockchain.info, Coinkite, Coinbase, Electrum, MultiBit, Bitcoin Wallet και KnC Wallet μπορεί να κολλήσουν περιμένοντας την επιβεβαίωση της συναλλαγής, η οποία δεν θα γίνει ποτέ με αποτέλεσμα ο χρήστης να μην μπορεί να κάνει επόμενη συναλλαγή. Οι clients αυτοί όμως δίνουν στον χρήστη την δυνατότητα να συγχρονίσει την λίστα των συναλλαγών με την Blockchain ή το κάνουν αυτόματα μετά από κάποιο χρονικό διάστημα και έτσι το πρόβλημα εξαφανίζεται. Στο Hive υπήρχε η δυνατότητα του reset της συναλλαγής μόνο χειροκίνητα. Στα BitGo και Mycelium το πρόβλημα ήταν πιο σοβαρό καθώς εμφάνιζαν την προβληματική συναλλαγή και δεν υπήρχε κάποιος τρόπος να κάνεις reset την λίστα των συναλλαγών.

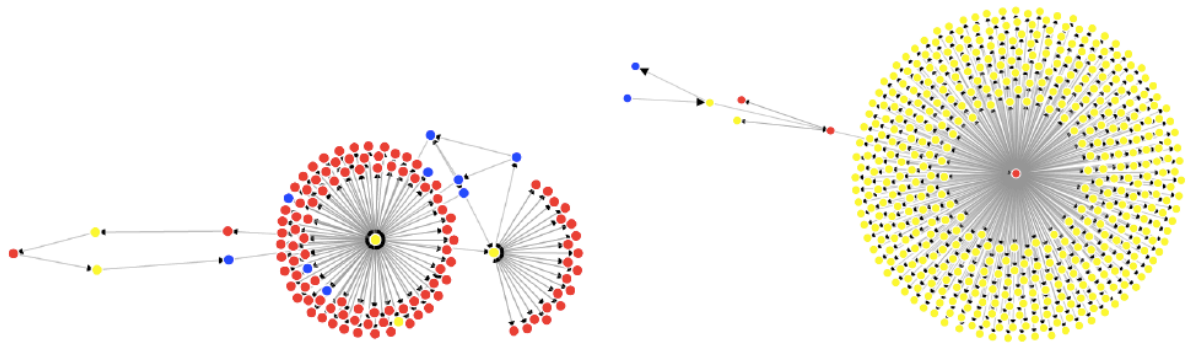
3.3 Κρυπτογραφικός έλεγχος στο Bitcoin

Στην ενότητα αυτή θα δούμε κάποιους κρυπτογραφικούς ελέγχους οι οποίοι έγιναν σε δεδομένα από την Bitcoin Blockchain που συγκεντρώθηκαν από τους Joppe W.Bos, J.Alex Halderman, Nadia Heninger, Jonathan Moore, Michael Naehrig και Eric Wustrow [16], με σκοπό να εντοπιστούν προβλήματα υλοποίησης τα οποία μπορεί να οδηγήσουν σε κρυπτογραφικές αδυναμίες.

3.3.1 Επαναλαμβανόμενα Ανά-Μήνυμα Μυστικά

Αρχικά έγινε μια εξαγωγή 47.093.121 σημείων ελλειπτικής καμπύλης από τις υπογραφές που συγκεντρώθηκαν και επιβεβαιώθηκε ότι είναι σωστά, δηλαδή ότι τα σημεία όντως ανήκουν στην καμπύλη secp256k1 η οποία χρησιμοποιείται στο Bitcoin. Επίσης έγινε έλεγχος για διπλές nonce (Number used ONCE) στις υπογραφές και βρέθηκε ότι 158 μοναδικά δημόσια κλειδιά είχαν χρησιμοποιήσει τις ίδιες nonce για την παραγωγή της r τιμής σε παραπάνω από μια υπογραφές, πράγμα που όπως έχουμε δει αφήνει εκτεθειμένα τα ιδιωτικά κλειδιά των χρηστών. Βρέθηκε ότι το συνολικό υπόλοιπο και των 158 αυτών λογαριασμών είναι πολύ μικρό, συγκεκριμένα 0.00031217 BTC, το οποίο ποσό δεν φτάνει ούτε για το transaction fee που χρειάζεται για να τα αξιώσεις. Βρέθηκε όμως και μια διεύθυνση η οποία φαίνεται ότι έχει κλεμμένα Bitcoins από 10 άλλες διευθύνσεις. Ο λογαριασμός αυτός έκανε 11 συναλλαγές στην περίοδο από Μάρτιο ως τον Οκτώβριο του 2013 και κάθε συναλλαγή περιείχε στην είσοδό της διευθύνσεις από αυτές που είχαν διπλές nonce υπογραφών. Οι συναλλαγές αυτές έδωσαν κέρδος στον λογαριασμό 59 BTC.

Για να γίνει κατανοητή η βασική αιτία των επαναλαμβανόμενων nonce των υπογραφών, οι συγγραφείς δημιούργησαν ένα γράφημα συναλλαγών αρχίζοντας από τις αδύναμες διευθύνσεις και προσθέτοντας ακμές σε διευθύνσεις δείχνοντας αν έχουν κάνει συναλλαγές μεταξύ τους. Στην συνέχεια δημιουργήθηκαν ακμές από εκείνες τις διευθύνσεις στο δεύτερο επίπεδο. Αυτό δημιούργησε πέντε ξεχωριστές συνεκτικές συνιστώσες με την μεγαλύτερη να περιέχει 1649 διευθύνσεις. Στην εικόνα 3.14 βλέπουμε το δεύτερο και τρίτο μεγαλύτερο συνεκτικό γράφημα. Με κόκκινο είναι οι κόμβοι οι οποίοι έχουν διπλές nonce στις υπογραφές και με κίτρινο και μπλε είναι αυτές οι διευθύνσεις των επόμενων επιπέδων που απομακρύνονται από το γράφημα των συναλλαγών. Η μοναδική σχεδίαση των δύο αυτών γραφημάτων υποδηλώνει ότι υπάρχουν αρκετά τέτοια σύνολα χρηστών ή υλοποιήσεων που δημιουργούν αυτού του είδους το πρόβλημα.



Εικόνα 3.14

Από το Bitcoincard αναγνωρίστηκαν 3 κλειδιά τα οποία ανήκαν σε αυτό, μια ενσωματωμένη συσκευή η οποία λειτουργούσε ως αυτόνομος Bitcoin client. Βρέθηκαν επίσης αρκετοί λογαριασμοί οι οποίοι ανήκουν στο Blockchain.info με διπλές nonce λόγο ενός bug στο JavaScript του client στην γεννήτρια τυχαίων αριθμών η οποία δεν χρησιμοποιούσε σωστά το seed [36]. Τα χρήματα αυτά μεταφέρθηκαν στην συνέχεια στην διεύθυνση που προαναφέραμε. Να σημειώσουμε ότι υπάρχει ένα timestamping σχήμα το οποίο διαρρέει το ιδιωτικό κλειδί μιας συναλλαγής χρησιμοποιώντας εσκεμμένα την ίδια τυχαία τιμή nonce [23]. Αν το σχήμα αυτό υλοποιηθεί και δοκιμαστεί τότε αυτό μπορεί να εξηγήσει υπογεγραμμένες συναλλαγές με διπλές nonce που περιέχουν ένα μικρό ποσό από Bitcoins.

3.3.2 Bitcoins που δεν μπορούν να ξοδευτούν και περίεργες HASH160 τιμές

Είναι πιθανό να υπάρξει μεταφορά Bitcoin σε διεύθυνση πού δεν υπάρχει αντίστοιχο ζεύγος κλειδιών. Τα Bitcoins αυτά παραμένουν για πάντα εκεί και δεν ξαναχρησιμοποιούνται ποτέ. Αυτό θα μπορούσε να οδηγήσει σε υποτίμηση, αυξάνοντας έτσι την αξία των υπόλοιπων Bitcoins. Αυτό συμβαίνει και με τα φυσικά χρήματα τα οποία μπορεί και αυτά να καταστραφούν. Το Υπουργείο Οικονομικών της U.S. εξαγοράζει κατεστραμμένα χρήματα κάθε χρόνο αξίας 30 εκατομμυρίων [17].

Ερευνήθηκε το κατώτερο όριο του αριθμού των Bitcoins τα οποία δεν μπορούν να ξοδευτούν. Οι Bitcoin διευθύνσεις προκύπτουν από την τιμή της HASH160 και έτσι αρκετοί χρήστες έχουν χρησιμοποιήσει μη έγκυρες τιμές για το ECDSA δημόσιο κλειδί. Συναλλαγές σε διευθύνσεις χωρίς τα αντίστοιχα ζεύγη κλειδιών είναι δυνατές λόγο του ότι τα ECDSA κλειδιά χρειάζονται μόνο αν θες να ξοδέψεις τα χρήματα. Έχοντας μια Bitcoin διεύθυνση ή την τιμή της HASH160, δεν είναι εφικτό να υπολογίσεις το αντίστοιχο ζεύγος κλειδιών λόγο των ιδιοτήτων των Hash Functions.

Απο την στιγμή που δεν χρειάζεται να γνωρίζεις κάποιο κλειδί για να παράγεις μια Bitcoin διεύθυνση παρά μόνο την HASH160 τιμή, αρχικά έγινε έλεγχος μήπως βρεθούν διευθύνσεις οι οποίες έχουν μια μικρή HASH160 τιμή i, όπου $0 \leq i < 100$. Βρέθηκε ότι οι 9 πρώτες τιμές υπάρχουν και έχουν μη μηδενικό υπόλοιπο. Με κίνητρο αυτό το γεγονός έγινε έλεγχος για επαναλαμβανόμενα σχέδια όταν η τιμή της HASH160 παρουσιάζεται σε δεκαεξαδική μορφή.

Όλες οι 16 αυτές πιθανότητες υπάρχουν και 3 από αυτές έχουν μη μηδενικό υπόλοιπο όπως φαίνεται και στην εικόνα 3.15 όπου βλέπουμε τις περίεργες HASH160 τιμές, τα δημόσια κλειδιά και τις αντίστοιχες Bitcoin διευθύνσεις με το υπόλοιπό τους.

HASH160	Bitcoin address	balance in BTC
0000000000000000000000000000000000000000	1111111111111111111114oLvT2	2.94896715
0000000000000000000000000000000000000001	1111111111111111111BZbvjr	0.01000000
0000000000000000000000000000000000000002	1111111111111111111HeBAGj	0.00000001
0000000000000000000000000000000000000003	1111111111111111111QekFQw	0.00000001
0000000000000000000000000000000000000004	1111111111111111111UpYBrS	0.00000001
0000000000000000000000000000000000000005	1111111111111111111g4hiWR	0.00000001
0000000000000000000000000000000000000006	1111111111111111111jGyPM8	0.00000001
0000000000000000000000000000000000000007	1111111111111111111o9FmEC	0.00000001
0000000000000000000000000000000000000008	1111111111111111111ufYVpS	0.00000001
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa	1GZQKjsC97yasxRj1wtYf5rC61AxpR1zmr	0.00012000
ffffffffffffffffffffffffffffffffffffffff	1QLbz7JHiBTspS962RLKV8GndWFwi5j6Qr	0.01000005
151 miscellaneous ASCII HASH160 values		1.32340175

public key	valid encoding	point on curve	Bitcoin address	balance in BTC
0	x	x	1HT7xU2Ngenf7D4yocz2SAcnNLW7rK8d4E	68.80080003
00	✓	✓	1FYMZEHnszCHKTBdFZ2DLrUuk3dGwYKQxh	2.08000002
0^{128}	x	x	13VmALKHkCdSN1JULkP6RqW3LcbpWvgryV	0.00010000
040^{126}	✓	x	16QaFeudRUt8NYy2yzjm3BMvG4xBbAsBFM	0.01000000

Εικόνα 3.15

Κάποιοι έχουν χρησιμοποιήσει μερικές φορές τις HASH160 τιμές για να ενσωματώσουν ένα ASCII encoded string σε μία ή περισσότερες HASH160 τιμές σε μια συναλλαγή. Το ASCII κωδικοποιεί 128 χαρακτήρες. Η πιθανότητα ένα ECDSA δημόσιο κλειδί να καταλήξει σε δεκαεξαδική μορφή της HASH160 περιέχοντας ASCII χαρακτήρες είναι μόνο 2^{-20}. Η βάση δεδομένων που έχουν εξάγει οι συγγραφείς περιέχει 53.019.716 HASH160 τιμές από τις οποίες οι 16.526.211 είναι μοναδικές. Οπότε αναμένεται να βρεθούν περίπου 16 έγκυρες Bitcoin διευθύνσεις με την τιμή του HASH160 να περιέχει μόνο ASCII χαρακτήρες.

Στην βάση δεδομένων βρέθηκαν τελικά 248 HASH160 τιμές που περιέχουν μόνο ASCII χαρακτήρες, από τις οποίες οι 180 είναι μοναδικές. Από αυτές οι 20 διευθύνσεις έχουν κάνει συναλλαγές οπότε και αντιστοιχούν σε έγκυρες Bitcoin διευθύνσεις. Από τις υπόλοιπες 160, οι 137 έχουν μη μηδενικό υπόλοιπο.

3.3.3 Περίεργες τιμές σε ECDSA δημόσια κλειδιά

Όπως και με τις HASH160 τιμές, έτσι και με τα δημόσια ECDSA κλειδιά, κάποιος θα μπορούσε να χρησιμοποιήσει περίεργες τιμές στην κατασκευή τους. Αρχικά ας δούμε ακριβώς πως είναι το format των ECDSA δημόσιων κλειδιών στην δεκαεξαδική τους μορφή που χρησιμοποιεί το Bitcoin. Ένα σημείο $P = (x, y)$ μπορεί αναπαρασταθεί ως εξής:

- Αν το P είναι το σημείο στο άπειρο τότε συμβολίζεται με το μονό byte 00.

- Αν το P είναι ασυμπίεστο σημείο τότε ξεκινάει με το byte 04 και ακολουθεί η x και y συντεταγμένη με 256 bit η κάθε μία $(04 \parallel x \parallel y)$. Οπότε για την αναπαράσταση κάθε σημείου χρησιμοποιούνται $2\lceil log_2(p)/8 \rceil + 1 = 65$ bytes.

- Ένα σημείο P συμπιέζεται υπολογίζοντας αρχικά το parity bit b της y συντεταγμένης ως $b = (y mod 2) + 2$ και μετατρέποντας το σε byte τιμή $(b \in \{02, 03\})$. Το συμπιεσμένο σημείο έχει $\lceil log_2(p)/8 \rceil + 1 = 33$ bytes και γράφεται ως $b \parallel x$.

Όπως ακριβώς ξεκίνησε η έρευνα μας για τις HASH160 τιμές έτσι και εδώ θα αρχίσουμε να ψάχνουμε για σημεία που κωδικοποιούν μια μικρή ακέραια τιμή. Δημιουργήθηκαν αρχικά όλες οι Bitcoin διευθύνσεις οι οποίες αντιστοιχούν σε δημόσια κλειδιά με τιμές τους πρώτους 256 ακέραιους i, $0 \leq i < 256$ και με δίφορες τιμές για το parity bit. Χρησιμοποιήθηκε ένα μονό byte που περιέχει το i, μια τιμή 33 byte $b \parallel 0 \overbrace{\ldots}^{60} 0 \parallel i$, όπου $b \in \{00, 02, 03\}$ και μια τιμή 65 byte $b \parallel 0 \overbrace{\ldots}^{124} 0 \parallel i$, όπου $b \in \{00, 04\}$. Βρέθηκαν 3 διευθύνσεις με μη μηδενικό υπόλοιπο, το μονό byte 00 και το 65 byte $b \parallel 0 \overbrace{\ldots}^{124} 0 \parallel i$ με $i = 00$ και $b \in \{00, 04\}$. Το πρώτο σημείο είναι το σημείο στο άπειρο, το οποίο είναι έγκυρο σημείο στην καμπύλη και με σωστή κωδικοποίηση. Όμως η τιμή αυτή δεν επιτρέπετε στα δημόσια κλειδιά. Οι 65 byte τιμές φαίνεται ότι προσπαθούν να κωδικοποιήσουν το σημείο στο άπειρο, όπου στην περίπτωση που το $b = 00$ η κωδικοποίηση δεν είναι έγκυρη ενώ στην περίπτωση που το $b = 04$ η κωδικοποίηση είναι έγκυρη αλλά το σημείο $(x, y) = (0, 0)$ δεν είναι στην καμπύλη.

Ψάχνοντας και για άλλες τιμές δοκιμάστηκε και το κενό για δημόσιο κλειδί $(\emptyset)$. Η διεύθυνση αυτή περιέχει περίπου 68 Bitcoins. Υποθέτουμε ότι οι μεταφορές αυτές έγιναν λόγο bugs των προγραμμάτων. Τα αποτελέσματα αυτά φαίνονται επίσης στην εικόνα 3.15. Συνολικά βρέθηκαν περίπου 75 Bitcoins τα οποία έχουν μεταφερθεί σε λογαριασμούς που δεν έχουν έγκυρα ECDSA ιδιωτικά κλειδιά.

3.4 Bitcoin Contracts

Στην ενότητα αυτή θα μιλήσουμε για τα Bitcoin Contracts και για την ασφάλεια τους. Θα περιγράψουμε μια malleability επίθεση στο πρωτόκολλο του Deposit και θα δούμε μια καινούργια τεχνική που αναπτύχθηκε από τους Marcin, Stefan, Daniel και Lukasz [2] η οποία κάνει τα πρωτόκολλα ανθεκτικά στις malleability επιθέσεις.

3.4.1 Το πρωτόκολλο Deposit

Το πρωτόκολλο Deposit [12] εκτελείται μεταξύ δύο χρηστών, τον Α και τον Β. Η βασική ιδέα του πρωτοκόλλου είναι να επιτρέψει στον χρήστη Α να δημιουργήσει μια κατάθεση αξίας d Bitcoins για κάποια χρονική περίοδο. Ο Α πρέπει να μπορεί να είναι σίγουρος ότι μετά το πέρας του χρόνου t θα πάρει τα χρήματα του πίσω και ο Β πρέπει να είναι σίγουρος ότι ο Α δεν θα διεκδικήσει τα χρήματα του νωρίτερα. Μια από τις εφαρμογές του πρωτοκόλλου είναι η περίπτωση που ο Β είναι ο server και ο Α είναι ο χρήστης ο οποίος θέλει να ανοίξει ένα λογαριασμό στον server Β. Στην περίπτωση αυτή ο Β θέλει να σιγουρευτεί ότι ο Α δεν είναι κάποιο robot το οποίο ανοίγει ψεύτικους λογαριασμούς και έτσι υποχρεώνει τον Α να κάνει μια μικρή κατάθεση για κάποιο χρονικό διάστημα. Για κάποιον τίμιο χρήστη αυτό δεν δημιουργεί πρόβλημα λόγο του ότι η κατάθεση αυτή είναι πολύ μικρή και ο χρήστης Α είναι σίγουρος ότι θα πάρει τα χρήματα του πίσω μετά το πέρας του χρόνου αυτού. Από την άλλη πλευρά, για κάποιον που δημιουργεί πολλούς ψεύτικους λογαριασμούς το ποσό των καταθέσεων θα είναι πολυδάπανο.

Θα περιγράψουμε τώρα το πρωτόκολλο Deposit. Ο Α καταθέτει τα χρήματα του χρησιμοποιώντας την συναλλαγή Deposit η οποία μπορεί να ξοδευτεί μόνο με τις υπογραφές των Α και Β. Για να είναι σίγουρος ο Α ότι θα πάρει τα χρήματα του πίσω δημιουργεί την συναλλαγή Fuse η οποία μπορεί να ξοδέψει την Deposit. Η συναλλαγή αυτή πρέπει να υπογραφτεί από τον Β, οπότε ο Α ζητάει από τον Β να την υπογράψει και μόνο όταν ο Α παραλάβει την υπογραφή αυτή δημοσιεύει την συναλλαγή Deposit στην Blockchain. Με σκοπό να αποτραπεί ο Α από την διεκδίκηση των χρημάτων νωρίτερα η συναλλαγή Fuse περιέχει ένα timelock t το οποίο την εμποδίζει να ξοδευτεί πριν το πέρας του χρόνου αυτού. Πιο αναλυτικά το πρωτόκολλο είναι το ακόλουθο:

1. Στην αρχή ο Α και ο Β ανταλλάσσουν τα δημόσια κλειδιά τους τα οποία χρησιμοποιούν για να υπογράφουν τις Bitcoin συναλλαγές τους και συμφωνούν για το μέγεθος της κατάθεσης d και για τον χρόνο t στον οποίον η κατάθεση θα απελευθερωθεί.

2. Ο Α δημιουργεί την συναλλαγή Deposit αξίας d Bitcoins αλλά δεν την δημοσιεύει ακόμα. Η συναλλαγή είναι κατασκευασμένη με τέτοιο τρόπο έτσι ώστε να μην μπορεί να ξοδευτεί χωρίς τις υπογραφές του Α και Β.

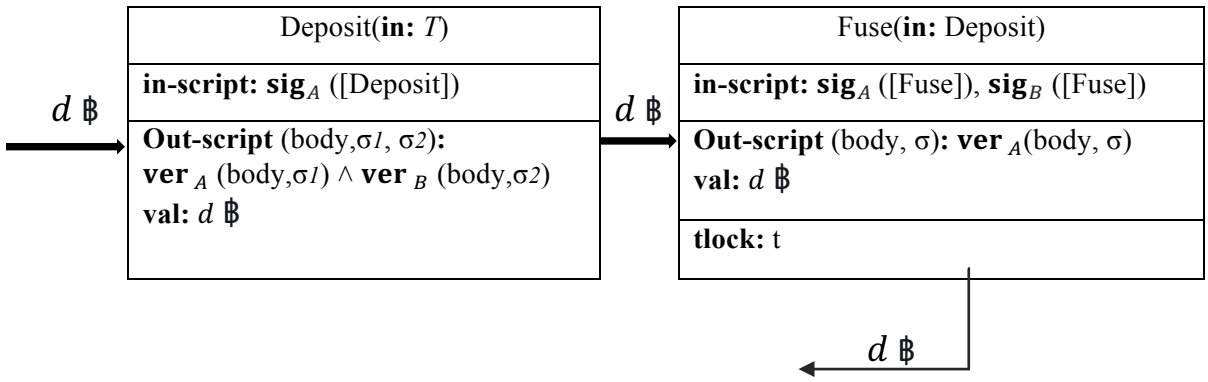
3. Στην συνέχεια ο Α δημιουργεί την συναλλαγή Fuse η οποία περιέχει ένα timelock t και ξοδεύει την συναλλαγή Deposit στέλνοντας τα χρήματα πίσω σε αυτόν. Η συναλλαγή δεν δημοσιεύεται ακόμα.

4. Ο Α στέλνει την συναλλαγή Fuse στον Β ο οποίος την υπογράφει και την στέλνει πίσω στον Α.

5. Μόνο τότε ο Α δημοσιεύει την συναλλαγή Deposit στην Blockchain.

6. Μετά το πέρας του χρόνου t ο Α δημοσιεύει την συναλλαγή Fuse και έτσι παίρνει πίσω τα χρήματα που είχε καταθέσει.

Στην εικόνα 3.16 βλέπουμε τις συναλλαγές του πρωτόκολλου Deposit.



Εικόνα 3.16

Οι ιδιότητες της ασφάλειας είναι οι ακόλουθες:

1. Ο Α δεν μπορεί να πάρει πίσω τα χρήματα του πριν το πέρας του χρόνου t, υποθέτοντας ότι ο Β ακολουθεί το πρωτόκολλο.

2. Ο Α δεν θα χάσει ποτέ τα χρήματα του αφού θα τα πάρει μετά το πέρας του χρόνου $(t + \Delta)$. Οπού με Δ συμβολίζουμε το χρόνο που μεσολαβεί από την δημοσίευση της συναλλαγής μέχρι την επιβεβαίωση της.

Είναι εύκολο να δούμε ότι η ιδιότητα (1) ισχύει, αφού δεν υπάρχει τρόπος να ξοδευτεί η συναλλαγή Deposit πριν το πέρας του χρόνου t αφού για να γίνει αυτό χρειάζεται και η υπογραφή του Β. Θα μπορούσαμε να πούμε ότι και η ιδιότητα (2) ισχύει αφού ο Α μπορεί να πάρει πίσω τα χρήματα του δημοσιεύοντας την συναλλαγή Fuse στην Blockchain. Η ιδιότητα αυτή βασίζεται στο γεγονός ότι η συναλλαγή Deposit έχει δημοσιευτεί στην Blockchain με την ίδια ακριβώς εκδοχή με την οποία η συναλλαγή Fuse δημιουργήθηκε. Αν λοιπόν κάποιος επιτιθέμενος τροποποιήσει την συναλλαγή Deposit κάνοντας χρήση της Malleability επίθεσης και δημοσιεύσει την Deposit’ τότε η Fuse συναλλαγή δεν θα μπορεί να ξοδευτεί αφού παίρνει ως είσοδο το hash της Deposit και όχι της Deposit’.

3.4.2 Το πρωτόκολλο Trading across chains

Εδώ θα περιγράψουμε το πρωτόκολλο Trading across chains και θα δούμε πως αυτό είναι ευάλωτο στις Malleability επιθέσεις. Η τεχνολογία του Bitcoin μπορεί να χρησιμοποιηθεί για την δημιουργία πολλών ανεξάρτητων νομισμάτων. Το NameCoin είναι ένα παράδειγμα τέτοιου νομίσματος το οποίο λειτουργεί με λίγο διαφορετικούς κανόνες και μπορεί να χρησιμοποιηθεί για την ενοικίαση ονομάτων (domain) στο διαδίκτυο. Τέτοιων ειδών νομίσματα τα οποία εφαρμόζουν την ίδια τεχνολογία με το Bitcoin πρέπει να μπορούν να συναλλάσσονται μεταξύ τους.

Για να εφαρμοστεί η ιδέα αυτή ο Tier Nolan έχει προτείνει το ακόλουθο πρωτόκολλο:

1. Ο χρήστης Α παράγει ένα τυχαίο δεδομένο x, το οποίο είναι το μυστικό.

2. Ο χρήστης Α παράγει την συναλλαγή Tx1 (πληρωμή), η οποία περιέχει μια έξοδο με το chain-trade script. Το script αυτό επιτρέπει την απελευθέρωση νομισμάτων είτε με τις υπογραφές των Α και Β (το κλειδί του Α και το κλειδί του Β), είτε με το μυστικό x του Α και το κλειδί του Β. Η συναλλαγή δεν δημοσιεύεται ακόμα. Το chain release script περιέχει τα hashes και όχι τα ίδια τα μυστικά.

3. Ο χρήστης Α παράγει την συναλλαγή Tx2 (συμβόλαιο), η οποία ξοδεύει την Tx1 και έχει ως έξοδο το δημόσιο κλειδί του Α. Έχει ένα locktime οπότε δεν μπορεί να ξοδευτεί πριν το πέρας του χρόνου αυτού. Ο Α υπογράφει την Tx2 και την στέλνει στον Β, ο οποίος την υπογράφει και αυτός και την στέλνει πίσω.

4. Ο Α δημοσιεύει την Tx1 και Tx2. Ο χρήστης Β μπορεί να δει τα νομίσματα αλλά δεν μπορεί να τα ξοδέψει αφού δεν υπάρχει ακόμα κάποια έξοδος που να πηγαίνει στο δημόσιο κλειδί του Β και η συναλλαγή δεν έχει οριστικοποιηθεί ακόμα.

5. Ο Β ακολουθεί το ίδιο πρωτόκολλο αντίστροφα στην άλλη αλυσίδα. Το locktime του Β θα πρέπει να είναι αρκετά μεγαλύτερο από αυτό του Α. Και οι δύο πλευρές της συναλλαγής είναι πλέον σε αναμονή.

6. Από την στιγμή που ο Α γνωρίζει το μυστικό x, τότε μπορεί να διεκδικήσει τα χρήματα του άμεσα. Όμως ο Α στην διαδικασία αυτή αποκαλύπτει το μυστικό x στον Β, ο οποίος με την σειρά του το χρησιμοποιεί για να ολοκληρώσει την συναλλαγή.

Το πρωτόκολλο αυτό κάνει εφικτές τις συναλλαγές αυτόματα με peer-to-peer διαδικασία. Όμως το πρωτόκολλο αυτό είναι ευάλωτο στις Malleability επιθέσεις. Το πρόβλημα δημιουργείται στο βήμα 3 όταν ο χρήστης Α παράγει την συναλλαγή Tx2 (συμβόλαιο) η οποία ξοδεύει την Tx1 και στην συνέχεια ζητάει από τον Β να την υπογράψει και αυτός και να την στείλει πίσω. Αυτό συμβαίνει πριν η Tx1 συμπεριληφθεί στην Blockchain, επομένως αν κάποιος επιτιθέμενος τροποποιήσει την Tx1 και καταφέρει να συμπεριλάβει την Tx1’ στην Blockchain τότε η συναλλαγή Tx2 δεν θα μπορεί να ξοδευτεί.

3.4.3 Η καινούργια τεχνική

Εδώ θα δούμε μια καινούργια τεχνική που αναπτύχθηκε από τους Marcin, Stefan, Daniel και Lukasz [2] η οποία κάνει τα πρωτόκολλα του Bitcoin όπως το Deposit, το Trading across chains, το Rapidly-adjusted (micro)payments to a pre-determined party [12] και το Lottery πρωτόκολλο των Back and Bentov [5] ανθεκτικά στις Malleability επιθέσεις. Ο λόγος που τα πρωτόκολλα αυτά είναι ευάλωτα στην επίθεση αυτή είναι λόγο του ότι όταν ένας χρήστης Α πρέπει αποκτήσει μια υπογραφή από έναν χρήστη Β σε μια συναλλαγή Tx1 της οποίας η είσοδος είναι η συναλλαγή Tx0 και αυτό πρέπει να γίνει πριν η συναλλαγή Tx0 συμπεριληφθεί στην Blockchain. Η βασική ιδέα προκύπτει από την παρατήρηση των ιδιοτήτων της γλώσσας προγραμματισμού του Bitcoin, η οποία μας δίνει την δυνατότητα να αλλάξουμε το βήμα αυτό κάνοντας την Tx0 να μπορεί να ξοδευτεί χρησιμοποιώντας όχι την υπογραφή του Β αλλά μια preimage s κάποιας τιμής h μέσω κάποιας Hash function. Τέτοιου είδος συναλλαγές στο Bitcoin ονομάζονται Hash locked. Να σημειωθεί ότι όταν μια συναλλαγή έχει στην έξοδο της ένα output script το οποίο χρειάζεται μόνο preimages και όχι υπογραφές τότε δεν είναι ασφαλής, γιατί ο επιτιθέμενος θα μπορούσε να δει ότι υπάρχει στο δίκτυο μια συναλλαγή με τέτοια έξοδο και να μάθει τις preimages με αποτέλεσμα να προσπαθήσει να εξαργυρώσει ο ίδιος την συναλλαγή. Στην δικιά μας περίπτωση όμως η συναλλαγή μας θα χρειάζεται και την υπογραφή του Α για να εξαργυρωθεί.

Ας δούμε τώρα πιο αναλυτικά τις προϋποθέσεις που χρειάζονται για να εξαργυρωθεί η συναλλαγή Tx0. Η έξοδος της συναλλαγής είναι η ακόλουθη:

$$out - script(body, \dots, \sigma): \dots \wedge ver_B(body, \sigma)$$

και εμείς θα την αντικαταστήσουμε με την ακόλουθη:

$$out - script(body, \dots, x): \dots \wedge H(x) = h$$

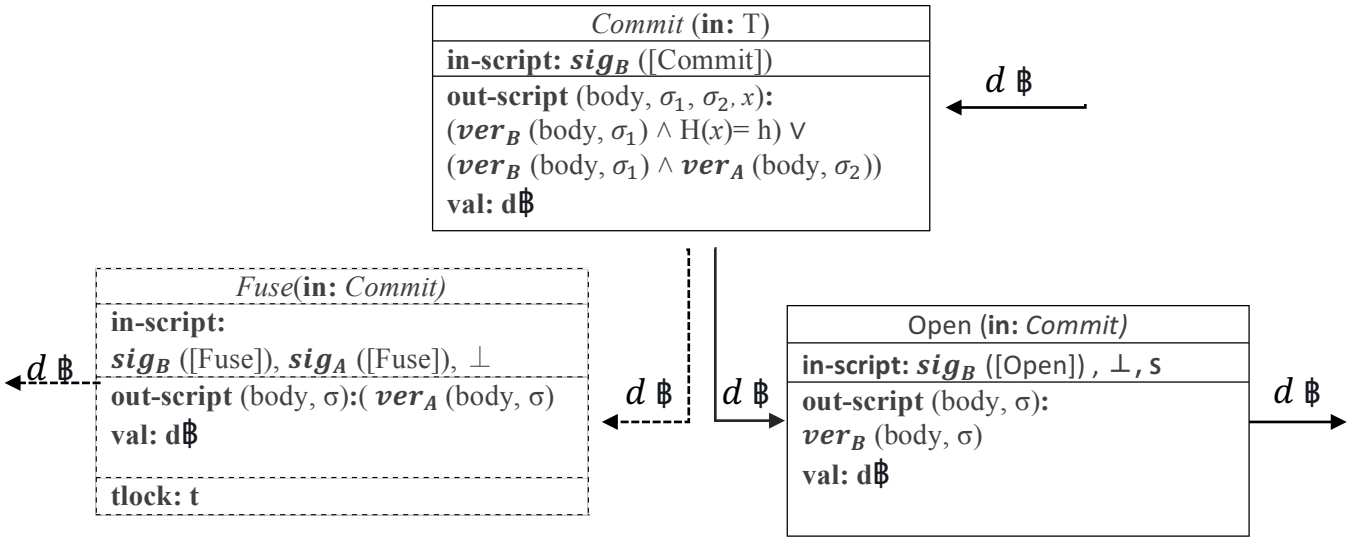
όπου το $h = H(s)$ κοινοποιείται από τον B στον A και το s το έχει διαλέξει ο B τυχαία. Αυτό θα επιτρέψει στον A να μπορεί να ξοδέψει την Tx0 ακόμα και αν αυτή έχει τροποποιηθεί από κάποιον επιτιθέμενο, εάν και εφόσον ο A μάθει το s. Όμως αρχικά βλέπουμε ότι δεν υπάρχει κάποιος σίγουρος τρόπος ο B να υποχρεωθεί να στείλει το s στον A, αφού αυτό θα πρέπει να γίνει αφού η Tx0 συμπεριληφθεί στην Blockchain αλλιώς αν ο A είχε το s νωρίτερα θα μπορούσε να ξοδέψει αμέσως την Tx0. Το πρόβλημα αυτό λύνεται προσθέτοντας ακόμα ένα στοιχείο στο πρωτόκολλο. Θα χρησιμοποιήσουμε το πρωτόκολλο Bitcoin-based timed commitment scheme το οποίο και θα δούμε πιο αναλυτικά στην συνέχεια και κάνει ακριβώς ότι χρειαζόμαστε. Επιτρέπει στον ένα χρήστη που ονομάζουμε Committer (ο B στην δική μας περίπτωση) να κάνει commit (να δεσμευτεί) σε ένα string s στέλνοντας το $h = H(s)$ στον παραλήπτη που ονομάζουμε Recipient (ο A στην δική μας περίπτωση). Στην συνέχεια ο B μπορεί να ανοίξει το commitment στέλνοντας το s στον A, ο οποίος μέχρι εκείνη την στιγμή δεν γνωρίζει το s. Η καλή ιδιότητα του σχήματος αυτού είναι ότι οι χρήστες μπορούν να προσδιορίσουν τον χρόνο t στον οποίο ο B πρέπει να ανοίξει το commitment. Αν δεν το κάνει αυτό μετά το πέρας του χρόνου t τότε είναι υποχρεωμένος να πληρώσει πρόστιμο σε Bitcoins στον A. Όπως θα δείξουμε και στην συνέχεια το Bitcoin-based timed commitment scheme είναι ασφαλές από τις Malleability επιθέσεις.

3.4.4 Το πρωτόκολλο Bitcoin-based timed commitment scheme

Το πρωτόκολλο αυτό έχει προταθεί από τους Marcin, Stefan, Daniel και Lukasz [2] και η κατασκευή του έχει βασιστεί στο πρωτόκολλο του coin-tossing του Blum. Κατάφεραν να προσαρμόσουν το πρωτόκολλο αυτό χωρίς να χρειάζεται κάποια αλλαγή στο σύστημα του Bitcoin. Τα κρυπτογραφικά εργαλεία που χρησιμοποίησαν είναι τα commitment σχήματα και οι hash functions. Το πρωτόκολλο είναι βασισμένο στις ιδιότητες του Bitcoin.

Το Bitcoin-based timed commitment scheme πρωτόκολλο (CS) εκτελείται μεταξύ του Committer B και του Recipient A. Κατά τη διάρκεια της φάσης της δέσμευσης ο Committer δεσμεύεται σε κάποιο string s αποκαλύπτοντας το hash του $h = H(s)$. Επιπλέον οι χρήστες συμφωνούν την χρονική στιγμή t στην οποία ο Committer θα πρέπει να ανοίξει την δέσμευσή του αποκαλύπτοντας το μυστικό s. Το πρωτόκολλο είναι κατασκευασμένο με τέτοιο τρόπο έτσι ώστε αν ο Committer δεν ανοίξει την δέσμευσή του την χρονική στιγμή t, τότε το συμφωνημένο ποσό των d Bitcoins μεταφέρεται αυτόματα από τον Committer στον Recipient.

Πιο συγκεκριμένα, ο Committer στην αρχή του πρωτοκόλλου κάνει μια κατάθεση d Bitcoins, η οποία του επιστέφεται αν ανοίξει την δέσμευσή του την χρονική στιγμή t, αλλιώς το ποσό αυτό πηγαίνει στον Recipient. Τα γραφήματα των συναλλαγών φαίνονται στην εικόνα 3.17.



Εικόνα 3.17

Στην συνέχεια ακολουθεί η πλήρη περιγραφή του πρωτοκόλλου:

Προ-Απαιτούμενες Συνθήκες:

1. Το πρωτόκολλο εκτελείται μεταξύ του Committer B, ο οποίος έχει το ζεύγος κλειδιών B και του Recipient A, ο οποίος έχει το ζεύγος κλειδιών A.

2. Ο Committer γνωρίζει το μυστικό string s.

3. Η Blockchain περιέχει μια μη-εξαργυρωμένη συναλλαγή T αξίας d Bitcoins, η οποία μπορεί να εξαργυρωθεί με το κλειδί B.

Η $CS.Commit\ (B, A, d, t, s)$ φάση:

1. Ο Committer υπολογίζει $h = H(s)$ και δημοσιεύει την συναλλαγή Commit. Αυτό σημαίνει ότι αποκαλύπτει το h αφού αποτελεί μέρος της συναλλαγής Commit.

2. Ο Committer περιμένει να εγκριθεί η συναλλαγή Commit και στην συνέχεια δημιουργεί το body της συναλλαγής Fuse, το υπογράφει και στέλνει την υπογραφή στον Recipient.

3. Αν ο Recipient δεν παραλάβει την υπογραφή ή είναι λάθος είτε η υπογραφή είτε η συναλλαγή Commit τότε παρατάει το πρωτόκολλο.

Η $CS.Open\ (B, A, d, t, s)$ φάση:

1. Όχι αργότερα από το πέρας της χρονικής στιγμής $(t - \Delta)$ ο Committer δημοσιεύει την συναλλαγή Open, στην οποία αποκαλύπτει το μυστικό s.

2. Αν μετά το πέρας του χρόνου t η συναλλαγή Open δεν εμφανιστεί στην Blockchain, τότε ο Recipient δημοσιεύει την συναλλαγή Fuse για να κερδίσει το ποσό των d Bitcoins.

Η κεντρική ιδέα είναι ότι αν ο Committer είναι τίμιος τότε δεν θα χρειαστεί η Fuse συναλλαγή παρά μόνο οι συναλλαγές Commit και Open. Αν παρόλα αυτά ο Committer δεν ανοίξει την δέσμευσή του, τότε ο Recipient θα δημοσιεύσει την συναλλαγή Fuse και θα απαιτήσει τα d Bitcoins του B. Η Fuse συναλλαγή περιέχει ένα time-lock, οπότε ένας κακόβουλος Recipient δεν μπορεί να απαιτήσει νωρίτερα από τον χρόνο t το ποσό των χρημάτων και ακόμα μετά το πέρας του χρόνου αυτού θα πρέπει ο B να μην έχει ανοίξει την δέσμευσή του για να μπορέσει να το κάνει. Ακόμα και αν η συναλλαγή Commit έχει τροποποιηθεί κακόβουλα πριν συμπεριληφθεί στην Blockchain, το πρωτόκολλο θα είναι επιτυχές γιατί η συναλλαγή Fuse έχει δημιουργηθεί μετά την συμπερίληψη της Commit στην Blockchain οπότε θα περιέχει σίγουρα το σωστό Hash του Commit. Με τον τρόπο αυτό το CS πρωτόκολλο είναι ανθεκτικό στις Malleability επιθέσεις.

Στην συνέχεια ακολουθούν οι ιδιότητες του CS πρωτοκόλλου:

a. Ο Recipient δεν έχει καμία πληροφορία για το μυστικό s πριν την δημοσίευση της συναλλαγής Open από τον Committer. Η ιδιότητα αυτή ονομάζεται hiding.

b. Ο Committer δεν μπορεί να ανοίξει την δέσμευσή του με κάποιον άλλο τρόπο παρά μόνο αποκαλύπτοντας το μυστικό s. Η ιδιότητα αυτή ονομάζεται binding.

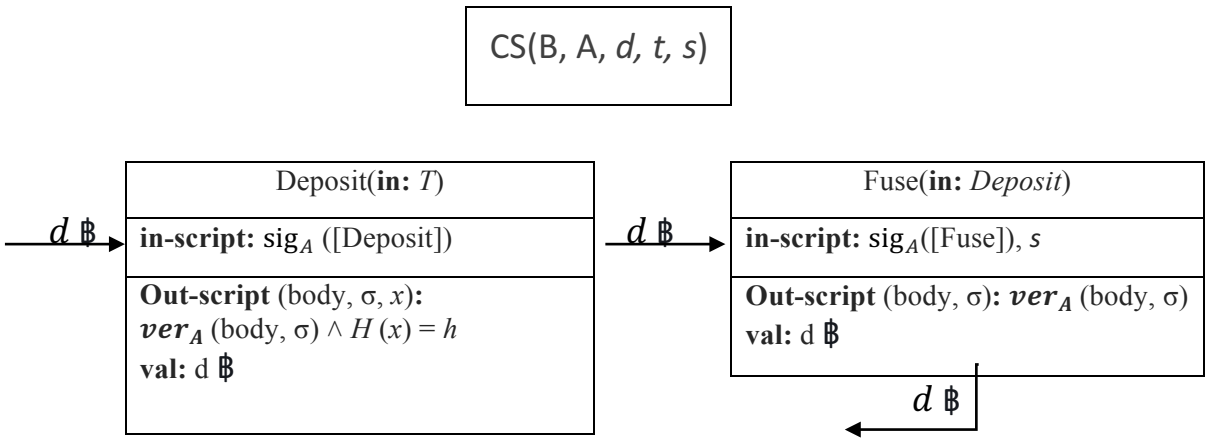
c. Ο τίμιος Committer δεν θα χάσει ποτέ την κατάθεση του και θα την πάρει πίσω μετά το πέρας του χρόνου t.

d. Αν ο Committer δεν αποκαλύψει το μυστικό s πριν το πέρας του χρόνου $(t + \Delta)$ τότε ο Recipient θα πάρει d Bitcoins ως αποζημίωση.

3.4.5 Η Λύση του Malleability προβλήματος για τα Bitcoin Contracts

Τώρα θα δούμε πιο αναλυτικά την λύση του Malleability προβλήματος για τα Bitcoin contracts και την εφαρμογή του στο πρωτόκολλο Deposit. Η λύση μπορεί να εφαρμοστεί σε όλα τα Bitcoin contracts τα οποία είναι ευάλωτα σε Malleability επιθέσεις.

Η βασική ιδέα είναι η χρήση του CS πρωτοκόλλου στην θέση των Fuse συναλλαγών. Στην αρχή του πρωτοκόλλου ο B διαλέγει το τυχαίο μυστικό s και δεσμεύεται σε αυτό στην φάση Commit του CS πρωτοκόλλου. Τώρα ο A γνωρίζει ότι ο B θα πρέπει να αποκαλύψει το μυστικό s μετά το πέρας του χρόνου t, έτσι ο A δημιουργεί την Deposit συναλλαγή με τέτοιο τρόπο που για να ξοδευτεί θα πρέπει να χρησιμοποιηθεί η υπογραφή του και η τιμή του μυστικού s. Οπότε μετά το πέρας του χρόνου t είτε ο B θα αποκαλύψει την τιμή s και ο A θα μπορεί να ξοδέψει την συναλλαγή Deposit ή ο A θα πάρει την κατάθεση του B από το πρωτόκολλο CS. Το πρωτόκολλο ονομάζεται NewDeposit και το γράφημα των συναλλαγών του φαίνεται στην εικόνα 3.18.



Εικόνα 3.18

Στην συνέχεια ακολουθεί η πλήρη περιγραφή του πρωτοκόλλου:

Προ-Απαιτούμενες Συνθήκες:

1. Ο καθένας από τους A και B έχει το αντίστοιχο ζεύγος κλειδιών.

Η φάση της δέσμευσης:

1. Ο B διαλέγει το τυχαίο μυστικό s.

2. Οι χρήστες εκτελούν το πρωτόκολλο $CS.Commit\ (B, A, d, t, s)$.

3. Ο A δημιουργεί την συναλλαγή Deposit χρησιμοποιώντας την τιμή h από την συναλλαγή $CS(B, A, d, t, s).Commit$ και την δημοσιεύει.

Η φάση του ανοίγματος της δέσμευσης:

1. Ο B ανοίγει την δέσμευσή του αποκαλύπτοντας το s την χρονική στιγμή $(t - \Delta)$.

2. Ο A μαθαίνει το μυστικό s και δημοσιεύει την συναλλαγή Fuse.

3. Αν ο B δεν ανοίξει την δέσμευσή του μέχρι την χρονική στιγμή t τότε ο A δημοσιεύει την συναλλαγή $CS(B, A, d, t, s).Fuse$ για να κερδίσει το ποσό των d Bitcoins.

Οι ιδιότητες του πρωτοκόλλου NewDeposit είναι οι ακόλουθες:

a. Ο A δεν μπορεί να πάρει πίσω την κατάθεσή του από την συναλλαγή Deposit πριν την χρονική στιγμή $(t - \Delta)$.

b. Ο τίμιος A δεν θα χάσει ποτέ την κατάθεση του και θα πάρει d Bitcoins πίσω πριν την χρονική στιγμή $(t + 2\Delta)$.

c. Ο τίμιος B αντίστοιχα δεν θα χάσει και αυτός ποτέ την κατάθεση του και θα την πάρει πίσω πριν την χρονική στιγμή t.

3.5 Secure Multiparty Computations στο Bitcoin

Τα Secure Multiparty Computation πρωτόκολλα (MPC) επιτρέπουν σε μια ομάδα από μέλη τα οποία δεν εμπιστεύεται το ένα το άλλο να υπολογίσουν μια κοινή συνάρτηση f με τις μυστικές τους εισόδους. Η ασφάλεια τέτοιων πρωτοκόλλων ορίζεται στο ιδανικό μοντέλο στο οποίο η f υπολογίζεται από ένα έμπιστο μέλος T_f. Κατά την διάρκεια της εκτέλεσης του πρωτοκόλλου τα μέλη δεν μπορούν να μάθουν πληροφορίες για τις εισόδους των άλλων μελών. Υποθέτουμε ότι η f υπολογίζεται από την T_f η οποία παραλαμβάνει τις εισόδους όλων των μελών, υπολογίζει την f και στέλνει την έξοδό της πίσω στα μέλη του πρωτοκόλλου. Επιπλέον ακόμα και αν κάποια μέλη δεν είναι τίμια και δεν ακολουθούν το πρωτόκολλο τότε αυτό δεν θα μπορεί να επηρεάσει την έξοδο των τίμιων μελών. Το μόνο που θα μπορούσαν τα κακόβουλα αυτά μέλη να κάνουν είναι να τροποποιήσουν τις δικές τους εισόδους και όχι των άλλων μελών.

Ας δούμε ένα παράδειγμα το οποίο ονομάζεται το πρόβλημα της πρότασης γάμου. Στην περίπτωση αυτή υπάρχουν δύο μέλη η Alice και ο Bob και η συνάρτηση που υπολογίζουν είναι μια σύζευξη $f_\wedge(a,b) = a \wedge b$ όπου $a,b \in \{0,1\}$ είναι Boolean μεταβλητές οι οποίες αντιστοιχούν στις εισόδους των Alice και Bob. Υποθέτουμε ότι η είσοδος του κάθε μέλους είναι η δήλωση αν αυτός ή αυτή θέλει να παντρευτεί τον άλλο ή την άλλη. Πιο συγκεκριμένα έστω ότι το $a = 1$ αν και μόνο αν η Alice θέλει να παντρευτεί τον Bob και $b = 1$ αν και μόνο αν ο Bob θέλει να παντρευτεί την Alice. Στην περίπτωση αυτή έχουμε $f_\wedge(a,b) = 1$ αν και μόνο αν και οι δύο θέλουν να παντρευτούν μεταξύ τους και αν για παράδειγμα είχαμε $b = 0$ τότε ο Bob δεν θα είχε κάποια πληροφορία για την είσοδο της Alice.

Έχει αποδειχτεί στο [37] ότι για οποιαδήποτε αποδοτικά υπολογίσιμη συνάρτηση f υπάρχει ένα αποδοτικό πρωτόκολλο το οποίο την υπολογίζει με ασφάλεια. Αν η μειονότητα των μελών του πρωτοκόλλου είναι κακόβουλοι και δεν ακολουθούν το πρωτόκολλο τότε το πρωτόκολλο πάντα θα τερματίζει και η έξοδος θα είναι γνωστή σε όλα τα τίμια μέλη. Αν τώρα τα κακόβουλα μέλη είναι περισσότερα τότε αυτά θα μπορούν να τερματίσουν το πρωτόκολλο μόλις μάθουν την έξοδο και να εμποδίσουν τα τίμια μέλη να την μάθουν και αυτά. Το πρόβλημα αυτό ονομάζεται lack of fairness [24] και είναι αναπόφευκτο. Έχουν γίνει κάποιες προσπάθειες να ξεπεραστεί αυτό το πρόβλημα χαλαρώνοντας τις απαιτήσεις ασφάλειας στα [39], [20] και [8]. Στα πρωτόκολλα δύο παιχτών δεν έχει νόημα να πούμε ότι η πλειοψηφία των παιχτών είναι τίμιοι αφού αυτό προϋποθέτει να είναι και οι δύο τίμιοι, οπότε στα πρωτόκολλα αυτά δεν προσφέρεται complete fairness εκτός και αν χαλαρώσει ο ορισμός της ασφάλειας.

Αρκετές προσπάθειες έχουν γίνει στα MPCs πρωτόκολλα για να γίνουν πιο αποδοτικά στα [47], [4] και [27]. Έχουν επίσης χρησιμοποιηθεί και σε εφαρμογές όπως online auctions [14]. Δεν έχουν όμως ακόμα χρησιμοποιηθεί σε εφαρμογές όπως το internet gambling παρόλο που θα ταίριαζαν στον τομέα αυτό. Προς το παρόν το internet gambling γίνεται από websites τα οποία παίζουν το ρόλο του "trusted party" και από την άποψη της ασφάλειας αυτό δημιουργεί πρόβλημα αφού οι διαχειριστές των websites είναι σε πλεονεκτική θέση και την θέση αυτή την χρησιμοποιούν για προσωπικό τους όφελος [64]. Οπότε βλέπουμε ότι θα ήταν ιδανικό να υπήρχαν κάποιες multiparty τεχνικές οι οποίες δεν θα χρειαζόντουσαν κάποιο "trusted party" για να λειτουργήσουν και θα μπορούσαν να αντικαταστήσουν τα παραδοσιακά gambling sites.

Οι βασικοί λόγοι οι οποίοι δεν επιτρέπουν στα MPCs πρωτόκολλα να χρησιμοποιηθούν σε online gambling είναι δύο. Ο πρώτος είναι γιατί τα πρωτόκολλα αυτά δεν έχουν fairness στην περίπτωση που η πλειοψηφία των παιχτών είναι κακόβουλοι και είναι ευάλωτα στις Sybil επιθέσεις [32] όπου ένα κακόβουλο μέλος θα μπορούσε να δημιουργήσει και να ελέγχει πολλούς ψεύτικους λογαριασμούς αποκτώντας έτσι την πλειοψηφία ανάμεσα στα μέλη και ο δεύτερος λόγος προέρχεται από τον ορισμό ασφάλειας των MPCs ο οποίος δεν παρέχει ασφάλεια παρά μόνο αν υπάρχει κάποιο "trusted party".

Την λύση μας την δίνει το Bitcoin το οποίο δεν χρειάζεται κάποιο "trusted party" για να το ελέγχει και έτσι κανένας κακόβουλος χρήστης δεν θα μπορούσε να πάρει τον έλεγχο του συστήματος και να δημιουργήσει μεγάλες ποσότητες νομισμάτων ή να κλείσει το σύστημα. Τα χρήματα μεταφέρονται απευθείας μεταξύ των χρηστών και δεν χρειάζεται να υπάρχει εμπιστοσύνη. Επίσης έχουμε και ανωνυμία μεταξύ των χρηστών.

Στην ενότητα αυτή θα δούμε την πρόταση των Marcin, Stefan, Daniel και Lukasz [3] οι οποίοι χρησιμοποιούν το Bitcoin-based timed commitment scheme πρωτόκολλο για να πετύχουν fairness σε κάποια MPC πρωτόκολλα. Κατασκεύασαν πρωτόκολλα για secure multiparty lotteries χρησιμοποιώντας το Bitcoin και χωρίς την χρήση κάποιου "trusted party". Με τον όρο lottery εννοούμε ένα πρωτόκολλο στο οποίο μια ομάδα από χρήστες αρχικά επενδύει κάποια χρήματα και στο τέλος του πρωτοκόλλου με τυχαία επιλογή ένας από τους χρήστες θα πάρει όλα τα χρήματα (τα χρήματα αυτά ονομάζονται pot). Τα πρωτόκολλα δουλεύουν σε peer-to-peer περιβάλλον και μπορούν να εκτελεστούν μεταξύ χρηστών οι οποίοι είναι ανώνυμοι και δεν εμπιστεύεται ο ένας τον άλλον. Οι κατασκευές αυτές των πρωτοκόλλων έχουν μια δυνατή εγγύηση ασφάλειας η οποία είναι ότι όπως και να φερθούν οι κακόβουλοι χρήστες, δεν θα μπορούν ποτέ να κλέψουν τους τίμιους. Κάθε τίμιος χρήστης θα μπορεί να είναι σίγουρος ότι μόλις το πρωτόκολλο ξεκινήσει τότε σίγουρα θα τερματίσει και είναι και δίκαιο.

3.5.1 Μοντέλο Ασφάλειας

Αρχικά θα περιγράψουμε το μοντέλο ασφάλειας που αντιστοιχεί στο σύστημα του Bitcoin. Υποθέτουμε ότι οι χρήστες είναι συνδεδεμένοι μέσω κάποιου καναλιού το οποίο είναι μη ασφαλές και έχουν πρόσβαση στην Blockchain του Bitcoin. Το πρωτόκολλο θα πρέπει να επιτρέπει σε οποιονδήποτε χρήστη να λάβει μέρος, επομένως δεν μπορούμε να υποθέσουμε ότι υπάρχει κάποια ασφαλής σύνδεση μεταξύ των χρηστών και οποιοδήποτε είδος man-in-the-middle επίθεσης είναι δυνατή.

Το μόνο έμπιστο μέλος είναι η Blockchain του Bitcoin. Θα υποθέσουμε στο μοντέλο μας ότι οι χρήστες έχουν πρόσβαση σε κάποιο "trusted third party" το οποίο ονομάζουμε Ledger και τα περιεχόμενα του είναι δημόσια διαθέσιμα. Αρχικά υποθέτουμε ότι οι χρήστες μπορούν να επιβεβαιώσουν την αυθεντικότητα του Ledger, δηλαδή κάθε χρήστης μπορεί να έχει πρόσβαση στα περιεχόμενα του Ledger και δημοσιεύει συναλλαγές σε αυτόν. Μετά την δημοσίευση της συναλλαγής, αν αυτή είναι έγκυρη, εμφανίζεται στον Ledger αλλά αυτό μπορεί να έχει κάποια χρονική καθυστέρηση. Συμβολίζουμε με max_{Ledger} το ανώτατο όριο της καθυστέρησης αυτής. Αυτό αντιστοιχεί στην υπόθεση ότι αργά ή γρήγορα κάθε συναλλαγή θα εμφανιστεί στην Blockchain. Μπορούμε να υποθέσουμε και ότι το $max_{Ledger} = 1\ day$ είναι αποδεκτό. Επίσης κάθε συναλλαγή που δημοσιεύεται στον Ledger έχει timestamp που αναφέρει την χρονική στιγμή που αυτή εμφανίστηκε στον Ledger.

Αυτό που είναι δύσκολο να ορίσουμε είναι η ασφαλή επικοινωνία μεταξύ των χρηστών και του Ledger. Δεν υποθέτουμε ότι οι συναλλαγές είναι κρυφές μέχρι την εμφάνιση τους στον ledger αφού αυτές μεταδίδονται μέσω των κόμβων του δικτύου και επιπλέον υπάρχει το Malleability πρόβλημα στις συναλλαγές που έχουμε ήδη εξηγήσει. Τα πρωτόκολλα που θα περιγράψουμε είναι όλα ασφαλή ακόμα και από τις malleability επιθέσεις. Υποθέτουμε ότι κάθε επικοινωνία μεταξύ των χρηστών και του Ledger είναι δημόσια γνωστή. Υποθέτουμε ότι η επικοινωνία μεταξύ των χρηστών δεν παίρνει χρόνο. Η ασφάλεια των πρωτοκόλλων δεν εξαρτάται από αυτές τις υποθέσεις και τα πρωτόκολλα είναι ασφαλή ακόμα και αν υπάρχει χρονική καθυστέρηση στην επικοινωνία μεταξύ των χρηστών. Τέλος υποθέτουμε ότι τα transaction fees είναι μηδενικά.

3.5.2 Το πρωτόκολλο Bitcoin-based timed commitment scheme

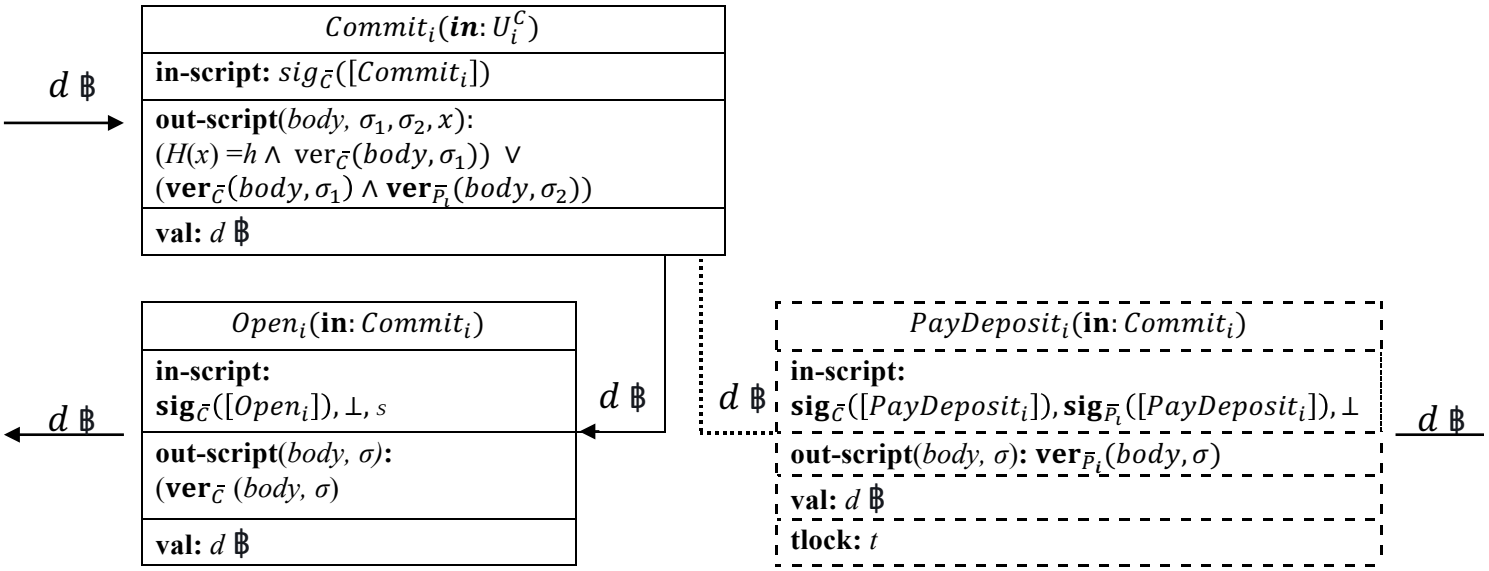
Τώρα θα ξαναδούμε το πρωτόκολλο Bitcoin-based timed commitment scheme στην γενικευμένη του μορφή στην οποία τα μέλη του πρωτοκόλλου αντί για 2 είναι πλέον $n + 1$. Ο Committer C και οι n παραλήπτες $P_1, P_2, \dots, P_n$. Ο Committer ξεκινάει το πρωτόκολλο με μια μυστική τιμή x και η τιμή αυτή γίνεται γνωστή στους παραλήπτες μετά την εκτέλεση της φάσης του ανοίγματος. Ισχύουν και εδώ οι ιδιότητες hiding και binding και ενώ στα συνηθισμένα Commitment schemes δεν υπάρχει κάποιος τρόπος που να υποχρεώνεται ο Committer να αποκαλύψει το μυστικό του x, με την βοήθεια του Bitcoin μπορούμε να το καταφέρουμε κάνοντας τον Committer να υποστηρίξει την δέσμευση του καταθέτοντας κάποιο χρηματικό ποσό, το οποίο ονομάζουμε Deposit και το οποίο πηγαίνει αυτόματα στα υπόλοιπα μέλη του πρωτοκόλλου στην περίπτωση που αρνηθεί να ανοίξει την δέσμευση του μετά το πέρας του χρόνου t.

Αρχικά υποθέτουμε ότι πριν το ξεκίνημα του πρωτοκόλλου ο Ledger περιέχει n μη εξαργυρωμένες συναλλαγές $U_1^C, \dots, U_n^C$ οι οποίες μπορούν να εξαργυρωθούν με ένα κλειδί που γνωρίζει μόνο ο C και η κάθε μια συναλλαγή έχει αξία d Bitcoins. Το πρωτόκολλο $CS(C, d, t, s)$ αποτελείται απο δύο φάσεις όπως έχουμε ήδη δει. Την φάση της δέσμευσης $CS.Commit(C, d, t, s)$ και την φάση του ανοίγματος της δέσμευσης $CS.Open(C, d, t, s)$. Υποθέτοντας ότι Committer είναι τίμιος τότε ο adversary δεν μαθαίνει κάποια πληροφορία για το x πριν την φάση του ανοίγματος και κάθε τίμιο μέλος μπορεί να είναι σίγουρο ότι υπάρχει το πολύ ένα x με το οποίο ο Committer μπορεί να ανοίξει την δέσμευση του στην φάση του ανοίγματος. Κάθε παραλήπτης μπορεί να εγκαταλείψει την φάση της δέσμευσης είτε γιατί ανακάλυψε ότι ο Committer κλέβει είτε γιατί ο Adversary διέκοψε την επικοινωνία. Επιπλέον, αν ο Committer δεν ανοίξει την δέσμευση του μέχρι την χρονική στιγμή t τότε τα υπόλοιπα μέλη κερδίζουν d Bitcoins. Για κάθε τίμιο P_i ο Ledger περιέχει μια συναλλαγή αξίας d Bitcoins η οποία μπορεί να εξαργυρωθεί μόνο με το κλειδί που γνωρίζει ο P_i.

Η υλοποίηση του πρωτοκόλλου μπορεί να βασιστεί σε οποιοδήποτε Commitment scheme αρκεί να είναι βασισμένο στις Hash functions, δηλαδή να έχει την ακόλουθη δομή. Έστω H μια Hash function. Κατά τη διάρκεια της φάσης της δέσμευσης ο Committer στέλνει στον παραλήπτη την τιμή h, η οποία υποδηλώνει την δέσμευση του στο x και στην φάση του ανοίγματος ο Committer στέλνει την τιμή s, έτσι ώστε $H(s) = h$. Αν $H(s) \neq h$ τότε ο παραλήπτης δεν αποδέχεται το άνοιγμα, αλλιώς υπολογίζει το x από το s. Ένα παράδειγμα ενός τέτοιου Commitment scheme είναι το ακόλουθο. Έστω $x \in \{0,1\}^*$. Στην φάση της δέσμευσης ο C υπολογίζει $s := (x \parallel r)$, όπου το r επιλέγεται τυχαία από το σύνολο $\{0,1\}^k$ και

στέλνει σε κάθε παραλήπτη $h = H(s)$. Στην φάση του ανοίγματος ο C στέλνει σε κάθε παραλήπτη την τιμή s έτσι ώστε να ελέγξει αν όντως $h = H(s)$ και ανακτά το x αφαιρώντας τα k τελευταία bits του s. Η binding ιδιότητα μας προσφέρεται από την ιδιότητα των Hash functions οι οποίες είναι ανθεκτικές στις συγκρούσεις, αφού για να ανοίξει με δύο διαφορετικούς τρόπους ένας κακόβουλος C την δέσμευση του θα πρέπει να βρει συγκρούσεις στην H. Για την hiding ιδιότητα υποθέτουμε ότι η H είναι ένα τυχαίο μαντείο και έτσι κανένας adversary δεν μπορεί να αποκτήσει πληροφορίες για το x χωρίς να μάθει το s, το οποίο ο τίμιος C το κρατάει κρυφό μέχρι την φάση του ανοίγματος.

Ο Committer θα μιλάει ανεξάρτητα με κάθε παραλήπτη P_i και για κάθε έναν θα δημιουργήσει στην φάση της δέσμευσης μια συναλλαγή $Commit_i$ αξίας d Bitcoins η οποία κανονικά θα εξαργυρωθεί από τον ίδιο στην φάση του ανοίγματος με την συναλλαγή $Open_i$. Η συναλλαγή $Commit_i$ θα κατασκευαστεί με τέτοιο τρόπο έτσι ώστε η συναλλαγή $Open_i$ να ανοίξει αυτόματα την δέσμευση. Αυτό θα γίνει κατασκευάζοντας το output script της $Commit_i$ με τέτοιο τρόπο έτσι ώστε η συναλλαγή που θα την εξαργυρώνει να παρέχει το s. Για να ορίσουμε τον χρόνο αναμονής του παραλήπτη, απαιτούμε από τον Committer να στείλει σε κάθε P_i την συναλλαγή $PayDeposit_i$ η οποία εξαργυρώνει την $Commit_i$ μετά το πέρας του χρόνου t. Ο κάθε παραλήπτης P_i θα πρέπει επίσης να ελέγξει αν η συναλλαγή $PayDeposit_i$ είναι σωστή. Στην εικόνα 3.19 βλέπουμε τα γραφήματα των συναλλαγών.



Εικόνα 3.19

Στην συνέχεια ακολουθεί η πλήρη περιγραφή του πρωτοκόλλου:

Προ-Απαιτούμενες Συνθήκες:

1. Το ζεύγος κλειδιών του C είναι το $\tilde{C}$ και το ζεύγος κλειδιών κάθε P_i είναι $\tilde{P_i}$.

2. Ο Ledger περιέχει n μη εξαργυρωμένες συναλλαγές $U_1^C, \dots, U_n^C$, οι οποίες μπορούν να εξαργυρωθούν με το κλειδί $\tilde{C}$ και κάθε μια έχει αξία d Bitcoins.

Η $CS.Commit(C, d, t, s)$ φάση:

1. Ο Committer C υπολογίζει $h = H(s)$ και στέλνει στον Ledger τις συναλλαγές $Commit_1, \dots, Commit_n$. Με τον τρόπο αυτό το h αποκαλύπτεται αφού περιέχεται στην συναλλαγή $Commit_i$.

2. Αν μετά το πέρας του χρόνου max_{Ledger} κάποιες από τις συναλλαγές $Commit_i$ δεν εμφανιστούν στον Ledger ή αν είναι τροποποιημένες, τότε τα μέλη εγκαταλείπουν το πρωτόκολλο.

3. Ο Committer δημιουργεί το body των συναλλαγών $PayDeposit_1, \dots, PayDeposit_n$, τις υπογράφει και για όλα τα i στέλνει το υπογεγραμμένο body $[PayDeposit_i]$ στον P_i. Αν η συναλλαγή δεν φτάσει ποτέ στον P_i τότε σταματάει το πρωτόκολλο.

Η $CS.Open(C, d, t, s)$ φάση:

1. Ο Committer C στέλνει στον Ledger τις συναλλαγές $Open_1, \dots, Open_n$ οι οποίες αποκαλύπτουν το μυστικό s.

2. Αν μετά το πέρας του χρόνου t η συναλλαγή $Open_i$ δεν εμφανιστεί στον Ledger, τότε ο P_i υπογράφει και στέλνει την συναλλαγή $PayDeposit_i$ στον Ledger και κερδίζει d Bitcoins.

3.5.3 Το πρωτόκολλο Lottery

Τώρα θα δούμε ένα MultiParty Computation πρωτόκολλο το οποίο χρησιμοποιεί το Bitcoin και δεν χρειάζεται κάποιο "Trusted Party" για να λειτουργήσει. Το πρωτόκολλο αυτό είναι μια λοταρία η οποία εκτελείται ανάμεσα σε μια ομάδα χρηστών $P_1, \dots, P_N$. Λέμε ότι ένα πρωτόκολλο λοταρίας είναι δίκαιο αν είναι ορθό και ασφαλές.

Για να ορίσουμε την ορθότητα υποθέτουμε ότι όλοι οι χρήστες ακολουθούν το πρωτόκολλο και τα κανάλια επικοινωνίας μεταξύ τους είναι ασφαλή. Αρχικά υποθέτουμε ότι πριν το ξεκίνημα του πρωτοκόλλου ο Ledger περιέχει τις μη εξαργυρωμένες συναλλαγές $T^1, \dots, T^N$ αξίας 1 Bitcoin η κάθε μια, οι οποίες είναι γνωστές σε όλους τους χρήστες και κάθε T^i συναλλαγή μπορεί να εξαργυρωθεί με το κλειδί που γνωρίζει μόνο ο P_i. Επίσης κάθε χρήστης θα πρέπει να μπορεί να πληρώσει τις αρχικές καταθέσεις που απαιτεί το πρωτόκολλο Bitcoin-based timed commitment scheme το οποίο και θα χρησιμοποιήσουμε. Για τις καταθέσεις αυτές έχουμε τις ακόλουθες συναλλαγές $\{U_j^i\}$, όπου $i, j \in \{1, \dots, N\}$ και $i \neq j$, όπου κάθε συναλλαγή U_j^i μπορεί να εξαργυρωθεί μόνο από τον P_i και έχει αξία d Bitcoins. Το πρωτόκολλο θα πρέπει να τερματίσει την χρονική στιγμή $O(max_{Ledger})$ και την στιγμή αυτή ο Ledger θα πρέπει να περιέχει μια συναλλαγή αξίας N Bitcoins, η οποία θα μπορεί να εξαργυρωθεί μόνο από τον P_w, όπου το w είναι τυχαία επιλογή από το σύνολο $\{1, \dots, N\}$. Επιπλέον ο Ledger περιέχει συναλλαγές οι οποίες επιστρέφουν τις καταθέσεις, για κάθε P_i υπάρχει μια συναλλαγή της οποίας η αξία είναι $(N-1)d$ Bitcoins. Υποθέτουμε ότι οι συναλλαγές έχουν μηδενικά fees.

Για να ορίσουμε την ασφάλεια θα δούμε την εκτέλεση του πρωτοκόλλου απο την πλευρά ενός μέλους της ομάδας των χρηστών του πρωτοκόλλου και αντίστοιχες είναι και οι άλλες περιπτώσεις. Έστω λοιπόν ότι το μέλος P_1 είναι τίμιο και ο Ledger περιέχει τις συναλλαγές $T^1, U_2^1, \dots, U_N^1$ των οποίων ο παραλήπτης είναι ο P_1 και η αξία τους είναι 1 Bitcoin για την T^1 και d Bitcoins για τις U_j^1. Ο P_1 δεν έχει κάποια εγγύηση ότι το πρωτόκολλο θα τερματίσει επιτυχώς, αφού κάποιο άλλο μέλος θα μπορούσε να εγκαταλείψει νωρίτερα. Το σημαντικό είναι ότι ο P_1 μπορεί να είναι σίγουρος ότι δεν θα χάσει χρήματα από αυτή την εγκατάλειψη του πρωτοκόλλου. Πιο συγκεκριμένα, ορίζουμε την πληρωμή του P_1 να είναι ίση με την διαφορά των χρημάτων που επένδυσε και των χρημάτων που κέρδισε. Άρα η πληρωμή του P_1 είναι ίση με $X_1 - ((N-1)d + 1)$ Bitcoins, όπου X_1 ορίζεται ως το σύνολο των αξιών των συναλλαγών από την εκτέλεση του πρωτοκόλλου συμπεριλαμβανομένου των συναλλαγών $T^1, U_2^1, \dots, U_N^1$ το οποίο ο P_1 μπορεί να εξαργυρώσει μετά τον τερματισμό του πρωτοκόλλου.

Ιδανικά θέλουμε η αναμενόμενη πληρωμή κάθε τίμιου μέλους να μην είναι αρνητική, όμως πρέπει να υπολογίσουμε και την αμελητέα πιθανότητα του Adversary ο οποίος καταφέρνει να σπάσει το πρωτόκολλο. Οπότε απαιτούμε οι τιμές αυτές να είναι τουλάχιστον αμελητέες για κάποια παράμετρο ασφάλειας k. Ορίζουμε ότι ένα πρωτόκολλο είναι ασφαλές αν λαμβάνοντας υπόψη οποιαδήποτε στρατηγική ακολουθήσει ο Adversary, ο οποίος ελέγχει το δίκτυο και μπορεί να διαφθείρει τα άλλα μέλη:

1. Η εκτέλεση του πρωτοκόλλου τερματίσει σε χρόνο $O(max_{Ledger})$

2. Η αναμενόμενη πληρωμή του κάθε τίμιου μέλους είναι τουλάχιστον αμελητέα

Στην περίπτωση που κάποιο κακόβουλο μέλος εγκαταλείψει σε κάποιο πρώιμο στάδιο του πρωτοκόλλου, τότε αυτό δημιουργεί πρόβλημα μόνο στην περίπτωση που τα fees των συναλλαγών δεν είναι μηδενικά.

Τώρα θα δούμε πιο αναλυτικά το πρωτόκολλο Lottery. Το πρωτόκολλο είναι βασισμένο στο coin-tossing του Blum το οποίο βασίζεται σε κρυπτογραφικές δεσμεύσεις. Το σχήμα του Blum μπορεί να προσαρμοστεί για N μέλη με τον ακόλουθο τρόπο. Κάθε μέλος P_i δεσμεύεται σε ένα $b_i \in Z_N$. Στην συνέχεια τα μέλη ανοίγουν τις δεσμεύσεις τους και ο νικητής είναι ο P_w όπου $w = (b_1 + \cdots + b_N mod\ N) + 1$. Προσαρμόζοντας το στην δική μας περίπτωση έχουμε ότι η δημιουργία και το άνοιγμα των δεσμεύσεων γίνεται από τα scripts των συναλλαγών με διπλή χρήση της SHA-256 Hash function. Επιπλέον κάθε μέλος αντί να δεσμεύεται σε κάποιο b_i, θα δεσμεύεται σε κάποιο string s_i το οποίο διαλέγεται τυχαία και με τυχαίο μήκος από το σύνολο $S_k^N = \{0,1\}^{8k} \cup \ldots \cup \{0,1\}^{8(k+N-1)}$ των strings τα οποία έχουν μήκος $k, \ldots, (k+N-1)$ bytes και k είναι η παράμετρος ασφάλειας. Ο νικητής καθορίζεται από την συνάρτηση επιλογής του νικητή $f(s_1, \ldots, s_N)$ και στην περίπτωσή μας έχουμε ότι $f(s_1, \ldots, s_N) = P_l$ αν $\sum_{i=1}^{N} |s_i| \equiv (l-1) mod\ N$ όπου $s_1, \ldots, s_N$ είναι τα μυστικά strings διαλεγμένα από το σύνολο S_k^N και $|s_i|$ είναι το μήκος του string s_i σε bytes. Τα τίμια μέλη αρχικά επιλέγουν το μήκος του string σε bytes από το σύνολο $\{k, \ldots, k+N-1\}$ και στην συνέχεια παράγουν το τυχαίο string. Εφόσον κάθε μέλος διαλέγει το μήκος του string του τυχαία τότε και η έξοδος της συνάρτησης $f(s_1, \ldots, s_N)$ είναι επίσης τυχαία.

Περιγραφή του πρωτοκόλλου Lottery

Αρχικά θα περιγράψουμε το πρωτόκολλο στην περίπτωση που τα μέλη είναι $N = 2$. Έστω A η Alice και B ο Bob με τα αντίστοιχα ζεύγη κλειδιών τους. Αρχικά ο Ledger θα περιέχει τις μη-εξαργυρωμένες συναλλαγές των Alice και Bob, T^A, U^A και T^B, U^B αντίστοιχα. Θα ξεκινήσουμε κάνοντας μια αρχική κατασκευή του πρωτοκόλλου και στην πορεία θα φτάσουμε στην τελική του μορφή. Το πρωτόκολλο ξεκινάει με την Alice και τον Bob να δημιουργούν τα κλειδιά τους $\tilde{A} = (\tilde{A}_{sk}, \tilde{A}_{pk})$ και $\tilde{B} = (\tilde{B}_{sk}, \tilde{B}_{pk})$ αντίστοιχα. Στην συνέχεια ανακοινώνουν μεταξύ τους τα δημόσια κλειδιά τους, επιλέγουν τα τυχαία μυστικά string τους s_A και s_B από το σύνολο S_k^2 και ανταλλάσσουν τα $h_A = H(s_A)$ και $h_B = H(s_B)$. Επιπλέον η Alice στέλνει στον Ledger την ακόλουθη συναλλαγή:

$PutMoney_1^A$ (**in**: T^A)
in-script: $\text{sig}_{\boldsymbol{A}}([\text{PutMoney}_1^A])$
out-script(body, σ):$ver_{\tilde{A}}$ (body, σ)
Val: 1 ฿

Αντίστοιχα ο Bob στέλνει και εκείνος στον Ledger την συναλλαγή $PutMoney_1^B$. Κατά την διάρκεια του πρωτοκόλλου αν κάποιο μέλος $P \in \{A, B\}$ αντιληφθεί ότι το άλλο κλέβει, τότε αυτό το μέλος θα δημοσιεύσει την συναλλαγή που εξαργυρώνει την $PutMoney_1^P$. Στο επόμενο βήμα κάποιο απο τα μέλη κατασκευάζει την ακόλουθη συναλλαγή $Compute_1$:

$Compute_1$(**in**: $PutMoney_1^A$,$PutMoney_1^B$)	
in $-$ **script**$_\mathbf{1}$:$sig_{\tilde{A}}$ [$(Compute_1)$]	**in** $-$ **script**$_\mathbf{2}$:$sig_{\tilde{B}}$ [$(Compute_1)$]
out-script(body,σ_1, σ_2,$\hat{s}_A$,$\hat{s}_B$): $(\hat{s}_A, \hat{s}_B \in S_k^2 \wedge H(\hat{s}_A) = h_A \wedge H(\hat{s}_B) = h_B$ $\wedge f(\hat{s}_A, \hat{s}_B) = A \wedge \text{ver}_{\tilde{A}}$ (body,σ_1)) $\vee$ $(\hat{s}_A, \hat{s}_B \in S_k^2 \wedge H(\hat{s}_A) = h_A \wedge H(\hat{s}_B) = h_B$ $\wedge f(\hat{s}_A, \hat{s}_B) = B \wedge \text{ver}_{\tilde{B}}$ (body,σ_2))	
val: 2 ฿	

Παρατηρούμε ότι το σώμα της συναλλαγής μπορεί να κατασκευαστεί από τις δημόσια διαθέσιμες πληροφορίες. Οπότε η κατασκευή αυτή θα γίνει με τον ακόλουθο τρόπο. Αρχικά ο Bob θα υπογράψει την συναλλαγή και θα στείλει την υπογραφή του $sig_{\tilde{B}}([Compute_1])$ στην Alice. Η Alice υπογράφει και αυτή και ολόκληρη η συναλλαγή δημοσιεύεται στον Ledger.

Το output script της συναλλαγής $Compute_1$ έχει εναλλακτικά δύο συνθήκες. Λόγω συμμετρίας αρκεί να δούμε μόνο την πρώτη συνθήκη και θα την ονομάσουμε γ. Για να αξιολογηθεί η συνθήκη αληθής θα πρέπει να έχει ως «μάρτυρες» τα $(\sigma_1, \hat{s}_A, \hat{s}_B)$, όπου τα $\hat{s}_A$ και $\hat{s}_B$ είναι οι αντίστροφες εικόνες των h_A και h_B της συνάρτησης H από το σύνολο S_k^2. Τα $\hat{s}_A$ και $\hat{s}_B$ είναι ίσα με τα string s_A και s_B αφού η H είναι ανθεκτική στις συγκρούσεις. Οπότε η συνθήκη γ μπορεί να ικανοποιηθεί μόνο αν η συνάρτηση επιλογής του νικητή f με είσοδο (s_A, s_B) βγάλει έξοδο την Α. Από την στιγμή που μόνο η Alice γνωρίζει το ιδιωτικό της κλειδί $\tilde{A}$, μόνο αυτή μπορεί να παρέχει την υπογραφή σ_1 το οποίο θα ικανοποιήσει το $ver_{\tilde{A}}(body, \sigma_1)$ και στην συνέχεια την συνθήκη γ.

Κάθε μέλος P μπορεί να αλλάξει γνώμη και να εξαργυρώσει την αρχική του συναλλαγή $PutMoney_1^P$ πριν την εμφάνιση της $Compute_1$ στον Ledger και έτσι ακυρώνεται η συναλλαγή $Compute_1$. Προφανώς αυτή η «αλλαγή γνώμης» δεν μπορεί να γίνει αφού το μέλος αυτό μάθει ότι δεν έχει κερδίσει. Οπότε η Alice και ο Bob περιμένουν μέχρι την χρονική στιγμή που η $Compute_1$ εμφανιστεί στον Ledger και μετά συνεχίζουν στο επόμενο βήμα της επιλογής του νικητή. Στο τελευταίο βήμα το μόνο που έχουν να κάνουν οι Alice και Bob είναι να δημοσιεύσουν τα s_A και s_B. Στην περίπτωση που $f(s_A, s_B) = A$ τότε η Alice μπορεί να εξαργυρώσει την κερδισμένη συναλλαγή $Compute_1$ με την συναλλαγή $ClaimMoney_1^A$ την οποία βλέπουμε στην συνέχεια:

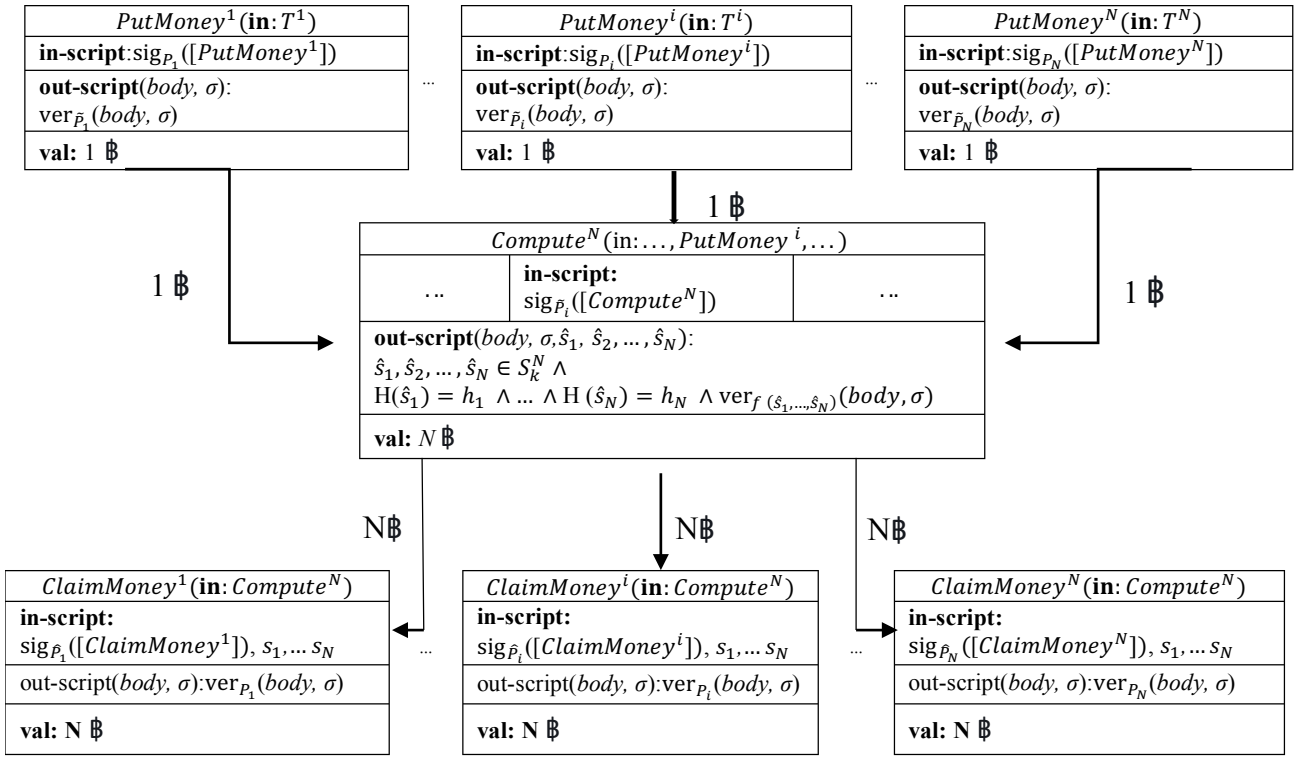
$ClaimMoney_1^A$ (**in**:$Compute_1$)
in $-$ **script**: $sig_{\tilde{A}}$ ($[ClaimMoney_1^A]$), $\perp, s_A, s_B$
out-script(body, σ):ver_A(body, σ)
val: 2 ฿

Από την άλλη πλευρά ο Bob δεν θα μπορούσε να εξαργυρώσει την $Compute_1$ αφού η συνθήκη $f(s_A, s_B) = B$ δεν ικανοποιείται.

Το πρόβλημα που δημιουργείται είναι ότι τα μέλη του πρωτοκόλλου δεν μπορούν να υποχρεωθούν και να στείλουν τα s_A και s_B. Στο σημείο αυτό θα χρησιμοποιήσουμε το Bitcoin-based timed commitment scheme το οποίο θα μας δώσει την λύση στο πρόβλημα αυτό.

Τώρα θα δούμε το Lottery πρωτόκολλο στην τελική του μορφή. Η γενική ιδέα είναι ότι κάθε μέλος αρχικά δεσμεύεται χρησιμοποιώντας το $CS(C, d, t, s)$ commitment scheme που έχουμε ήδη αναφέρει. Η φάση του ανοίγματος της δέσμευσης γίνεται στέλνοντας μια τιμή s και το άνοιγμα αυτό αυτό επιβεβαιώνεται ελέγχοντας αν $H(s) = h$ για κάποια τιμή h την οποία έχει στείλει ο Committer στην φάση της δέσμευσης. Αρχικά το πρωτόκολλο εκτελείται με την Alice ως Committer και τον Bob ως Recipient. Έστω s_A και h_A οι αντίστοιχες μεταβλητές s και h. Αντίστοιχα στην συνέχεια ο Bob εκτελεί το πρωτόκολλο ως Committer και την Alice Recipient και οι αντίστοιχες μεταβλητές είναι s_B και h_B. Αφού εκτελεστούν επιτυχώς οι δύο φάσεις δέσμευσης, τότε τα μέλη συνεχίζουν στα επόμενα βήματα τα οποία είναι τα ακόλουθα και είναι ίδια με αυτά που έχουμε περιγράψει πιο πάνω. Ο καθένας από τα μέλη δημοσιεύει την αντίστοιχη συναλλαγή του $PutMoney$ στον Ledger. Μόλις εμφανιστούν στον Ledger όλες αυτές οι συναλλαγές, τότε δημιουργείται και η συναλλαγή $Compute$ με τον τρόπο που έχουμε εξηγήσει και τέλος μόλις αυτή εμφανιστεί στον Ledger κάθε μέλος ανοίγει την δέσμευση του. Η διαφορά μας εδώ είναι ότι χρησιμοποιήθηκε το CS Commitment scheme το οποίο μας δίνει την δυνατότητα της τιμωρίας αν κάποιο μέλος δεν ανοίξει την δέσμευση του μετά το πέρας του χρόνου t, εκτελώντας την συναλλαγή $PayDeposit$. Επιπλέον κάθε τίμιο μέλος δεν θα χάσει ποτέ τα χρήματα που έχει βάλει στο ξεκίνημα του πρωτοκόλλου αφού αυτό εγγυάται ότι θα πάρει την κατάθεση του πίσω.

Στην συνέχεια ακολουθεί το γράφημα των συναλλαγών και η αναλυτική περιγραφή του πρωτοκόλλου:



Προ-απαιτούμενες συνθήκες

1. Για κάθε i, το μέλος P_i έχει ένα ζεύγος κλειδιών (P_i^{sk}, P_i^{pk}).

2. Για κάθε i, ο Ledger περιέχει μια συναλλαγή T^i αξίας 1 Bitcoin της οποίας ο παραλήπτης είναι ο P_i. Ο Ledger περιέχει επίσης τις συναλλαγές $\{U_j^i\}$, για $i, j \in \{1, \dots, N\}$ και $i \neq j$, έτσι ώστε κάθε U_j^i μπορεί να εξαργυρωθεί από τον P_i και έχει αξία $d = N$ Bitcoins.

Αρχική Φάση

a. Για κάθε i, ο P_i δημιουργεί το ζεύγος κλειδιών $(\tilde{P}_i^{sk}, \tilde{P}_i^{pk})$ και στέλνει το δημόσιο κλειδί του $\tilde{P}_i^{pk}$ σε όλα τα μέλη του πρωτοκόλλου.

b. Για κάθε i, ο P_i επιλέγει το μυστικό string s_i από το σύνολο S_k^N.

Φάση των Καταθέσεων

1. Έστω t η χρονική στιγμή που βρισκόμαστε τώρα. Για κάθε i, η φάση της δέσμευσης $CS.Commit(P_i, d, t + 4max_{Ledger}, s_i)$ εκτελείται με τις συναλλαγές $\{U_j^i\}$ ως είσοδο.

2. Σε περίπτωση που οποιεσδήποτε δύο δεσμεύσεις διαφορετικών μελών είναι ίδιες $h_i = h_j$ για $i \neq j$, τότε τα μέλη εγκαταλείπουν το πρωτόκολλο.

Φάση της Εκτέλεσης

1. Για κάθε i, ο P_i δημοσιεύει την συναλλαγή $PutMoney^\iota$ στον Ledger. Στην περίπτωση που δεν εμφανιστούν στον Ledger όλες οι συναλλαγές πριν την χρονική στιγμή $t + 2max_{Ledger}$ τότε τα μέλη εγκαταλείπουν το πρωτόκολλο.

2. Για κάθε $i \geq 2$, ο P_i υπογράφει στην συναλλαγή $Compute^N$ και την στέλνει στο μέλος P_1.

3. Ο P_1 μαζεύει όλες της υπογραφές των μελών μαζί με την δικιά του και τις βάζει ως είσοδο στην συναλλαγή $Compute^N$ και την δημοσιεύει στον Ledger. Αν η συναλλαγή δεν εμφανιστεί στον Ledger μέχρι την χρονική στιγμή $t + 3max_{Ledger}$ τότε τα μέλη εγκαταλείπουν το πρωτόκολλο.

4. Για κάθε i, ο P_i δημοσιεύει τις συναλλαγές $Open$ στον Ledger, οι οποίες αποκαλύπτουν το μυστικό και στέλνει πίσω στον ίδιο τις καταθέσεις που έκανε κατά την διάρκεια της εκτέλεσης του CS πρωτοκόλλου στο βήμα 3 της φάσης της δέσμευσης. Αν κάποιο μέλος δεν αποκάλυψε το μυστικό του μέχρι την χρονική στιγμή $t + 4max_{Ledger}$, τότε τα άλλα μέλη στέλνουν τις συναλλαγές $PayDeposit$ εκείνου του μέλους από το CS πρωτόκολλο και κερδίζουν N Bitcoins.

5. Ο νικητής $P_{f(s_1,\dots,s_N)}$ παίρνει τα χρήματα στέλνοντας την συναλλαγή $ClaimMoney^{f(s_1,\dots,s_N)}$ στον Ledger.

Τώρα ας δούμε την επιλογή των παραμέτρων t και d. Ο χρόνος που χρειάζεται για να ολοκληρωθεί το πρωτόκολλο είναι το πολύ $4max_{Ledger}$ μετά την χρονική στιγμή t' στην οποία ξεκινάει το πρωτόκολλο. Οπότε έχουμε $t = t' + 4max_{Ledger}$. Η παράμετρος d πρέπει να επιλεχθεί με τέτοιο τρόπο έτσι ώστε κάθε μέλος θα αποζημιώνεται στην περίπτωση που κάποιο μέλος εγκαταλείψει το πρωτόκολλο. Ας υπολογίσουμε τώρα την αποζημίωση κάποιου μέλους έστω P_1. Στην χειρότερη περίπτωση έχουμε ότι:

a. Το πρωτόκολλο διακόπτεται πάντα όταν ο P_1 πρόκειται να κερδίσει

b. Μόνο ένα μέλος εγκαταλείπει πάντα το πρωτόκολλο, οπότε ο P_1 πληρώνεται μόνο από μία κατάθεση εκείνου του μέλους.

Οπότε η αναμενόμενη αποζημίωση είναι $-\frac{N-1}{N}$ Bitcoin για την περίπτωση (a) και $\frac{d-1}{N}$ Bitcoin για την περίπτωση (b). Για να κάνουμε την τιμή ίση με το 0 ορίζουμε $d = N$ Bitcoin. Έτσι τα συνολικά χρήματα που θα πρέπει το κάθε μέλος να καταθέτει είναι $N(N-1)$ Bitcoin όπου N ο αριθμός των μελών του πρωτοκόλλου.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Marcin Andrychowicz, Stefan Dziembowski, Daniel Malinowski, and Łukasz Mazurek. Fair two-party computations via bitcoin deposits. In Rainer Bohme, Michael Brenner, Tyler Moore, and Matthew Smith, editors, Financial Cryptography and Data Security, Lecture Notes in Computer Science, pages 105–121. Springer Berlin Heidelberg, 2014.

[2] Marcin Andrychowicz, Stefan Dziembowski, Daniel Malinowski, and Łukasz Mazurek. On the Malleability of Bitcoin Transactions. Financial Cryptography and Data Security, Lecture Notes in Computer Science, pages 1-18. Springer Berlin Heidelberg, 2015.

[3] Marcin Andrychowicz, Stefan Dziembowski, Daniel Malinowski, and Łukasz Mazurek. Secure multiparty computations on bitcoin. In Security and Privacy (SP), 2014 IEEE Symposium on Security and Privacy, May 2014.

[4] Assaf Ben-David, Noam Nisan, and Benny Pinkas. FairplayMP: a system for secure multi-party computation. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, ACM CCS 08: 15th Conference on Computer and Communications Security, pages 257–266. ACM Press, October 2008.

[5] Adam Back and Iddo Bentov. Note on fair coin toss via bitcoin, 2013. http://www.cs.technion.ac.il/%7Eidddo/cointossBitcoin.pdf.

[6] A. Back. Hashcash – a denial of service counter-measure. 2002.

[7] S Barber, X. Boyen, E. Shi, , and E. Uzun. Bitter to better –how to make bitcoin a better currency. Financial Cryptography and Data Security, pages 399–414, 2012.

[8] Amos Beimel, Yehuda Lindell, Eran Omri, and Ilan Orlov. 1/p-Secure multiparty computation without honest majority and the best of both worlds. In Phillip Rogaway, editor, Advances in Cryptology – CRYPTO 2011, volume 6841 of Lecture Notes in Computer Science, pages 277–296. Springer, August 2011.

[9] N. Benger, J. van de Pol, and N.P. Smart. Ooh aah . . . just a little bit: A small anount of side channel can go a long way. IACR Cryptology ePrint Archive 2014: 161, 2014.

[10] bips/bip-0065.mediawiki. http://github.com/bitcoin/bips/blob/master/bip-0065.mediawiki, accessed od 10.12.2014.

[11] Bitcoin Wiki. Transaction malleability. http://en.bitcoin.it/wiki/Transaction Malleability, accessed on 20.10.2014.

[12] Bitcoin Wiki. Contracts. http://en.bitcoin.it/wiki/Contracts, accessed on 20.10.2014.

[13] J. Blomer, M. Otto, and J. P. Seifert. Sign change fault attacks on elliptic curve cryptosystems. Fault Diagnosis and Tolerance in Cryptography, pages 36–52, 2006.

[14] Peter Bogetoft, Dan Lund Christensen, Ivan Damg°ard, Martin Geisler, Thomas Jakobsen, Mikkel Kroigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt Nielsen, Jakob Pagter, Michael I. Schwartzbach, and Tomas Toft. Secure multiparty computation goes live. In Roger Dingledine and Philippe Golle, editors, FC 2009: 13th International Conference on Financial Cryptography and Data Security, volume 5628 of Lecture Notes in Computer Science, pages 325–343. Springer, February 2009.

[15] J. Bonneau and A. Miller. A cryptocurrency without public-key cryptography. 2014.

[16] Joppe W. Bos, J. Alex Halderman, Nadia Heninger, Jonathan Moore, Michael Naehrig and Eric Wustrow. Elliptic Curve Cryptography in Practice. Financial Cryptography and Data Security Volume 8437 of the series Lecture Notes in Computer Science pp 157-175. Springer Berlin Heidelberg, 2014.

[17] Bureau of Engraving and Printing - U.S. Department of the Treasury. Damaged currency. http://moneyfactory.gov/uscurrency/damagedcurrency.html, 2013.

[18] V. Buterin. Bitcoin is not quantum-safe, and how wen can fix it when needed. Bitcoin Maganzine, 2013.

[19] Vitalik Buterin. Bitcoin network shaken by Blockchain fork, March 2013. Bitcoin Magazine, available at http://bitcoinmagazine.com/3668/bitcoin-network-shaken-by-blockchain-fork

[20] Christian Cachin and Jan Camenisch. Optimistic fair secure computation. In Mihir Bellare, editor, Advances in Cryptology – CRYPTO 2000, volume 1880 of Lecture Notes in Computer Science, pages 93–111. Springer, August 2000.

[21] M. Ciet and M. Joye. (virtually) free randomization techniques for elliptic curve cryptography. Information and Communications Security, pages 348–359, 2003.

[22] M. Ciet and M. Joye. Elliptic curve cryptosystems in the presence of permanent and transient faults. Designs, codes and cryptography, 26(1):33–43, 2005.

[23] J. Clark and A. Essex. CommitCoin: Carbon dating commitments with bitcoin - (short paper). In A. D. Keromytis, editor, Financial Cryptography, volume 7397 of LNCS, pages 390{398. Springer, 2012.

[24] Richard Cleve. Limits on the security of coin flips when half the processors are faulty. In Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing, STOC ’86, pages 364–369, New York, NY, USA, 1986. ACM.

[25] CoinDesk. Why use bitcoin?

[26]. Coron J. S., Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems, In Proc. of 1999 Workshop on Cryptographic Hardware and Embedded Systems (CHES 1999) Koç Ç. K. and Paar C. (Eds.), Springer-Verlag LNCS 1717, 1999, 292.

[27] Ivan Damg°ard, Marcel Keller, Enrique Larraia, Valerio Pastro, Peter Scholl, and Nigel P. Smart. Practical covertly secure MPC for dishonest majority - or: Breaking the SPDZ limits. In ESORICS 2013: 18th European Symposium on Research in Computer Security, Lecture Notes in Computer Science, pages 1–18. Springer, 2013.

[28] Christian Decker and Roger Wattenhofer. Bitcoin transaction malleability and mtgox. In Miroslaw Kutylowski and Jaideep Vaidya, editors, Computer Security - ESORICS 2014, volume 8713 of Lecture Notes in Computer Science, pages 313–326. Springer International Publishing, 2014.

[29] C. Decker and R. Wattenhofer. Bitcoin transaction malleability and mtgox. arXiv preprint arXiv: 1403.6676, 2014.

[30]. Diffie W. and Hellman M. E., New Directions in Cryptography, IEEE Transactions on Information Theory, vol. IT-22, Nov. 1976, pp: 644-654. Koblitz N., Elliptic curve cryptosystems. Mathematics of Computation, 48, 20, 1987.

[31] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. SIAM Journal on Computing, 30(2):391–437, 2000.

[32] John R. Douceur. The sybil attack. In Revised Papers from the First International Workshop on Peer-to-Peer Systems, IPTPS '01, pages 251–260, London, UK, UK, 2002. Springer-Verlag.

[33] J. Fildes. iPhone hacker publishes secret Sony PlayStation 3 key. BBC News Technology,
2011.

[34] P.A. Fouque, D. Real, and F. Valette. The carry leakage on the randomized exponent countermeasure. Cryptographic Hardware and Embedded Systems- CHES 2008, pages 198–213, 2008.

[35] P.A. Fouque and F. Valette. The doubling attack- why upwards is better than downwards. Cryptographic Hardware and Embedded Systems- CHES 2003, pages 269–280, 2003.

[36] D. Gilson. Blockchain.info issues refunds to Bitcoin theft victims. http://www.coindesk.com/ blockchain-info-issues-refunds-to-bitcoin-theft-victims/, Aug 2013.

[37] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, 19th Annual ACM Symposium on Theory of Computing, pages 218–229. ACM Press, May 1987.

[38] D.M. Gordon. A survey of fast exponentiation methods. Journal of algorithms 27.1, pages 129–146, 1998.

[39] S. Dov Gordon and Jonathan Katz. Partial fairness in secure two-party computation. In Henri Gilbert, editor, Advances in Cryptology – EUROCRYPT 2010, volume 6110 of Lecture Notes in Computer Science, pages 157–176. Springer, May 2010.

[40] S. Haber and W. S. and Stometta. How to time-stamp a digital document. Journal of Cryptology, 3(2):99–111, 1991.

[41]. Hankerson D., Menezes A. and Vanstone S., Guide to Elliptic Curve Cryptography, Springer-Verlag & Hall/CRC, New York, 2004.

[42] J.W. hung, S.G. Sim, and P. J. Lee. Fast implementation of elliptic curve defines over gf (pm) on calmrisc with mac2424 coprocessor. Cryptographic Hardware and Embedded Systems –CHES 2000, pages 57–70, 2000.

[43] K. Itoh, M. Takenaka, and N. Torii. ‘fast implementation of public key cryptography on a
dsp tms320c6201. Cryptographic Hardware and Embedded Systems, pages 61–72, 1999.

[44] E. Kasper. Fast elliptic curve cryptography in openSSl. Financial Cryptography and Data Security, pages 27–39, 2012.

[45]. Kocher P., Timing attacks on implementations of Diffe-Hellman, RSA, DSS and other systems, Advances in Cryptology, Proceedings of Crypto' 96, Koblitz N., (Ed.), Springer-Verlag LNCS 1109, 1996, 104. 176

[46]. Kocher P., Jaffe J. and Jun B., Differential Power Analysis, Advances in Cryptology, Proceedings of Crypto' 99, Wiener M. (Ed.), Springer-Verlag LNCS 1666, 1999, 388.

[47] Dahlia Malkhi, Noam Nisan, Benny Pinkas, and Yaron Sella. Fairplay - a secure two-party computation system. In Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13, SSYM’04, pages 20–20, Berkeley, CA, USA, 2004. USENIX Association.

[48] L. Martin. The remote timing attack against openssl’s ECDSA. Voltage Security, 2011.

[49]. Menezes A., Okamoto T. and Vanstone S., Reducing elliptic curve logarithms to logarithms in a finite field, IEEE Transactions on Information Theory 39, 1639, 1993.

[50] A.J. Menezes, P. C. Van Oorschot, and S.A. Vanstone. Handbook of applied cryptography. CRC press, 2012.

[51] R.C. Merkle. A digital signature based on a conventional encryption function. Advances in Cryptology – CRYPTO’87, 1988.

[52]. Miller V., Use of elliptic curves in cryptography. In Proc. Advances in Cryptology—CRYPTO’85, Springer-Verlag LNCS 218, 1986, 417.

[53] P.L. Montgomery. Speeding the pollard and elliptic curve methods of factorization. Mathematics of computation, pages 243–264, 1987.

[54] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system. 2008.

[55] K. Okeya, H. Kurumatani, and K. Sakurai. Elliptic curves with the Montgomery form and
their cryptographic applications. Public Key Cryptography, pages 238–257, 2000.

[56] A. Dominguez Oviedo. On fault-based attacks and countermeasures for elliptic curve cryptosystems. 2008.

[57] T. Pornin. Deterministic usage of the digital signature algorithm (DSA) and elliptic curve
digital signature algorithms (ECDSA). 2013.

[58] D. Ron and A. Shamir. Quantitative analysis of the full bitcoin transaction graph. Financial Cryptography and Data Security, pages 6–24, 2013.

[59] K. Shirriff. Bitcoins the hard way: Using the raw bitcoin protocol. Ken Shirriff's blog, 2014.

[60] Ken Shirriff Bitcoin transaction malleability, looking at the bytes. Ken Shirriff's blog, 2014.

[61]. Silverman J. H., The Arithmetic of Elliptic Curves, GTM 106, Springer-Verlag, 1986

[62]. Silverman J. H. and Tate J., Rational Points on Elliptic Curves, Springer-Verlag, 1992

[63] Y. Sung-Ming, S. Kim, and S. Lim. A countermeasure against one physical cryptanalysis may benefit another attack. Information Security and Cryptology – ICISC 2001, pages
414–427, 2002.

[64] The Washington Post. Cheating scandals raise new questions about honesty, security of internet gambling, November 30, 2008.

[65] L. Tung. Security flaw leaves android bitcoin wallets vulnerable to theft. ZDNet, 2013.

[66]. Washington L. C., Elliptic Curves: Number Theory and Cryptography, Chapman & Hall/CRC, New York, 2003.

[67] Joe Weisenthal. Bitcoin just completely crashed as major exchange says withdrawals remain halted, Feb. 2014. Business Insider, available at www.businessinsider.com/mtgox-statement-on-withdrawals-2014-2.

[68] P. Wuille. Dealing with malleability. BIP0062, 2014.