ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΚΡΥΠΤΟΓΡΑΦΙΑ

(2021 – 2022)

4η Σειρά Ασκήσεων

Ονοματεπώνυμο:

- Χρήστος Τσούφης

Αριθμός Μητρώου:

- 031 17 176

Στοιχεία Επικοινωνίας:

- el17176@mail.ntua.gr
- chris99ts@gmail.com

1η Άσκηση

Δίνεται το παρακάτω πρωτόκολλο μεταξύ ενός prover $\mathcal{P}$ και ενός verifier $\mathcal{V}$ το οποίο έχει στόχο την απόδειξη γνώσης του μηνύματος που αντιστοιχεί σε ένα δεδομένο κρυπτοκείμενο RSA με δημόσιο κλειδί (e, n), δηλαδή $m \in \mathbb{Z}_n^*$ τέτοιο ώστε $y = m^e (mod\ n)$. Επιπλέον θεωρήστε ότι e πρώτος.

- *Ο $\mathcal{P}$ επιλέγει τυχαία ένα $t \in \mathbb{Z}_n^*$ και στέλνει στον $\mathcal{V}$ το $h = t^e\ mod\ n$.*
- *Ο $\mathcal{V}$ επιλέγει ένα τυχαία c, $c \in \{0, \dots, e-1\}$, και το στέλνει στον P.*
- *Ο $\mathcal{P}$ υπολογίζει το $r = tm^c\ mod\ n$ και το στέλνει στον V.*
- *Ο $\mathcal{V}$ αποδέχεται αν και μόνο αν $r^e \equiv hy^c (mod\ n)$.*

Να αποδείξετε ότι το παραπάνω είναι Σ-πρωτόκολλο. Για την ιδιότητα HVZK η απόδειξη πρέπει να είναι στο επίπεδο ανάλυσης που ακολουθήθηκε στις διαφάνειες, αλλά να φαίνονται αναλυτικά τα transcripts του πρωτοκόλλου και η πιθανότητα εμφάνισής τους.

Λύση:

Για να χαρακτηριστεί ένα πρωτόκολλο ως Σ-πρωτόκολλο, θα πρέπει να έχει πληρότητα, ειδική ορθότητα (special soundness) και HVZK (honest verifier zero knowledge). Στην συνέχεια εξετάζονται αυτές οι συνθήκες:

1ος Τρόπος Επίλυσης:

1. Πληρότητα: Η απόδειξη είναι προφανής:

$$r = tm^c \Rightarrow r^e = t^e \cdot m^{c \cdot e} \Rightarrow r^e = h \cdot y^c (mod\ n)$$

2. Ειδική Ορθότητα: Έστω ότι ο prover στέλνει δυο φορές το ίδιο commitment h. Οπότε, από την πρώτη εκτέλεση θα είναι [h, c, r] και από την δεύτερη θα είναι [h, c', r'].

Το $\frac{r}{r'} = m^{c-c'}$ υπολογίζεται αν τα c, c' είναι κοντινές τιμές και τότε μπορεί να σπάσει ο Μικρός Διακριτός Λογάριθμος. Μάλιστα, αν τα c επιλέγονται τυχαία με πολύ μεγάλη πιθανότητα σε ένα μικρό αριθμό επαναλήψεων, τότε θα υπάρξουν δύο τιμές του c που να είναι αρκετά κοντά ώστε τελικά να υπάρξει σπάσιμο.

3. HVZK: Έστω S που έχει την δυνατότητα για restarts και επίσης έστω honest V.

- Πρώτα, στέλνει ο S στον V έναν αριθμό h και ο V απαντάει με μια πρόκληση c.
- Αν ο S απαντήσει σωστά με αμελητέα πιθανότητα τότε συνεχίζεται η διαδικασία κανονικά αλλιώς γίνεται rewind.
- Μετά το rewind ο S δεσμεύεται στην τιμή $y = y^{-c} \cdot t^e (mod\ n)$.
- Ο V ξαναστέλνει την πρόκληση c και ο S απαντάει με t.
- Ο V σίγουρα θα δεχτεί αφού $t^e = h \cdot y^c (mod\ n)$

Οπότε, παρατηρεί κανείς ότι το $[t \in \mathbb{Z}_n^*, h = t^e\ mod\ n, c \in \mathbb{Z}_{e-1}, r = t \cdot m^e (mod\ n)]$ έχει ίδια κατανομή με το $[t \in \mathbb{Z}_n^*, c \in \mathbb{Z}_{e-1}, h = y^{-c} t^e (mod\ n), t]$.

2ος Τρόπος Επίλυσης:

1. Πληρότητα: Ένας τίμιος (honest) Prover πείθει πάντοτε έναν τίμιο Verifier:

$$r = tm^c \Rightarrow r^e = (t \cdot m^c)^e \Rightarrow r^e = t^e \cdot (m^e)^c \Rightarrow r^e = h \cdot y^c (mod\ n)$$

2. Ειδική Ορθότητα (Special Soundness): Το ίδιο commit οδηγεί σε αποκάλυψη του μυστικού. Έστω ότι ο committer δεσμεύεται στο $y = t^e (mod\ n)$. Κατά την πρώτη εκτέλεση θα ισχύει ότι η πρόκληση (challenge) c_1 θα έχει response $r_1 = t \cdot m^{c_1}$. Αντίστοιχα, για την δεύτερη εκτέλεση θα ισχύει ότι για c_2 θα έχει response $r_2 = t \cdot m^{c_2}$. Συνεπώς, $\frac{r_1}{r_2} = m^{c_1 - c_2} \Rightarrow m = \left(\frac{r_1}{r_2}\right)^{\frac{1}{c_1 - c_2}}$.

3. HVZK (Ορθότητα): Ένας κακόβουλος Prover πείθει με αμελητέα πιθανότητα έναν τίμιο Verifier. Αυτό μπορεί να συμβεί μόνο αν δεσμευτεί με το σωστό challenge το οποίο έχει πιθανότητα $\frac{1}{e}$.

Επιπλέον, για να θεωρηθεί πρωτόκολλο μηδενικής γνώσης με τίμιο Verifier θα πρέπει να υπάρχει ppt Simulator S που μπορεί να πείσει ένα τίμιο Verifier.

- Πρώτα, ο S επιλέγει ένα $t \in \mathbb{Z}_n^*$ και δεσμεύεται στο $h = t^e (mod\ n)$.
- Μετά, ο V επιλέγει τυχαία ένα $c \in \{0, e-1\}$.
- Ο S απαντάει σωστά με αμελητέα πιθανότητα ενώ, αν δεν μπορεί να απαντήσει κάνει rewind τον V.
- Κατά την δεύτερη εκτέλεση, ο S δεσμεύεται στο $h = t^e \cdot y^{-c} (mod\ n)$, και εφόσον υπάρχει τίμιος Verifier, θα στείλει πάλι το ίδιο challenge c.
- Ο S απαντάει $r = t$ και ο V δέχεται μόνο αν $y^c \cdot h = y^c \cdot t^e \cdot y^{-c} = t^e = r^e$.

Με την ίδια κατανομή επιτυγχάνεται και η συνομιλία μεταξύ Prover και Simulator.

2η Άσκηση

Έστω το παρακάτω πρωτόκολλο μηδενικής γνώσης. Οι δημόσιες παράμετροι είναι $\langle p, m, g, h \rangle$ *και ο prover γνωρίζει ένα x τέτοιο ώστε* $g^x = h(mod\ p)$.

- *O prover επιλέγει τυχαία ένα* $t \in \mathbb{Z}_m^*$ *και στέλνει στον verifier το* $y = g^t(mod\ p)$.
- *O verifier επιλέγει τυχαία* $c \in \mathbb{Z}_m^*$ *και το στέλνει στον prover.*
- *O prover υπολογίζει το* $s = t + c + x$ *και το στέλνει στον verifier.*
- *O verifier αποδέχεται αν και μόνο αν* $g^s = yg^c h(mod\ p)$.

Εξετάστε αν το παραπάνω πρωτόκολλο είναι μηδενικής γνώσης για τίμιους επαληθευτές.

Λύση:

Ένα πρωτόκολλο θεωρείται ότι είναι Μηδενικής Γνώσης για τίμιους verifiers όταν ο verifier δεν μαθαίνει καμία επιπλέον πληροφορία εκτός από αυτό που θέλει να αποδείξει ο prover. Δηλαδή, όταν υπάρχει ppt Simulator S που να μπορεί να πείσει ένα τίμιο επαληθευτή.

1ος Τρόπος Επίλυσης:

Η διαδικασία αυτή πραγματοποιείται ως εξής:

- Έστω S με δυνατότητα rewind και V.
- Ο S δεσμεύεται σε μια τιμή, έστω y και ο V επιλέγει τυχαία ένα challenge c.
- Αν ο S απαντήσει σωστά με αμελητέα πιθανότητα, τότε συνεχίζεται κανονικά η διαδικασία, διαφορετικά γίνεται rewind.
- Ο S δεσμεύεται στην τιμή $y = g^{t-c} \cdot h^{-1}(mod\ p)$.
- Ο V κάνει το ίδιο challenge c.
- Ο S απαντάει με t και ο V αποδέχεται αφού $g^t = y \cdot g^c \cdot h = g^{t-c} \cdot h^{-1} \cdot g^c \cdot h = g^t$.

Τελικά, παρατηρεί κανείς ότι αυτός ο "διάλογος" έχει ίδια κατανομή με ένα διάλογο ενός P που θα είχε μόνο ένα witness.

Ωστόσο, κρίνεται απαραίτητο να σημειωθεί ότι το πρωτόκολλο αυτό, παρά το γεγονός ότι δεν μεταφέρει γνώση από τον P στον V, δεν μπορεί να θεωρηθεί απόδειξη ως έχει. Αυτό ισχύει διότι, αν ο P δεσμευτεί αρχικά στην τιμή $y = h^{-1}(mod\ p)$ και στο challenge c απαντήσει με $s = c$, τότε πάντα ο V θα κάνει accept.

2ος Τρόπος Επίλυσης:

1. Πληρότητα: Ένας τίμιος (honest) Prover πείθει πάντοτε έναν τίμιο Verifier Αν υπάρχει x τέτοιο ώστε $h = g^x \ mod \ p$, ο V θα αποδέχεται αν ισχύει ότι:

$$g^s \equiv g^{t+c+x} \equiv g^t \cdot g^c \cdot g^x \equiv y \cdot g^c \cdot h (mod\ p)$$

Οπότε, πράγματι ο V αποδέχεται.

2. Ορθότητα: Αν δεν είναι γνωστό το x ώστε $h = g^x \ mod \ p$, τότε δεν θα πρέπει ο V να αποδέχεται με καλή πιθανότητα. Όμως, αν ο P στείλει $y = h^{-1}g^t$ και $s = t + c$ τότε:

$$g^s \equiv g^{t+c} \equiv g^t \cdot g^c \equiv g^t \cdot g^c \cdot h^{-1} \cdot h \ (mod\ p) \Rightarrow g^s \equiv g^t h^{-1} g^c h \equiv y g^c h \ (mod\ p)$$

Οπότε, ο V θα αποδέχεται λανθασμένα. Άρα το Πρωτόκολλο δεν έχει την ιδιότητα της ορθότητας, άρα δεν είναι HVZK.

Με άλλα λόγια, ένας κακόβουλος Prover πείθει με αμελητέα πιθανότητα έναν τίμιο Verifier. Αυτό μπορεί να συμβεί μόνο αν δεσμευτεί με το σωστό challenge το οποίο έχει πιθανότητα $\frac{1}{m}$.

3. HVZK: Είναι θεμιτό οι κατανομές των συζητήσεων μεταξύ Prover & honest Verifier, Prover & Simulator να είναι ίδιες. Επιπλέον, για να θεωρηθεί πρωτόκολλο μηδενικής γνώσης με τίμιο Verifier θα πρέπει να υπάρχει ppt Simulator S που μπορεί να πείσει ένα τίμιο Verifier.

- Πρώτα, ο S επιλέγει ένα $t \in \mathbb{Z}_m^*$ και δεσμεύεται στο $y = g^t (mod\ p)$.
- Μετά, ο V επιλέγει τυχαία ένα $c \in \mathbb{Z}_m^*$.
- Ο S απαντάει σωστά με αμελητέα πιθανότητα ενώ, αν δεν μπορεί να απαντήσει κάνει rewind τον V.
- Κατά την δεύτερη εκτέλεση, ο S δεσμεύεται στο $y = g^t \cdot g^{-c} \cdot h^{-1} (mod\ p)$, και εφόσον υπάρχει τίμιος Verifier, θα στείλει πάλι το ίδιο challenge c.
- Ο S απαντάει $s = t$ και ο V δέχεται μόνο αν $y \cdot g^c \cdot h = g^t \cdot g^{-c} \cdot h^{-1} \cdot g^c \cdot h = g^t = g^s$.

Οπότε, οι συζητήσεις:

$$\langle P, V \rangle = \langle t \in \mathbb{Z}_m^* : g^t, c \in \mathbb{Z}_m^*, t + c + x \rangle$$

$$\langle S, V \rangle = \langle t \in \mathbb{Z}_m^* : h^{-1} g^t, c \in \mathbb{Z}_m^*, t + c \rangle$$

θα έχουν ακριβώς την ίδια κατανομή.

3η Άσκηση

Να αποδείξετε ότι ένα σχήμα δέσμευσης δεν μπορεί να διαθέτει ταυτόχρονα τις ιδιότητες τέλειας δέσμευσης και τέλειας απόκρυψης.

Λύση:

Ένα σχήμα τέλειας δέσμευσης σημαίνει ότι κανείς (ούτε ο αποστολέας) δεν μπορεί να αλλάξει την τιμή στην οποία δεσμεύτηκε το σχήμα εκ των υστέρων. Δηλαδή, δεν μπορούν δύο διαφορετικές τιμές να δώσουν την ίδια δέσμευση διότι, τότε ο καθένας θα μπορούσε να αλλάξει την αρχική του τιμή επιλέγοντας μια άλλη με την ίδια δέσμευση. Με λίγα λόγια, κάθε commitment αντιστοιχεί σε ακριβώς ένα μήνυμα

Ένα σχήμα τέλειας απόκρυψης σημαίνει ότι δυο διαφορετικές τιμές καταλήγουν στην ίδια δέσμευση έτσι ώστε να μην μπορεί κανείς να αποφανθεί την αρχική τιμή (δεν μπορεί να διαρρεύσει το μήνυμα της δέσμευσης). Δηλαδή, αν το σχήμα είναι "1-1" τότε, μέσω διαδοχικών δοκιμών, ο καθένας μπορεί να υπολογίσει την αρχική τιμή στην οποία δεσμεύτηκε ο άλλος. Με άλλα λόγια, απαιτεί δυο διαφορετικά μηνύματα να παράγουν το ίδιο commitment για να μην αποκαλύπτεται το μήνυμα.

Ωστόσο, ένας powerful committer είναι δυνατό να αλλάξει την δέσμευσή του οπότε δεν είναι εφικτή η τέλεια δέσμευση. Αντίστοιχα, ένας powerful αντίπαλος είναι δυνατό να βρει το μήνυμα οπότε δεν είναι εφικτή η τέλεια απόκρυψη.

Συνεπώς, οι ιδιότητες της τέλειας δέσμευσης και της τέλειας απόκρυψης δεν γίνει να συνυπάρχουν.

4η Άσκηση

Να δώσετε τις μη-διαλογικές αποδείξεις χρησιμοποιώντας την τεχνική Fiat-Shamir για τις παρακάτω ιδιότητες του πρωτοκόλλου ηλεκτρονικών ψηφοφοριών CGS97 που παρουσιάζεται στις διαφάνειες της ενότητας "Αποδείξεις Μηδενικής Γνώσης και Εφαρμογές":

- *Ορθή αποκρυπτογράφηση (βλ. διαφάνεια 56)*
- *Εγκυρότητα αρνητικής ψήφου (βλ. διαφάνεια 57)*

Λύση:

- Ορθή αποκρυπτογράφηση:

Με το πέρας της διαδικασίας και την αποκρυπτογράφηση έχει προκύψει ένα x για το οποίο ισχύει ότι:

$$g^x = g^{\#Yes-\#No}(mod\ n)$$

Για την απόδειξη θα χρησιμοποιηθεί Schnorr με Fiat-Shamir ως εξής:

- Πρώτα, επιλέγεται ένα τυχαίο t για τον υπολογισμό της ποσότητας $y = g^t(mod\ p)$.
- Έπειτα, υπολογίζεται η τιμή $c = H(y)$, όπου H μια collision free hash function.
- Μετά, υπολογίζεται η τιμή $s = t + c \cdot x(mod\ p)$.
- Τέλος, ο καθένας μπορεί να ελέγξει και να αποδεχθεί εάν ισχύει:

$$g^s = y \cdot h^c(mod\ p)$$

- Εγκυρότητα αρνητικής ψήφου:

Για την απόδειξη θα χρησιμοποιηθεί OR-Schnorr με Fiat-Shamir ως εξής:

- Πρώτα, για να δειχθεί ότι η ψήφος ανήκει στο σύνολο των αποδεκτών τιμών, επιλέγονται t_1, t_2 και υπολογίζονται οι ποσότητες $T_1 = g^{t_1} \cdot a^{-c_1}$, $T_2 = y^{t_1} \cdot \frac{\beta^{-c_1}}{G}$, $T_3 = g^{t_2}$ και $T_4 = y^{t_2}$.
- Έπειτα, υπολογίζεται η τιμή $c = H(T_1 \parallel T_2 \parallel T_3 \parallel T_4)$, όπου H μια collision free hash function.
- Μετά, υπολογίζονται τα $c_2 = c - c_1$, $s_1 = t_1$, $s_2 = t_2 + c_2 \cdot r$.
- Ύστερα, δημοσιεύονται τα c, c_1, c_2, s_1, s_2.
- Τέλος, ο καθένας μπορεί να ελέγξει και να αποδεχτεί εάν ισχύει:

$$c = c_1 + c_2, g^{s_1} = T_1 \cdot a^{c_1}, y^{s_1} = T_2 \cdot \beta^{c_1} \cdot G^{-c_1}, g^{s_2} = T_3 \cdot a^{c_2}, y^{s_2} = T_4 \cdot \beta^{c_2} \cdot G^{c_2}$$

5η Άσκηση

Μία συνάρτηση σύνοψης $\mathcal{H}:\{0,1\}^* \to \{0,1\}^n$ *είναι ασφαλής για χρήση σε συστήματα Proof of Work (PoW) αν για κάθε είσοδο x είναι δύσκολο να βρεθεί λύση r ώστε να ισχύει* $\mathcal{H}(x \parallel r) \in Y$*, όπου Y κάποιο σημαντικά μικρό υποσύνολο του* $\{0,1\}^n$.

1. *Να αποδείξετε ότι μία συνάρτηση σύνοψης που έχει την ιδιότητα collision resistance δεν είναι απαραίτητα ασφαλής για PoW.*
   Υπόδειξη: Να κατασκευάσετε ένα αντιπαράδειγμα, δηλαδή μια συνάρτηση $\mathcal{H}'$ *που είναι collision resistant, αλλά όχι ασφαλής για PoW, επεκτείνοντας μια συνάρτηση* $\mathcal{H}$ *που είναι collision resistant και ασφαλής για PoW.*
2. *Να δείξετε ότι η συνάρτηση* $\mathcal{G}(z) = \mathcal{H}(z) \parallel LSB(z)$*, όπου* $LSB(z)$ *είναι το λιγότερο σημαντικό bit του z, είναι ασφαλής για PoW αλλά δεν έχει αντίσταση πρώτου ορίσματος.*

Λύση:

1ος Τρόπος Επίλυσης:

1. Έστω μια συνάρτηση σύνοψης $\mathcal{H}:\{0,1\}^* \to \{0,1\}^n$ που είναι collision resistant.

Έστω, επίσης, μια συγκεκριμένη δυσκολία d για Πρόβλημα Proof of Work (PoW).

Έστω, ακόμη, μια συνάρτηση $\mathcal{H}'(x) = 0^k \parallel \mathcal{H}(x)$ οπότε $\mathcal{H}:\{0,1\}^* \to \{0,1\}^{n+k}$.

Τότε, για δυσκολία d στο PoW θα πρέπει

$$\mathcal{H}'(x) < \frac{2^{n+k}}{d} \xrightarrow{k=\log_2 d} \mathcal{H}'(x) < 2^n$$

Αυτό ισχύει διότι η μέγιστη τιμή που μπορεί να πάρει το $\mathcal{H}(x)$ είναι 2^{n-1}.

Επιπλέον, αφού η $\mathcal{H}$ είναι collision resistant, τότε και η $\mathcal{H}'$ θα είναι. Αυτό ισχύει διότι, για z, z':

$$\mathcal{H}'(z) = \mathcal{H}'(z') \Rightarrow 0^k \parallel \mathcal{H}(z) = 0^k \parallel \mathcal{H}(z') \Rightarrow \mathcal{H}(z) = \mathcal{H}(z') \Rightarrow z = z'$$

2. Εδώ διακρίνονται οι εξής περιπτώσεις:

- Εάν η $\mathcal{H}$ είναι ασφαλής για PoW, τότε και η $\mathcal{G}$ θα είναι. Αυτό ισχύει διότι με την επιλογή του z επηρεάζεται μόνο το LSB του $\mathcal{G}$ οπότε δεν μειώνεται η δυσκολία του προβλήματος στην ουσία.
- Εάν η $\mathcal{H}$ είναι collision resistant, τότε και η $\mathcal{G}$ θα είναι. Αυτό συνεπάγεται αντίσταση πρώτου ορίσματος διότι για z, z':

$$\mathcal{G}(z) = \mathcal{G}(z') \Rightarrow \mathcal{H}(z) \parallel LSB(z) = \mathcal{H}(z') \parallel LSB(z') \Rightarrow$$
$$\Rightarrow \mathcal{H}(z) = \mathcal{H}(z') \wedge LSB(z) = LSB(z') = LSB(z') \Rightarrow z = z'$$

2ος Τρόπος Επίλυσης:

1. Έστω μια συνάρτηση σύνοψης $\mathcal{H}: \{0,1\}^l \to \{0,1\}^n$ που είναι collision resistant και ασφαλής για PoW. Έστω επίσης, η συνάρτηση σύνοψης $\mathcal{H}': \{0,1\}^{2l} \to \{0,1\}^{n+l}$ έτσι ώστε $\mathcal{H}'(x \parallel r) = \mathcal{H}(x) \parallel r$.

Αρχικά, θα δειχθεί ότι $\mathcal{H}'$ είναι collision resistant.

Απόδειξη:

Έστω ότι δεν είναι collision resistant. Τότε, θα υπάρχουν $x_1 \neq x_2$ με $x_1 = x_1' \parallel r_1$ και $x_2 = x_2' \parallel r_2$ ώστε: $\mathcal{H}'(x_1) = \mathcal{H}'(x_2) \Rightarrow \mathcal{H}(x_1') \parallel r_1 = \mathcal{H}(x_2') \parallel r_2 \Rightarrow \mathcal{H}(x_1') = \mathcal{H}(x_2') \,\&\, r_1 = r_2$

Οπότε, είναι εφικτός ο αποδοτικός υπολογισμός collision για την $\mathcal{H}$, το οποίο είναι άτοπο.

Έπειτα, θα δειχθεί ότι $\mathcal{H}'$ δεν είναι ασφαλής για PoW.

Απόδειξη:

Για κάθε είσοδο x, μπορεί να θεωρηθεί r σταθερό, π.χ. r = 0 … 0 (l-φορές).

Τότε, $\mathcal{H}'(x \parallel r) = \mathcal{H}'(x \parallel 0) = \mathcal{H}(x) \parallel 0 \in \{0,1\}^n$

Άρα, $y = \{0,1\}^n$, με $|y| = 2^n \ll |\{0,1\}^{n+l}| = 2^{n+l}$ που είναι το πεδίο τιμών της $\mathcal{H}'$.

Άρα, υπάρχει συνάρτηση $\mathcal{H}'$ τέτοια ώστε να είναι collision resistant και μη ασφαλής για PoW. Συνεπώς, collision resistance δεν συνεπάγεται και ασφάλεια για PoW.

2. Αρχικά, θα δειχθεί ότι η $\mathcal{G}(z)$ είναι ασφαλής για PoW.

Απόδειξη:

Έστω ότι δεν είναι ασφαλής για PoW. Τότε, για κάποιο x θα υπάρχει αλγόριθμος A που να υπολογίζεται y τέτοιος ώστε:

$$\mathcal{G}(x \parallel r) = \mathcal{H}(x \parallel r) \parallel LSB(x \parallel r) = \mathcal{H}(x \parallel r) \parallel LSB(r) \in Y$$

Όπου $|y| \ll 2^{n+1}$, αφού $\mathcal{G}: \{0,1\}^* \to \{0,1\}^{n+1}$.

Τότε, για το σύνολο $Y' = \{y' : y = y' \parallel i, y \in Y \; \gamma\iota\alpha \; \kappa\acute{\alpha}\pi o\iota o \; i\}$ θα ισχύει ότι $|Y'| = |Y| \ll 2^{n+1} \Rightarrow |Y'| \ll 2^n$.

Οπότε, μπορεί να χρησιμοποιηθεί ο A, με είσοδο x, για να προκύψει το r ώστε: $\mathcal{H}(x \parallel r) \in Y'$, αφού $\mathcal{G}(x \parallel r) \in Y$. Συνεπώς, δείχθηκε ότι η $\mathcal{H}$ δεν είναι ασφαλής για PoW, το οποίο είναι άτοπο.

Επίσης, θα δειχθεί ότι η $\mathcal{G}(z)$ δεν έχει αντίσταση πρώτου ορίσματος.

Απόδειξη:

Για $y \in Y$, μπορεί να βρεθεί x τέτοιο ώστε $\mathcal{G}(z) = y$. Πράγματι, για $x = x' \parallel i, i \in \{0,1\}$ υπολογίζεται το $y = \mathcal{H}(x)$ και προκύπτει $y' = y \parallel i$. Τότε, για $y' \in Y = \{0,1\}^{n+1}$ θα ισχύει: $\mathcal{G}(x) = \mathcal{H}(x) \parallel LSB(x) = y \parallel i = y'$.

6η Άσκηση

1. *Περιγράψτε ένα σενάριο στο οποίο δύο miners στο bitcoin δίκτυο ενώ ακολουθούν το πρωτόκολλο πιστά δημιουργούν δύο διαφορετικές αλυσίδες τις οποίες και ακολουθούν.*
2. *Να επιχειρηματολογήσετε ότι το παραπάνω σενάριο που περιγράψατε συμβαίνει με μικρή πιθανότητα.*
3. *Η Μίνα, μια κακόβουλη miner, σε κάθε block που βλέπει αλλάζει το coinbase transaction ώστε να πληρώνεται η ίδια πριν το κάνει relay στο δίκτυο. Γιατί η Μίνα δεν βγάζει επιπλέον κέρδη;*

Λύση:

1. Δύο miners στο Bitcoin δίκτυο, ενώ ακολουθούν το πρωτόκολλο πιστά, τελικά δημιουργούν αλυσίδες τις οποίες και ακολουθούν όταν από την επιλογή διαφορετικών συναλλαγών, συμπεριλάβουν στο block τους και τελικά επικυρώσουν μέσω PoW σχεδόν ταυτόχρονα, οπότε δεν θα προλάβει ο ένας να ενημερωθεί για την ύπαρξη block από τον άλλο.

Με άλλα λόγια, όταν εισαχθεί ένα block στο blockchain, θα πρέπει ο miner να κάνει PoW. Δηλαδή απαιτείται ένα nonce τέτοιο ώστε το hash value του νέου block να είναι μικρότερο από ένα target που ορίζεται από το πρωτόκολλο, γεγονός που απαιτεί αρκετές δοκιμές για την επίτευξή του. Μάλιστα, για την ταυτόχρονη εισαγωγή δυο block στην αλυσίδα, θα πρέπει οι δυο τίμιοι miners να βρουν ταυτόχρονα nonce για τα block τους και να μην έχουν προλάβει να ενημερωθούν για την αλλαγή που έκανε στο δίκτυο ο άλλος.

2. Όταν συμβεί το παραπάνω σενάριο, το ένα νέο block θα μπει ως συνέχεια του άλλου block δημιουργώντας τελικά μακρύτερη αλυσίδα σε σχέση με την άλλη. Έτσι, όλοι οι miners θα ακολουθήσουν αυτό το block. Ο μοναδικός τρόπος για να συνεχιστούν και τα δύο block είναι αν πάλι σχεδόν ταυτόχρονα κλείσουν από ένα block στην άκρη της κάθε αλυσίδας. Επομένως, αφού είναι ήδη αρκετά δύσκολο να βρουν απάντηση στο PoW ταυτόχρονα, είναι ακόμα σπανιότερα να συμβεί δύο φορές. Με άλλα λόγια, το PoW απαιτεί αρκετή δουλειά και συνεπώς, έχει πολύ μικρή πιθανότητα να γίνει εισαγωγή νέου block ταυτόχρονα.

3. Η Μίνα, μια κακόβουλη miner, σε κάθε block που βλέπει, αλλάζει το coinbase transaction ώστε να πληρώνεται η ίδια πριν το κάνει relay στο δίκτυο. Με την αλλαγή του coinbase transaction, αλλάζει το block. Ωστόσο, εάν το block χρησιμοποιείται ως nonce στο PoW, θα αλλάξει το nonce και έτσι, θα πρέπει εκ νέου να υπολογιστεί η τιμή για το PoW αφού πλέον η παλιά θεωρείται εσφαλμένη. Συνεπώς, η Μίνα θα πρέπει να κάνει ξανά mine το block οπότε δεν θα βγάλει κάποιο επιπλέον κέρδος από αυτή την διαδικασία.

7η Άσκηση (‘bonus’)

Σε αυτή την άσκηση, θα γνωρίσουμε καλύτερα ορισμένες τεχνικές πτυχές του Bitcoin, και θα δοκιμάσουμε κάτι σαν “mining”. Συγκεκριμένα, μπορείτε να διαβάσετε περισσότερα για τα transactions του Bitcoin εδώ και σχετικά με την ακριβή κωδικοποίηση και δημιουργία των διευθύνσεων εδώ, καθώς και σε άλλες πηγές όπου μπορείτε να δείτε πώς ακριβώς προκύπτουν από private/public keys με την βοήθεια ενός elliptic curve.

(α) *Εμείς εδώ θα πειραματιστούμε με την αλυσίδα ”δοκιμών” του Bitcoin, δηλαδή το Bitcoin testnet και όχι το Bitcoin mainnet (προφανώς). Σε πρώτη φάση, κατασκευάστε ένα private key και την αντίστοιχη Bitcoin testnet διεύθυνσή του (διεύθυνση 1). Μπορείτε για αυτό να χρησιμοποιήσετε θεμελιώδεις συναρτήσεις, όπως RIPEMD-160, SHA-256 hash functions, και ECDSA signatures, αλλά* ***μην*** *χρησιμοποιήσετε έτοιμες βιβλιοθήκες για Bitcoin ή συναφή που να παρέχουν δημιουργία διευθύνσεων, έστω και τμηματικά. Ο σκοπός είναι να κάνετε τη δική σας διαδικασία δημιουργίας private key και (δημόσιας) διεύθυνσης, βάσει των προαναφερθέντων primitives.*

Ψάξτε σε μηχανή αναζήτησης για “bitcoin testnet faucet” για να λάβετε δωρεάν κάποια νομίσματα (τα ονομάζουμε tBTC από το test Bitcoin) στην παραπάνω διεύθυνσή σας, για να μπορείτε να εκτελέσετε τις επόμενες λειτουργίες και να πειραματιστείτε.

Έπειτα, δημιουργήστε μια διεύθυνση (διεύθυνση 2) με την εξής ιδιότητα: αντί για το κλασσικό RIPEMD160 (SHA256(...)) θα έχετε το RIPEMD160(SHA256(Αριθμός_Μητρώου_σας)). ***Παράδειγμα*** *(για να ελέγξετε τον κώδικά σας) αποτελεί η διεύθυνση*

`mkaWYS2DeChGv3u5tZMR59WSQkJEk61E3k`

που θα ήταν η ζητούμενη για τον ΑΜ 03112345. (το μηδενικό συμπεριλαμβάνεται στην κωδικοποίηση!) Είναι εύκολο να λάβετε τα χρήματα από την παραπάνω (διεύθυνση 2) που δημιουργήσατε εσείς βάσει του ΑΜ σας, ή όχι; Αν ναι, περιγράψτε ακριβώς με ποια βήματα μπορείτε να ανακτήσετε τα χρήματα από εκείνη τη διεύθυνση. Αν όχι, γιατί;

Στη συνέχεια, κατασκευάστε (με όποιον τρόπο επιθυμείτε, όχι κατ’ ανάγκη με κώδικα) ένα transaction με input(s) από την διεύθυνσή σας (διεύθυνση 1), με το οποίο θα αποστέλλετε στη διεύθυνση

`n3Uk2aQLXogYEbzYJnKyk9JSCeUAKVyB7q`

το ποσό των 0.01 tBTC, καθώς επίσης και στην (διεύθυνση 2) σας το ποσό των 0.01 tBTC, και εάν σας έχει περισσέψει κάτι σε ποσό πίσω στη (διεύθυνση 1) σας. Δηλαδή το transaction αυτό θα έχει 2 ή 3 outputs. Μην ξεχάσετε να αφήσετε ένα (πολύ μικρό, πχ 0.00001 tBTC) ποσό και για transaction fee, ειδάλλως το transaction σας μπορεί να μην γίνει confirm ποτέ! Ένα παράδειγμα τέτοιου transaction που είναι ήδη επάνω στο δίκτυο είναι αυτό με transaction id:

`0071ed6b67e53750b6fb54a536a5f7fee0764485be1f0044c388ac754b7de2df`

Συμπεριλάβετε τα βήματα που ακολουθήσατε, τον κώδικα που γράψατε, καθώς και όλες τις σχετικές διευθύνσεις (1, 2) και το παραπάνω transaction id, όπως εμφανίζονται επάνω στο δίκτυο και αποδεικνύουν την εκτέλεση των παραπάνω.

(β) *Τώρα θα κάνουμε κάτι σαν "mining" επάνω στο transaction που παραγάγατε. Συγκεκριμένα, σας ζητείται να βρείτε τέτοιο nonce (οποιουδήποτε μεγέθους σε πλήθος bits εσείς κρίνετε κατάλληλο) σε δεκαεξαδική μορφή, ώστε το concatenation του nonce και του transaction σας να έχουν ένα νέο, αρκούντως μικρό "transaction id". Προφανώς, τα παραπάνω nonce/transaction αναφέρονται σε hex μορφή, που υποδηλώνει το περιεχόμενο των αντίστοιχων bytes, και* ***όχι*** *ASCII/UTF-8 κωδικοποίηση, πχ το SHA-256 hash του hex "transaction" 0x0100deadbeef θα ήταν αυτό:*

`e2b6d72e359802bd6ffeb28aaeaf67947be21bb06961defbc8d84638e9628740`

και όχι αυτό:

`9f659f3c812af3c4e18cee90896d31d8245d5b5c7e52c79923425fbfb024ce24`

Μπορείτε να βρείτε την hex μορφή του transaction σας από κάποιον Bitcoin testnet explorer, αν δεν το έχετε ήδη. Γράψτε κώδικα που να προσδιορίζει κατάλληλο nonce με σκοπό τα πρώτα ***32*** *bits του νέου " transaction id" να είναι μηδενικά. Προφανώς, θα πρέπει το SHA256(SHA256(tx_hex)) να δίνει το αρχικό σας transaction id που βλέπετε επάνω στο δίκτυο, ενώ το SHA256(SHA256(nonce || tx_hex)) θα είναι το νέο "transaction id".* ***Λάβετε υπόψιν*** *σας ότι τα transactions είναι εσωτερικά κωδικοποιημένα με little-endian μορφή, ενώ το transaction id φαίνεται στον χρήστη ως big-endian, με λίγα λόγια θα χρειαστεί να αντιστρέψετε το "byte order" της εξόδου από little σε big έπειτα από την διπλή SHA256. Σιγουρευτείτε ότι το ακόλουθο παράδειγμα σάς δουλεύει σωστά: το transaction με id*

`0071ed6b67e53750b6fb54a536a5f7fee0764485be1f0044c388ac754b7de2df`

έχει hex content εδώ: https://jasoncs.eu.org/hex_tx.txt οπότε θα πρέπει να σας βγαίνει το εν λόγω transaction id με "κενό" nonce.

(γ) *Εκτελέστε το παραπάνω ερώτημα (β), όπου όμως τα πρώτα* ***42*** *bits θα πρέπει να είναι μηδενικά. Ενδέχεται να χρειαστεί να μεταβιβάσετε κάποιο τμήμα του κώδικά σας σε GPU, για να καταφέρετε να το υπολογίσετε χωρίς να περάσει αρκετός καιρός...*

Λύση:

1ος Τρόπος Επίλυσης:

(α) Το ερώτημα έχει επιλυθεί με 2 implementations, όπως φαίνεται και στο Notebook.

Η υλοποίηση του ερωτήματος φαίνεται στο συνοδευόμενο αρχείο crypto4.ipynb.

Σχολιασμός:

Παρατηρείται ότι το ιδιωτικό και το δημόσιο κλειδί που δημιουργήθηκαν με τη βοήθεια της βιβλιοθήκης ECDSA φαίνονται ίδια στην 1η υλοποίηση. Επιπλέον, παρατηρεί κανείς και τις 2 διευθύνσεις testnet που δημιουργήθηκαν. Η Address #2 που δημιουργήθηκε με βάση τον Α.Μ. είναι μία "useless" διεύθυνση, διότι δεν είναι γνωστό το ιδιωτικό κλειδί που της αντιστοιχεί κι έτσι, δεν μπορούν να ανακτηθούν τα κρυπτονομίσματα που στέλνονται σε αυτή. Σημειώνεται ότι δεν μπορεί να υπάρξει κανένα έγκυρο ιδιωτικό κλειδί, αφού το "δημόσιο" κλειδί που χρησιμοποιήθηκε έχει πολύ λιγότερα bytes από αυτά που παράγονται με την ECDSA. Έτσι, τα μόνα αποδεκτά είναι τα ζεύγη κλειδιών που δημιουργήθηκαν με αυτή από το Πρωτόκολλο του Bitcoin. Ακολουθώντας τη διαδικασία που περιγράφεται [6] εκτελέστηκε μια συναλλαγή κατά τα ζητούμενα.

(b) Η υλοποίηση του ερωτήματος φαίνεται στο συνοδευόμενο αρχείο.

Σχολιασμός:

Παρατηρείται ότι για bytes λιγότερα από 32 (π.χ. 24) έβγαλε έξοδο ενώ για τα 32 δεν τερμάτισε μετά από αρκετό χρόνο.

2ος Τρόπος Επίλυσης:

Αυτή η προσέγγιση ίσως να μην είναι η σωστότερη αλλά προστέθηκε λόγω πληρότητας αλλά και για να φανεί ένας άλλος τρόπος επίλυσης.

Η υλοποίηση του ερωτήματος φαίνεται στο συνοδευόμενο αρχείο crypto4_extra.ipynb.

(α) Τα κλειδιά δημιουργήθηκαν με την library της Python, την `secrets` η οποία προσφέρει ασφάλεια για την συγκεκριμένη λειτουργία.

Επίσης, μέσω της βιβλιοθήκης `bitcoinaddress` και του `module Wallet`, επιβεβαιώθηκε η σωστή δημιουργία της παραπάνω διεύθυνσης.

Σημειώνεται ότι είναι δύσκολη η λήψη των χρημάτων από αυτή την διεύθυνση εφόσον δεν είναι γνωστό το private key.

Τέλος, η αποστολή 0.01 Bitcoin στην δοσμένη διεύθυνση (όχι στην Address #2) γίνεται χρησιμοποιώντας την εντολή `send` της βιβλιοθήκης `bit`.

Resources:

[1] Διαφάνειες Μαθήματος & Βοηθητικό Υλικό

[2] https://en.bitcoin.it/wiki/Base58Check_encoding

[3] https://developer.bitcoin.org/devguide/transactions.html

[4] https://jasoncs.eu.org/hex_tx.txt

[5] https://live.blockcypher.com

[6] https://medium.com/coinmonks/estep-by-step-create-and-broadcast-a-bitcoin-transaction-on-testnet-588daacc2b7a