Blockchain for Supply Chain

Georgios Bolatoglou

Blockchain στο supply chain μιας εταιρίας

Η χρήση του blockhain για την αποθήκευση δεδομένων της εφοδιαστικής αλυσίδας αποφέρει πολλά πλεονεκτήματα σε μια εταιρία. Κάποια από αυτά είναι:

- Βοηθά στην καλύτερη διαχείριση και αποθήκευση των εμπορευμάτων της.
- Προσφέρει γρήγορη πρόσβαση/αποθήκευση στα αντίστοιχα έγγραφα που αφορούν κάποια συναλλαγή/παραλαβή εμπορεύματος.
- Μειώνει την χαρτογραφία
- Προσφέρει εύκολο εντοπισμό των εμπορευμάτων.
- Προσφέρει διαφάνεια(transparency) τόσο μεταξύ προμηθευτών και εταιρίας όσο και εσωτερικά της εταιρίας.
- Εξαλείφει την νοθεία/απάτη
- Δημιουργεί εμπιστευτικότητα μεταξύ εταιρίας, πελατών και προμηθευτών.
- Διαρκής Παρακολούθηση όλων των κινήσεων των εμπορευμάτων
- Εξάλειψη λαθών και λανθασμένων πράξεων από υπαλλήλους της εταιρίας

Έτσι, λοιπόν, καταλαβαίνουμε πως η χρήση blockchain μπορεί να περιορίσει ένα σωρό σημαντικά προβλήματα και, αναμφίβολα, να βελτιώσει την απόδοση και την παραγωγικότητα των υπαλλήλων της εταιρίας.

Δημιουργούμε ένα blockchain που θα μας επιτρέπει την καταχώρηση αυτών των δεδομένων που αφορούν την τροφοδοτική αλυσίδα της εταιρίας.

Ανάλυση Κώδικα

Με την συνάρτηση hashMe, η οποία χρησιμοποιεί την τεχνική κατακερματισμού sha256(έτοιμο από library), δημιουργούμε το hash του κάθε μπλοκ παραγόμενο από τα επιμέρους στοιχεία του κάθε μπλοκ. Σαν όρισμα παίρνει το μήνυμα που θέλουμε να κατακερματίσουμε. Ουσιαστικά, το hash του κάθε μπλοκ θα χρησιμοποιηθεί για την ένωση των μπλοκς μεταξύ τους και ως εκ τούτου την δημιουργία του blockchain.

```
def hashMe(msg=""):
    if type(msg)!=str:
        msg = json.dumps(msg,sort_keys=True)

    if sys.version_info.major == 2:
        return unicode(hashlib.sha256(msg).hexdigest(),'utf-8')
    else:
        return hashlib.sha256(str(msg).encode('utf-8')).hexdigest()
```

Δημιουργούμε την συνάρτηση makeEntry ώστε να γίνονται οι καταχωρήσεις στο κάθε μπλοκ και έπειτα, εφόσον οι καταχωρήσεις γίνουν αποδεκτές, στο blockchain. Στην προκειμένη περίπτωση υποθέτουμε, λοιπόν, ότι η εταιρία θέλει να κάνει καταχωρήσεις που να περιέχουν το ID του προϊόντος, την τιμή και την ποσότητά του, το όνομα του προμηθευτή και την ημερομηνία της συγκεκριμένης συναλλαγής τότε δημιουργούμε την συνάρτηση ώστε να επιστρέφει την συγκεκριμένες

παραμέτρους. Στα πλαίσια της απλής επίδειξης, θα χρησιμοποιήσουμε την συνάρτηση random που θα μας δίνει τυχαίες τιμές του ID προϊόντος, της ποσότητας, τιμής κλπ.
Έπειτα βάζουμε τις καταχωρήσεις σε ένα ξεχωριστό buffer, στο οποίο έχουμε ορίσει το όριο του. Έτσι μας δίνεται η δυνατότητα να περάσουμε στα blocks πολλαπλές καταχωρήσεις.

```
def makeEntry(maxValue=10):

    ID = random.randint(1,maxValue)
    Quantity   = 100 * random.randint(1,maxValue)
    Price = 0.1 * random.randint(1, maxValue)
    sup = str("Supplier")
    date = str( "18/01/2018")

    return {u'Date': date, u'Supplier':sup , u'Product_ID':ID,u'Quantity':Quantity, u'Price': Price}

entryBuffer = [makeEntry() for i in range(5)]
```

Ορισμός της αρχικής κατάστασης του blockchain (genesis block)
Στο genesis block τοποθετούμε το όνομα της εταιρίας και ότι αλλα δεδομένα θέλουμε, π.χ. ΑΦΜ κλπ., και έπειτα κάνουμε το πρώτο hash του blockchain μας!

```
name = str("bolatoglou AE")
state = {u'Company': name}  # Define the initial state
genesisBlock = [state]
genesisBlockContents = {u'blockNumber':0,u'parentHash':None,u'entriesCount':1,u'entry':genesisBlock}
genesisHash = hashMe( genesisBlockContents )
genesisBlock = {u'hash':genesisHash,u'contents':genesisBlockContents}
genesisBlockStr = json.dumps(genesisBlock, sort_keys=True)
```

Δημιουργία τον Blocks.
Δημιουργούμε την συνάρτηση makeBlock που παίρνει σαν εισόδους το buffer των καταχωρήσεων και την ήδη υπάρχον αλυσίδα.
Λειτουργεί ως εξής: Απ’ το τελευταίο block της αλυσίδας(parentBlock) λαμβάνει το hash του(parentHash) και το νούμερο του. Καταχωρεί στο καινούριο block που θέλουμε να δημιουργήσουμε τον αριθμό του(parentNumber+1), το hash του προηγούμενου block και στην συνέχεια όλες τις παραμέτρους της κάθε καταχώρησης που έχουμε ήδη περάσει στο buffer καταχωρήσεων. Οι παράμετροι κατακερματίζονται και έτσι βγαίνει το hash του εκάστοτε block.

```
def makeBlock(entries,chain):
    parentBlock = chain[-1]
    parentHash  = parentBlock[u'hash']
    blockNumber = parentBlock[u'contents'][u'blockNumber'] + 1
    entryCount    = len(entries)
    blockContents = {u'blockNumber':blockNumber,u'parentHash':parentHash,
                     u'entryCount':len(entries),'entries':entries}
    blockHash = hashMe( blockContents )
    block = {u'hash':blockHash,u'contents':blockContents}

    return block
```

Έλεγχος ορθότητας του blockchain
Εδώ δημιουργούμε την συναρτήση checkChain που καλεί τις υποσυναρτήσεις checkBlockHash και checkBlockValidity.

```
def checkChain(chain):

    checkBlockHash(chain[0])
    parent = chain[0]
    for block in chain[1:]:
        state = checkBlockValidity(block,parent)
        parent = block

    return state
```

Ουσιαστικά η checkChain κάνει τους εξής δύο ελέγχους:

1. Ελέγχει αν το hash του κάθε block ανταποκρίνεται με το αναμενόμενο hash(συνάρτηση checkBlockHash).

```
def checkBlockHash(block):

    expectedHash = hashMe( block['contents'] )
    if block['hash']!=expectedHash:
        print('Hash does not match contents of block %s'%
                               block['contents']['blockNumber'])
        return False

    return True
```

2. Ελέγχει αν το νούμερο του καινούριου block είναι το νούμερο του προηγούμενου + 1(checkBlockValidity). Αν π.χ. προσπαθήσουμε να βάλουμε το ίδιο μπλοκ και δεύτερη φορά(με τον ίδιο αριθμό του) θα μας πετάξει error!

```
def checkBlockValidity(block,parent):

    parentNumber = parent['contents']['blockNumber']
    parentHash   = parent['hash']
    blockNumber  = block['contents']['blockNumber']

    checkBlockHash(block) # Check hash integrity; raises error if inaccurate

    state = True
    if blockNumber!=(parentNumber+1):
        print("Wrong Block Number")
        state = False

    if block['contents']['parentHash'] != parentHash:
        state = False

    return state
```

Έλεγχος των καταχωρήσεων σε κάθε μπλοκ

Πριν γίνει η οποιαδήποτε καταχώρηση σε μπλοκ πρέπει να ελέγξουμε αν αυτή η καταχώρηση είναι αληθής, σωστή και αποδεκτή. Για την συγκεκριμένη επίδειξη δεν θα διεξαχθεί κάποιος έλεγχος καταχώρησης, μιας και όλες οι καταχωρήσεις βγαίνουν από τυχαίες τιμές που ανταποκρίνονται στις προδιαγραφές μας(συνάρτηση isValidEntry πάντα επιστρέφει True). Σε ένα πραγματικό περιβάλλον , όμως, θα μπορούσαμε να ελέγξουμε αν το ID του προϊόντος ανταποκρίνεται στα υπάρχοντα ή και ακόμα να επιβεβαιώνει κάποιος υπεύθυνος της εταιρίας την παραλαβή κάποιου προϊόντος. Επίσης θα μπορούσαμε να δημιουργήσουμε ένα σύστημα επιβεβαίωσης και από τις δύο μεριές, και απ' τον προμηθευτή και από την εταιρία, για την κάθε συναλλαγή.

Καταχώρηση blockchain σε .txt αρχείο
Δημιουργούμε την συνάρτηση writechain που έχει ως είσοδο το υπάρχουν blockchain και απλά γράφουμε **όλη** την αλυσίδα γραμμή γραμμή(block) στο .txt αρχείο.

```
def writechain(chain):
    f = open("Test.txt", "w");
    for line in chain:
        s = str(line);
        f.write(s)
        f.write("\n")
    f.close()
writechain(chain)
```

Επειδή, όμως θέλουμε να διατηρείται η αλυσίδα όπως είναι και απλά να περνάμε **κάθε καινούριο** block δημιουργούμε την συνάρτηση appendchain που κάνει ακριβώς αυτό, αφού πρώτα ελέγξει με την χρήση της checkchain αν ικανοποιεί τα δύο παραπάνω bullets.

```
def appendchain(block):
    chain.append(block)
    if checkChain(chain):
        fap = open("Test.txt", "a")
        blockstr = str(chain[-1])
        fap.write(blockstr)
        fap.write("\n")
        print("Blockchain on Node A is currently %s blocks long"%len(chain))
        fap.close()
    else:
        chain.pop()
        print("Enter block again")
```

Εκτέλεση προγράμματος

Παράδειγμα 1
Εμφανίζεται το μηνυμα "Blockchain on Node A is currently 1 blocks long". Το μοναδικό μας block είναι το genesis. Πληκτρολογώντας chain εμφανίζεται όλη το blockchain. Στην προκειμένη περίπτωση:

```
>>> chain
[{'contents': {'entry': {'Company': 'bolatoglou AE'}, 'entriesCount': 1, 'parentHash': None,
'blockNumber': 0}, 'hash': '508339257432099d49084103206018c3140d44307e489b11c0dee4fda4ac7488'
}]
```

Τώρα για να προσθέσουμε ένα μπλοκ καλούμε τις παραπάνω συνάρτησεις όπως παρουσιάζεται παρακάτω:
Καλούμε την makeEntry που επιλέγει κάποιες τυχαίες τιμές για τις παραμέτρους μας(product ID,date,supplier etc.):
entry1 = [makeEntry()]
Έπειτα, την makeBlock για να προσθέσουμε το block στο υπάρχουν blockchain:
newblock = makeBlock(entry1,chain)
Και τέλος κάνουμε append το blockchain με την appendchain, η οποία παράλληλα γράφει το blockchain sto .txt και ελέγχει την ορθότητα των blocks.
appendchain(new)

Και παίρνουμε το μήνυμα “Blockchain on Node A is currently 2 blocks long”.
Προσθέσαμε λοιπόν επιτυχώς ενα block στο blockchain.

Τώρα το blockchain έγινε:

[{'hash': '508339257432099d49084103206018c3140d44307e489b11c0dee4fda4ac7488', 'contents': {'blockNumber': 0, 'entry': {'Company': 'bolatoglou AE'}, 'entriesCount': 1, 'parentHash': None}}, {'hash': '152be4a69233ce97989089df08f44c0d17d57ca21ef0abc20eb97f574746423b', 'contents': {'entries': [{'Price': 1.0, 'Product_ID': 3, 'Quantity': 200, 'Date': '12/01/2018', 'Supplier': 'Supplier'}], 'blockNumber': 1, 'entryCount': 1, 'parentHash': '508339257432099d49084103206018c3140d44307e489b11c0dee4fda4ac7488'}}]

Όπως παρατηρούμε το καινούριο block εκτός από όλες τις πληροφορίες της «παραλαβής του προιόντος» έχει το hash του genesis block και με αυτόν τον τρόπο συνδέονται μεταξύ τους.

Παράδειγμα 2
Προσθέτοντας επιπλεόν block παίρνουμε ένα blockchain της παρακάτω μορφής. (επισημαίνεται το κάθε μπλοκ με διαφορετικό χρώμα)
[{'hash': '508339257432099d49084103206018c3140d44307e489b11c0dee4fda4ac7488', 'contents': {'blockNumber': 0, 'entry': {'Company': 'bolatoglou AE'}, 'entriesCount': 1, 'parentHash': None}},
{'hash': '152be4a69233ce97989089df08f44c0d17d57ca21ef0abc20eb97f574746423b', 'contents': {'entries': [{'Price': 1.0, 'Product_ID': 3, 'Quantity': 200, 'Date': '12/01/2018', 'Supplier': 'Supplier'}], 'blockNumber': 1, 'entryCount': 1, 'parentHash': '508339257432099d49084103206018c3140d44307e489b11c0dee4fda4ac7488'}},
{'hash':

'9be5aa725ea0139f653e54063a40801a2949120e2fe78177d0eb1e8421c0e0c1', 'contents': {'entries': [{'Price': 1.0, 'Product_ID': 5, 'Quantity': 900, 'Date': '12/01/2018', 'Supplier': 'Supplier'}, {'Price': 0.2, 'Product_ID': 3, 'Quantity': 500, 'Date': '12/01/2018', 'Supplier': 'Supplier'}, {'Price': 0.8, 'Product_ID': 2, 'Quantity': 600, 'Date': '12/01/2018', 'Supplier': 'Supplier'}, {'Price': 0.6000000000000001, 'Product_ID': 9, 'Quantity': 200, 'Date': '12/01/2018', 'Supplier': 'Supplier'}, {'Price': 1.0, 'Product_ID': 7, 'Quantity': 600, 'Date': '12/01/2018', 'Supplier': 'Supplier'}], 'blockNumber': 2, 'entryCount': 5, 'parentHash': '152be4a69233ce97989089df08f44c0d17d57ca21ef0abc20eb97f574746423b'}}, {'hash': 'a71ec67444e5cf33f778c83dcc6a4ed43e2c9d9833df77eba05b4e12e557d2af', 'contents': {'entries': [{'Price': 0.8, 'Product_ID': 4, 'Quantity': 900, 'Date': '12/01/2018', 'Supplier': 'Supplier'}], 'blockNumber': 3, 'entryCount': 1, 'parentHash': '9be5aa725ea0139f653e54063a40801a2949120e2fe78177d0eb1e8421c0e0c1'}}]

Τώρα, το blockchain μας αποτελείται από 4 μπλοκς. Παρατηρούμε πως στο 3ο block έχουν γίνει πολλαπλές καταχωρήσεις('entryCount': 5) κάτι που όπως έχει προαναφερθεί γίνεται πολύ εύκολα με την δημιουργία ενός απλού buffer καταχωρήσεων entry2 = [makeEntry() for i in range(5)] (αντί για [makeEntry()] σκέτο). Επίσης, όλα τα μπλοκ συνδέονται μεταξύ τους με τα parentHash! Και οι αριθμοί του κάθε μπλοκ ανταποκρίνονται στην σειρά τους.

Παράδειγμα 3
Τώρα θα ελέγξουμε τι γίνεται στην περίπτωση εσφαλμένης καταχώρησης μπλοκ. Για παράδειγμα θα προσπαθήσουμε να βάλουμε κάποιο μπλοκ που έχει ήδη μπει(και έχει τον ίδιο blockNumber).

```
>>> newEntry = [makeEntry()]
>>> newBlock = makeBlock(newEntry,chain)
>>> appendchain(newBlock)
Blockchain on Node A is currently 3 blocks long
>>> appendchain(newBlock)
Wrong Block Number
Enter block again
>>>
```

Όπως βλέπουμε δεν μας αφήνει να βάλουμε στο blockchain το εσφαλμένο block. Έπειτα προσθέτουμε άλλο ένα μπλοκ με 3 καταχωρήσεις και ελέγχουμε την κατάσταση της αλυσίδας.

```
>>> new1 = [makeEntry() for i in range(3)]
>>> block1 = makeBlock(new1,chain)
>>> appendchain(block1)
Blockchain on Node A is currently 4 blocks long
>>> checkChain(chain)
True
>>> len(chain)
4
```

Επιβεβαιώνεται έτσι ότι το λάθος block δεν μπαίνει ποτέ.

Με αυτόν τον κώδικα προσεγγίζουμε σε καλό βαθμό πως θα λειτουργούσε το blockchain για την τροφοδοτική αλυσίδα μια εταιρίας. Παρουσιάζονται τα βασικότερα χαρακτηριστικά του και τα περισσότερα στοιχεία που το διέπουν και το αναδεικνύουν μια τόσο καινοτόμο δομή αποθήκευσης δεδομένων.