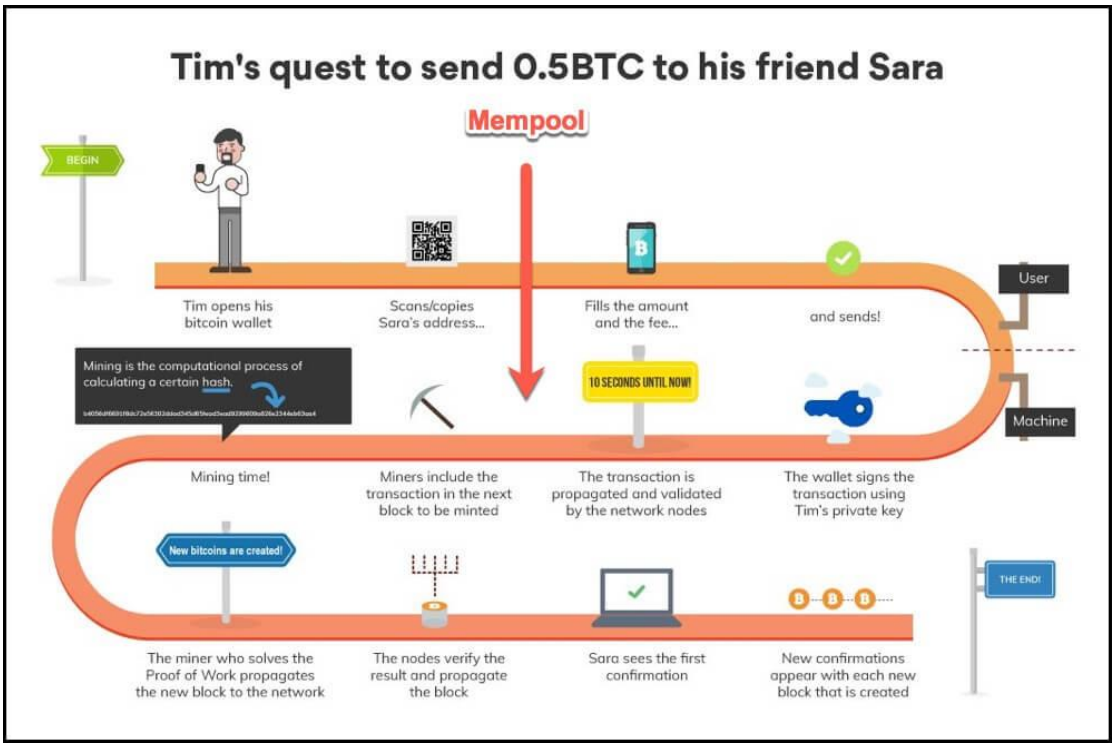
העברות

חלק העוסק בהעברות הכספים עצמם הוא החלק המורכב והגדול ביותר לכל הקשור לביטקוין.
העברות הכספים חייבות להיות מדויקות, ייחודיות, ללא שגיאות ככל האפשר ומספיק מצומצמות כדי שכל העברה כזאת תוכל להגיע לכל רשת הביטקוין במהירות ובקלות.



ראשית, נבין שבכל העברה – מועבר **כל** סכום הביטקוין שנמצא בכתובת (לא משנה בסכום ההעברה) אלא שרק סכום העברה מועבר לכתובת הרצויה ושאר הסכום מועבר אל הכתובת הציבורית של המשלם חזרה.

כדי להבין את הרציונל טוב יותר, נחשוב על דוגמה –
אדם רוצה לקנות תפוח שעולה 2 שקלים אבל יש לו רק שטר של 20, ישלם למוכר את כל ה20,
המוכר יקבל רק 2 שקלים מתוכם ויחזיר 18 למשלם.
נתבונן בנתוני העברה שיש במוניטור אונליין:

Transaction View information about a bitcoin transaction

b6aef433a739a8bf73bbfa51e8d35cc75f03a748ee41b632e34790f9aee9daab

1U81JRUEFiRwk6cyatedV9ETDxtbXMP9J	→	35pHZC6cMBkoUVBhJTJBfWDAkjeP327Zda	0.00472944 BTC
		1U81JRUEFiRwk6cyatedV9ETDxtbXMP9J	43.34480381 BTC

Unconfirmed Transaction! 43.34953325 BTC

Summary	
Size	518 (bytes)
Weight	2072
Received Time	2019-05-29 06:10:58
Visualize	View Tree Chart

Inputs and Outputs	
Total Input	43.34983145 BTC
Total Output	43.34953325 BTC
Fees	0.0002982 BTC
Fee per byte	57.568 sat/B
Fee per weight unit	14.392 sat/WU
Estimated BTC Transacted	0.00472944 BTC
Scripts	Show scripts & coinbase

נראה שבכתובות הנמענים יש את כתובת השולח גם היא, ואליה מועבר שאר הסכום שנמצא בכתובת.

נוכל לראות שאפשר להסיק חלק מהנתונים מבלי הצגה מפורשת שלהם -

למשל, מתוך הסכום הכולל שיש בכתובת, נוכל לחסר את הסכום הנשלח בפועל (המופיע בירוק) ולקבל את העמלה (Fees) ששולמה עבור העסקה.

את הסכום הנשלח בפועל עצמו נוכל גם כן להסיק מתוך הכתובות המופיעות בנמענים ולסכום אותם. נוכל כבר לראות שיש שדות מיותרים. ברשת תקשורת עמוסה כמו ביטקוין נרצה לצמצם ככל שניתן את תכולת ההעברות – לכן:

היה אפשר לחשוב שכל הנתונים המוצגים מעלה הם חלק מנתוני ההעברה ומופיעים בכל מסר של העברה מחדש. למעשה, חלק מהנתונים המוצגים אינם חלק מנתוני ההעברות, ומוצגים ע"י תוכנות לקוח שמסיקות את הנתונים בעיבוד שיכבה גבוהה (high level).

נתונים כמו כתובת המקור וכתובת הנמען, הסכום המועבר ע"י המשלם אינם מוצגים בכלל ע"י הביטים המועברים בפועל.

בביטים, אין מטבעות, אין "קבלה", אין מאזן כספים ואין כתובות(!), כל הדברים האלה מועבדים מתוך ההעברה ברמה גבוהה יותר של ארנק וממשק משתמש כדי להציג את הנתונים באופן מובן יותר ויעיל יותר לשימוש.

נתונים המועברים בפועל להלן:

General format of a Bitcoin transaction (inside a block)

Field	Description	Size
Version no	currently 1	4 bytes
Flag	If present, always 0001, and indicates the presence of witness data	optional 2 byte array
In-counter	positive integer VI = VarInt	1 - 9 bytes
list of inputs	the first input of the first transaction is also called "coinbase" (its content was ignored in earlier versions)	<in-counter>-many inputs
Out-counter	positive integer VI = VarInt	1 - 9 bytes
list of outputs	the outputs of the first transaction spend the mined bitcoins for the block	<out-counter>-many outputs
Witnesses	A list of witnesses, 1 for each input, omitted if flag above is missing	variable, see Segregated_Witness
lock_time	if non-zero and sequence numbers are < 0xFFFFFFFF: block height or timestamp when transaction is final	4 bytes

Principle example of a Bitcoin transaction with 1 input and 1 output only

Data

```
Input:
Previous tx: f5d8ee39a430901c91a5917b9f2dc19d6d1a0e9cea205b009ca73dd04470b9a6
Index: 0
scriptSig: 304502206e21798a42fae0e854281abd38bacd1aeed3ee3738d9e1446618c4571d10
90db022100e2ac980643b0b82c0e88ffdfec6b64e3e6ba35e7ba5fdd7d5d6cc8d25c6b241501

Output:
Value: 5000000000
scriptPubKey: OP_DUP OP_HASH160 404371705fa9bd789a2fcd52d2c580b65d35549d
OP_EQUALVERIFY OP_CHECKSIG
```

https://bitcoin.org/en/developer-reference#getrawtransaction (הסבר על שדות ההעברה האמיתיים).

החלוקה של הביטים של העברה כלשהי מתחלקת ל- כותרת (2 שדות), קלט (4 שדות) ופלט (2 שדות לכל כתובת).

כדי לראות את החלוקה של ההעברה, נוכל לכתוב בbitcoin core:

```
$ bitcoin-cli getrawtransaction
```

וכיוון שלא ציינו מספר העברה (הארנק יכול לתת מידע רק עבור העברות שביצע אלא אם נגדיר במפורש אחרת) נקבל פורמט להעברה:

```
{
  "in_active_chain": b, (bool) Whether specified block is in the active chain or not (on
ly present with explicit "blockhash" argument)
  "hex" : "data",       (string) The serialized, hex-encoded data for 'txid'
  "txid" : "id",        (string) The transaction id (same as provided)
  "hash" : "id",        (string) The transaction hash (differs from txid for witness tra
nsactions)
  "size" : n,             (numeric) The serialized transaction size
  "vsize" : n,            (numeric) The virtual transaction size (differs from size for
witness transactions)
  "weight" : n,           (numeric) The transaction's weight (between vsize*4-3 and vsiz
e*4)
  "version" : n,          (numeric) The version
  "locktime" : ttt,       (numeric) The lock time
  "vin" : [               (array of json objects)
     {
       "txid": "id",    (string) The transaction id
       "vout": n,         (numeric)
       "scriptSig": {     (json object) The script
         "asm": "asm",  (string) asm
         "hex": "hex"   (string) hex
       },
       "sequence": n      (numeric) The script sequence number
       "txinwitness": ["hex", ...] (array of string) hex-encoded witness data (if any)
     }
     ,...
  ],
  "vout" : [              (array of json objects)
     {
       "value" : x.xxx,            (numeric) The value in BTC
       "n" : n,                    (numeric) index
       "scriptPubKey" : {          (json object)
         "asm" : "asm",          (string) the asm
         "hex" : "hex",          (string) the hex
         "reqSigs" : n,            (numeric) The required sigs
         "type" : "pubkeyhash",  (string) The type, eg 'pubkeyhash'
         "addresses" : [           (json array of string)
           "address"        (string) bitcoin address
           ,...
         ]
       }
     }
     ,...
  ],
  "blockhash" : "hash",   (string) the block hash
  "confirmations" : n,      (numeric) The confirmations
  "time" : ttt,             (numeric) The transaction time in seconds since epoch (Jan 1
 1970 GMT)
  "blocktime" : ttt         (numeric) The block time in seconds since epoch (Jan 1 1970
GMT)
}
```

נוכל לראות שהעברה מתחלקת ל3 חלקים: headers לפני ואחרי, מערך קלט (vin), מערך פלט (vout).


```
{
  "version": 1,
  "locktime": 0,
  "vin": [
    {
      "txid":
"7957a35fe64f80d234d76d83a2a8f1a0d8149a41d81de548f0a65a8a999f6f18",
      "vout": 0,
      "scriptSig" :
"3045022100884d142d86652a3f47ba4746ec719bbfbd040a570b1deccbb6498c75c4ae24cb02204
b9f039ff08df09cbe9f6addac960298cad530a863ea8f53982c09db8f6e3813[ALL]
0484ecc0d46f1918b30928fa0e4ed99f16a0fb4fde0735e7ade8416ab9fe423cc5412336376789d1
72787ec3457eee41c04f4938de5cc17b4a10fa336a8d752adf",
      "sequence": 4294967295
    }
  ],
  "vout": [

      {
        "value": 0.01500000,
        "scriptPubKey": "OP_DUP OP_HASH160
 ab68025513c3dbd2f7b92a94e0581f5d50f654e7 OP_EQUALVERIFY OP_CHECKSIG"
      },
      {
        "value": 0.08450000,
        "scriptPubKey": "OP_DUP OP_HASH160
 7f9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a8 OP_EQUALVERIFY OP_CHECKSIG",
      }
    ]
  }
```

על מנת להבין טוב יותר מאיפה הנתונים נשאבים ואת פעולת ה"ארנק", נצטרך להבין את מושג הUTXO – פלט יתרת ההעברה.

הUTXO הוא יחידה של ביטקוין קבועה. לחלק אותה לחלקים מתאפשר רק באמצעות הארנק. מכאן, כששולחים סכום מכתובת מסויימת, אין טעם לציין את יתרת המטבע. יתרת המטבע לקוחה מההעברה הקודמת והיא מההעברה שקדמה לה עד יצירת המטבע.

חלוקה של העברה לפי ביטים והצגה של מספר סיריאלי:
https://learnmeabitcoin.com/glossary/transaction-data

פלט:
הפלט מורכב מ3 שדות לכל כתובת שאליה מועבר כסף:
- סכום
- אורך הסקריפט הנועל
- סקריפט הנועל עצמו

קלט:
הקלט מורכב מ4 שדות מכל כתובת שממנה מועבר כסף:
- מצביע להעברה קודמת שממנה ניקח את המטבע UTXO.
- אינדקס הoutput בהעברה הקודמת
- סקריפט המפענח שיאשר שהעברה הקודמת מקושרת
- מפתח ציבורי של השולח

סקריפטים:
מוגדרים כבעלי שלמות טיורינג. בהעברה ישנם 2 סקריפטים – סקריפט הקלט וסקריפט ופלט.

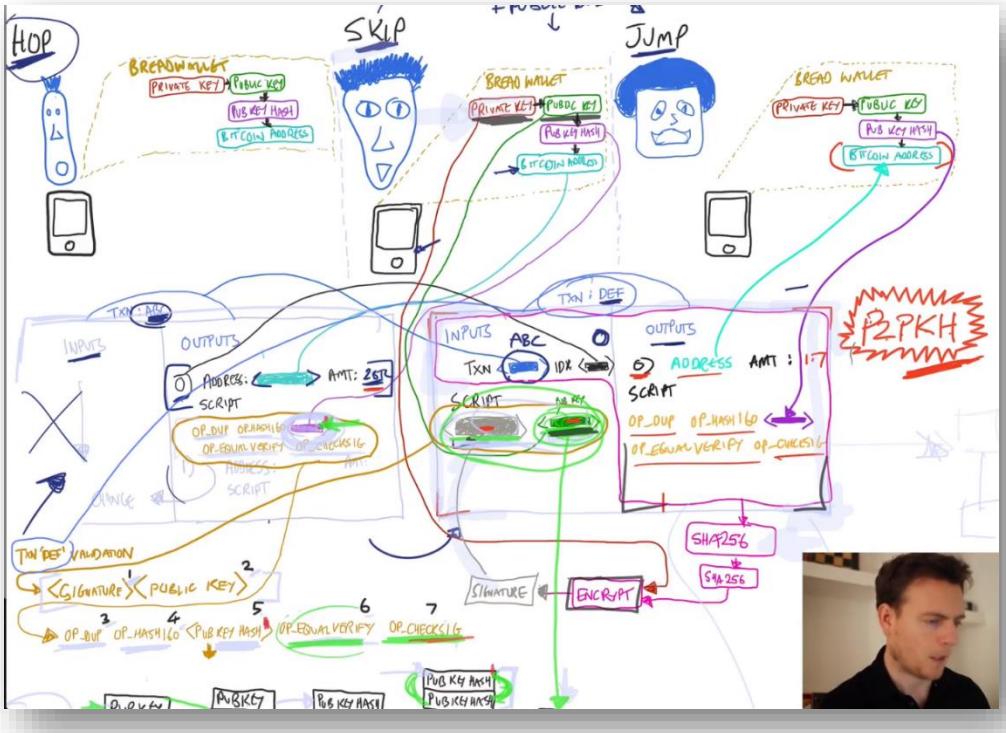
2 3 OP_ADD 2 OP_MUL 1 OP_ADD 11 OP_EQUAL
הוא דוגמא לסקריפט שבודק את התנאי: $(2+3)*2+1==11$

Op_checksig היא הפונקציה הבאה:

```
bool TransactionSignatureChecker::CheckSig(const vector<unsigned char>& vchSigIn, const vector<unsigned char>& vchPubKey, const CScript& sc
{
    CPubKey pubkey(vchPubKey);
    if (!pubkey.IsValid())
        return false;

    // Hash type is one byte tacked on to the end of the signature
    vector<unsigned char> vchSig(vchSigIn);
    if (vchSig.empty())
        return false;
    int nHashType = vchSig.back();
    vchSig.pop_back();

    uint256 sighash = SignatureHash(scriptCode, *txTo, nIn, nHashType);

    if (!VerifySignature(vchSig, pubkey, sighash))
        return false;

    return true;
}
```

הבודקת את חתימת ההעברה ע"י השולח עם המפתח הציבורי שלו.
החתימה היא תוכן הoutput שמועבר בפעמיים sha256 ומוצפן באמצעות המפתח הפרטי.
הפונקציה בעצם מפענחת את החתימה באמצעות המפתח הציבורי של השולח ובכך מקבלת Hash כלשהו שצריך להיות שווה לפעמיים SHA256 על תוכן הoutput – אם שווה מחזיר אמת, אחרת שקר.
https://en.bitcoin.it/wiki/Script
https://bitcoin.org/en/developer-reference#opcodes

חתימה



פעולת החתימה היא פעולה על עקומה אליפטית כך שבנוסף למפתח הפרטי של השולח, הוא יוצר זוג חדש של מפתחות פרטי וציבורי זמניים עבור כל העברה.
An ECDSA signature is a pair (r,s) where r is the X coordinate of kG, and s = (m+r*x)/k (where k=nonce, m=message hash, x=private key, G=curve generator)

מלבד הרצת הסקריפט (כאמור, בכל בדיקת העברה הפרוטוקול מריץ את הסקריפט על מנת לוודא אמינות) ישנן עוד **19** הרצות עבור **כל** העברה בכל בלוק, ועוד **19** הרצות עבור כל בלוק בשרשרת הביטקוין.
https://en.bitcoin.it/wiki/Protocol_rules#cite_note-7

20 פעולות עבור כל העברה בכל בלוק:

1. בדיקת נכונות תחבירית (עבור מפתח ציבורי)
2. וידוא שקיימת לפחות רשומה אחת ברשימת הקלט וברשימת הפלט (אינן רשימות ריקות)
3. גודל הבלוק קטן מהגודל המקסימלי המוגדר (משתנה)
4. סכום של כל פלט והסכום הכללי הם בגבולות הסכומים האפשריים (legal money range)
5. אף קלט אינו עם הגיבוב 0 (למעט מטבע הבסיס שניתן כפרס על כרייה)
6. nLockTime <= INT_MAX (זהו שדה חיצוני בheader של פקטת ביטקוין וגם בהעברה יש לפחות 100 בייט והחתימות הכלולות בשדה חתימה בהעברות סטנדרטיות גדול מ2
7. סקריפט אינו סטנדרטי. מבצע פעולות מורכבות מלבד להוציא מהמחסנית ולהכניס אליה וכו'
8. בודק שאין העברה דומה ב"בריכת ההעברות" או בשרשרת הבלוקצ'יין
9. אם ההעברה הקודמת בוזבזה כבר בהוצאה אחרת
10. אם הקישור להעברה הקודמת לא קיים ההעברה נוספת ל"העברות יתומות"
11. אם המטבע הוא מטבע בסיס (יש מאין) מספר האישורים להעברות שנכללו בבלוק הכרייה הוא לפחות 100
12. עבור כל קלט, בדיקה שהקישור לפלט רלוונטי אכן קיים (כפול עבור 8? עבור 9?)
13. בדיקה שערכי הקלט הם בטווח הסכום האפשרי (כמו 4 רק שכאן נדרש ללכת לקישור העברה האחרונה על מנת להגיע לסכום הקלט)
14. בדיקה שערכי הפלט קטנים מערכי הקלט (לא מבוזבז כסף שלא קיים)

15. ההעברה נדחית אם סכום העמלה קטן מכדי להיכנס לבלוק
16. **וידוא הסקריפט הקיים בהעברה שאכן מביא תוצאה נכונה (True)**
17. הוסף את ההעברה ל"בריכת ההעברות"
18. הוסף את ההעברה לארנק כהעברה שנכרית
19. קשר את ההעברה לרשת הביטקוין (peer)
20. עבור כל עסקה יתומה, הרץ את כל השלבים מחדש באופן רקורסיבי