Bitcoin Forum

Bitcoin => Development & Technical Discussion => Topic started by: Gavin Andresen on June 17, 2010, 11:38:31 AM

Title: **Transactions and Scripts: DUP HASH160 ... EQUALVERIFY CHECKSIG**
Post by: **Gavin Andresen** on **June 17, 2010, 11:38:31 AM**

So I'm writing a little tool that dissects the Bitcoin wallet.dat, mainly because I want to understand better exactly how Bitcoin works.

And I see that the outputs of transactions have a value (number of bitcoins) and a bunch of bytes that are run through the little Forth-like scripting language built in to bitcoin. E.g.:
['TxOut: value: 100.00 Script: DUP HASH160 6fad...ab90 EQUALVERIFY CHECKSIG']

First: it make me a little nervous that bitcoin has a scripting language in it, even though it is a really simple scripting language (no loops, no pointers, nothing but math and crypto). It makes me nervous because it is more complicated, and complication is the enemy of security. It also makes it harder to create a second, compatible implementation. But I think I can get over that.

Looking at the code, new transactions are verified by pushing the signature an then public key on the interpreter's stack and then running the TxOut script (did I get that right?).

Could I write code to create transactions with any valid script in the TxOut?
E.g. could I create a TxOut with a script of: OP_2DROP OP_TRUE
... to create a coin that could be spent by anybody?

And is flexibility in the types of coins created the reason it is coded this way?

Title: **Re: Transactions and Scripts: DUP HASH160 ... EQUALVERIFY CHECKSIG**
Post by: **satoshi** on **June 17, 2010, 06:46:08 PM**

The nature of Bitcoin is such that once version 0.1 was released, the core design was set in stone for the rest of its lifetime. Because of that, I wanted to design it to support every possible transaction type I could think of. The problem was, each thing required special support code and data fields whether it was used or not, and only covered one special case at a time. It would have been an explosion of special cases. The solution was script, which generalizes the problem so transacting parties can describe their transaction as a predicate that the node network evaluates. The nodes only need to understand the transaction to the extent of evaluating whether the sender's conditions are met.

The script is actually a predicate. It's just an equation that evaluates to true or false. Predicate is a long and unfamiliar word so I called it script.

The receiver of a payment does a template match on the script. Currently, receivers only accept two templates: direct payment and bitcoin address. Future versions can add templates for more transaction types and nodes running that version or higher will be able to receive them. All versions of nodes in the network can verify and process any new transactions into blocks, even though they may not know how to read them.

The design supports a tremendous variety of possible transaction types that I designed years ago. Escrow transactions, bonded contracts, third party arbitration, multi-party signature, etc. If Bitcoin catches on in a big way, these are things we'll want to explore in the future, but they all had to be designed at the beginning to make sure they would

be possible later.

I don't believe a second, compatible implementation of Bitcoin will ever be a good idea. So much of the design depends on all nodes getting exactly identical results in lockstep that a second implementation would be a menace to the network. The MIT license is compatible with all other licenses and commercial uses, so there is no need to rewrite it from a licensing standpoint.

Title: **Re: Transactions and Scripts: DUP HASH160 ... EQUALVERIFY CHECKSIG**
Post by: **laszlo** on **June 17, 2010, 06:50:31 PM**

How long have you been working on this design Satoshi? It seems very well thought out, not the kind of thing you just sit down and code up without doing a lot of brainstorming and discussion on it first. Everyone has the obvious questions looking for holes in it but it is holding up well :)

Title: **Re: Transactions and Scripts: DUP HASH160 ... EQUALVERIFY CHECKSIG**
Post by: **Gavin Andresen** on **June 17, 2010, 07:58:14 PM**

Quote from: satoshi on June 17, 2010, 06:46:08 PM

> I don't believe a second, compatible implementation of Bitcoin will ever be a good idea. So much of the design depends on all nodes getting exactly identical results in lockstep that a second implementation would be a menace to the network. The MIT license is compatible with all other licenses and commercial uses, so there is no need to rewrite it from a licensing standpoint.

Good idea or not, SOMEBODY will try to mess up the network (or co-opt it for their own use) sooner or later. They'll either hack the existing code or write their own version, and will be a menace to the network.

I admire the flexibility of the scripts-in-a-transaction scheme, but my evil little mind immediately starts to think of ways I might abuse it. I could encode all sorts of interesting information in the TxOut script, and if non-hacked clients validated-and-then-ignored those transactions it would be a useful covert broadcast communication channel.

That's a cool feature until it gets popular and somebody decides it would be fun to flood the payment network with millions of transactions to transfer the latest Lady Gaga video to all their friends...

Title: **Re: Transactions and Scripts: DUP HASH160 ... EQUALVERIFY CHECKSIG**
Post by: **satoshi** on **June 18, 2010, 04:17:14 PM**

A second version would be a massive development and maintenance hassle for me. It's hard enough maintaining backward compatibility while upgrading the network without a second version locking things in. If the second version screwed up, the user experience would reflect badly on both, although it would at least reinforce to users the importance of staying with the official version. If someone was getting ready to fork a second version, I would have to air a lot of disclaimers about the risks of using a minority version. This is a design where the majority version wins if there's any disagreement, and that can be pretty ugly for the minority version and I'd rather not go into it, and I don't have to as long as there's only one version.

I know, most developers don't like their software forked, but I have real technical reasons in this case.

Quote from: gavinandresen on June 17, 2010, 07:58:14 PM

> I admire the flexibility of the scripts-in-a-transaction scheme, but my evil little mind immediately starts to think of ways I might abuse it. I could encode all sorts of interesting information in the TxOut script, and if non-hacked clients validated-and-then-ignored those transactions it would be a useful covert broadcast communication channel.

> That's a cool feature until it gets popular and somebody decides it would be fun to flood the payment network with millions of transactions to transfer the latest Lady Gaga video to all their friends...

That's one of the reasons for transaction fees. There are other things we can do if necessary.

> Quote from: laszlo on June 17, 2010, 06:50:31 PM
>
> How long have you been working on this design Satoshi? It seems very well thought out, not the kind of thing you just sit down and code up without doing a lot of brainstorming and discussion on it first. Everyone has the obvious questions looking for holes in it but it is holding up well :)

Since 2007. At some point I became convinced there was a way to do this without any trust required at all and couldn't resist to keep thinking about it. Much more of the work was designing than coding.

Fortunately, so far all the issues raised have been things I previously considered and planned for.

Title: **Re: Transactions and Scripts: DUP HASH160 ... EQUALVERIFY CHECKSIG**
Post by: **laszlo** on **June 18, 2010, 04:50:52 PM**

Cool, I just hope you don't take it the wrong way when guys like me and Gavin are trying to shoot holes in it.. I think we're all here to see this thing develop and turn into something bigger but we're also interested in finding ways to break it (so it can be fixed). It seems like you have done a great job covering the bases so far because you've been able to add features without breaking compatibility.

Title: **Re: Transactions and Scripts: DUP HASH160 ... EQUALVERIFY CHECKSIG**
Post by: **teppy** on **June 23, 2010, 03:07:11 PM**

This is interesting, and is a possible way to embed messages inside anonymous payments, at the cost of transaction fees.

So if a payment contains a No-op string that says "Message encrypted for Public Key [xxx]: [yyyyy]" then that gets passed along to the destination? Or even "Cleartext message to recipient: [zzzzz]."

Of course the content of these messages - the [xxx] or [yyyyy] have nothing to do with Bitcoin, but they could be used as part of a layer on top of Bitcoin.

The reason I'm so interested in embedding messages is that they allow use of a "static" anonymity network like Freenet, rather than a "live" network like Tor or I2P. Live networks have exit nodes, a few of which can be compromised. If a government compromises 1% of all exit nodes, then they have a small chance of figuring out where a given site is hosted. So to use my Heroin Store example, they would send an N byte message to the store and watch all of their compromised exit nodes for N byte messages. They'd do this over a few hours, watching where N byte messages got sent to, eventually discovering the IP address of the store.

In a network like Freenet, data just floats around - there is no one machine where a website (for example) lives. You publish data to the cloud, and as long as people access it periodically, it stays around.

Title: **Re: Transactions and Scripts: DUP HASH160 ... EQUALVERIFY CHECKSIG**
Post by: **bytemaster** on **August 03, 2010, 03:12:01 AM**

So I have gathered enough information to see that scripts are intended to provide extensibility.

If bitcoins represent one type of asset divided into 23,000,000 "shares" which are

distributed via generation.

I would like to enable users to create other custom assets. The asset would have a globally unique hash and could ben sent / recved to addresses. For example, a company could issue "shares" that could then be exchanged and verified by bitcoin. A game engine could issue its own type of currency and have it tracked.

Is anything like this even remotely possible with scripts or would that require a breaking change? Is it even possible to 'upgrade' the network to support such an option at this point?

Title: **Re: Transactions and Scripts: DUP HASH160 ... EQUALVERIFY CHECKSIG**
Post by: **Gavin Andresen** on **August 03, 2010, 11:44:19 AM**

Quote from: bytemaster on August 03, 2010, 03:12:01 AM

> I would like to enable users to create other custom assets. The asset would have a globally unique hash and could ben sent / recved to addresses. For example, a company could issue "shares" that could then be exchanged and verified by bitcoin. A game engine could issue its own type of currency and have it tracked.
>
> Is anything like this even remotely possible with scripts or would that require a breaking change? Is it even possible to 'upgrade' the network to support such an option at this point?

I don't think you need scripts to do something like that.

Just send a bitcoin to yourself, and then declare that transaction is the "root transaction for My Valuable Asset."

You'd need a custom client that only accepted or spent transactions that could be entirely traced back to that Root Transaction (not hard, all transactions can be tracked back). And shows fractions of that coin as units of your custom currency. And you'd probably want to make some other changes so your users didn't accidently (or purposely) mix Your Valuable Asset coin with regular bitcoins (like a custom wallet and different scheme for generating payment addresses).