txid malleability and segregated witness

2018-03-19
Tadge Dryja
schedule stuff

hope people were able to attend the expo, it was fun

office hours tomorrow

pset03 due wed 21st
I'm endorsing an ICO - Anne's intermittent cookie offering
malleability
ability to deform under pressure

bitcoin is modeled after gold, which is the most malleable metal; thus bitcoin is a highly malleable system
malleability

actually, it's when adversaries can modify ciphertexts, messages, signatures, etc and things still 'work'

In the case of bitcoin, transactions can be changed and still be valid!
tx asymmetry
recall the tx format; inputs and outputs don't look the same

<table>
<thead>
<tr>
<th>txid:index (36B)</th>
<th>script (25B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>signature (100B)</td>
<td>amount (8B)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>txid:index</th>
</tr>
</thead>
<tbody>
<tr>
<td>signature</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>script (pubkey)</th>
</tr>
</thead>
<tbody>
<tr>
<td>amount</td>
</tr>
</tbody>
</table>
what gets signed
sign the whole transaction, inputs and outputs
But inputs contain signatures
and you can't sign the signature
what gets signed
remove the signature fields, sign,
then put signatures in
change any bit of the signed message,
and the signature is invalid
what gets signed
remove the signature fields, sign,
then put signatures in
change any bit of the signed message,
and the signature is invalid
but txid is the hash of the message,
including signatures
signature malleability

3rd party malleability

leading zeros

"low s" can flip the sign of the signature and it's still valid
signature malleability

1st party

recall signing uses a nonce k

use a different k, different
signature on the same message

RFC6979 defines deterministic k algo, but not detectable by observers
so you've been malleated

taxid changes

outputs are still the same

which inputs also still the same

so no big deal?
so you've been malleated
in most cases, some wallets have trouble
broadcast tx 2d5cac, which never got confirmed
Instead malleated to 9cba3e
Wallet shows unconfirmed forever
dependent txs
spending unconfirmed change output
from tx1 7feec1. Sign and broadcast tx2
tx1 changes to b2068c!
tx2 invalid, refers to txid which can never be confirmed
dependent txs
txid change is annoying but can refer to malleated txids and re-sign
what if you can't re-sign?
dependent txs

taxid change is annoying but can refer to malleated txids and re-sign

what if you can't re-sign?

multisig, pre-signed txs

very important in payment channels / lightning network
different ideas
use non-malleable signatures?

lamport signatures were non-malleable
but many useful signature schemes are malleable
different ideas

don't sign your inputs at all!

I really like this idea, allows many fun features

but dangerous: allows signature replays. Sign once, use many
how to fix malleability?
find out!

after intermission
segregated witness
strange name for straightforward idea
Don't include signatures in txids; txs are now defined by input pointers and outputs only
signature changes but txid doesn't
But backwards compatibility...?
soft fork

would have been easier to start out this way

But doable as a soft fork

but how...?

make outputs which don't require signatures
segwit version numbers

output script:

0 <pubkey hash>

sig script:

(nothing)
segwit version numbers
output script:
0 <pubkey hash>
sig script:
(nothing)
<pubkey hash> on top of stack; non-zero, coins move!
pubkey hash template
output script:
0 <pubkey hash>
now means pay to pubkey hash
but put the signature somewhere else
the "witness" field, old software never sees
## new tx type

## old tx format

<table>
<thead>
<tr>
<th>txid:index (36B)</th>
<th>script (25B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>signature (100B)</td>
<td>amount (8B)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>txid:index</th>
<th>script (pubkey)</th>
</tr>
</thead>
<tbody>
<tr>
<td>signature</td>
<td>amount</td>
</tr>
</tbody>
</table>
new tx type
new tx format

<table>
<thead>
<tr>
<th>txid:index (36B)</th>
<th>script (25B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>signature (0B)</td>
<td>amount (8B)</td>
</tr>
<tr>
<td>[witness]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>txid:index</th>
<th>script (pubkey)</th>
</tr>
</thead>
<tbody>
<tr>
<td>signature [witness]</td>
<td>amount</td>
</tr>
</tbody>
</table>
omit to old nodes
when people ask for witness txs, include the witness
when they just ask for txs, give it to them without the witness field
omit to old nodes

old nodes: signature can't change; there isn't one!

new nodes: signature can change, but doesn't affect txid
(dis)agreement
new & old nodes agree on outputs, and
which inputs get spent
just don't agree on how they got spent
also don't agree on..?
(dis)agreement
new & old nodes agree on outputs, and which inputs get spent
just don't agree on how they got spent
also don't agree on..?

hint: biggest argument, from 2010...
(dis)agreement
new & old nodes agree on outputs, and which inputs get spent
just don't agree on how they got spent
also don't agree on..?
transaction size! (in bytes)
size (dis)agreement
old nodes don't see witness field;
the 100+ bytes of pubkey / signature
aren't there
those bytes don't count towards the
1M block size limit
-> block size increase soft fork
witness discount
to prevent spamming new nodes, witness bytes still count: $\frac{1}{4}$ a regular byte
(in new software, multiply non-witness bytes by 4 and count max block size as 4M)
end result: ~80% more txs / block
commit signatures
if signatures aren't in txid, they aren't in the merkle root
agree on utxo set, disagree on signature data
weird! disagreement on who signed multisig; bad for accountability
commit signatures
commit to all txids
commit signatures
make witness
hash merkle
tree; commit to
witness root in
coinbase tx
upgrade path

output script:

0 <pubkey hash>

now means pay to pubkey hash

1...16 <data>

means... no witness needed (yet!)
upgrade path
16 more versions to upgrade to
currently don't need anything, but new nodes can require new scripts, smart contracts, etc
nicer upgrade with less ugly code
don't send to 2 <pubkey> today!
segwit fixes malleability, increases block size, does other stuff
some people don't like it
unclear why