The Case for Bitcoin Cash

Smart contracts












bitcoin



#	Coin		Price	1h	24h	7d	24h Volume	Circulating Supply	Mkt Cap	Last 7 Days
☆ 1	Bitcoin	BTC	£6,784.95	0.1%	-3.9%	-4.4%	£19,866,959,638	18,042,200	£122,415,424,890	
☆ 2	Ethereum	ETH	£143.66	-0.5%	-2.8%	2.3%	£6,545,182,714	108,516,936	£15,588,103,235	
☆ 3	XRP	XRP	£0.213498	-0.2%	-2.4%	-4.8%	£1,354,112,741	43,298,481,757	£9,247,298,190	
☆ 4	Bitcoin Cash	BCH	£222.61	-0.2%	-2.9%	-1.2%	£2,084,061,995	18,107,889	£4,031,326,120	
☆ 5	Tether	USDT	£0.777539	0.1%	-0.5%	0.6%	£21,145,222,182	4,288,917,905	£3,336,313,825	
☆ 6	Litecoin	LTC	£47.98	-1.0%	-3.4%	6.4%	£1,966,174,050	63,659,308	£3,055,157,822	
☆ 7	EOS	EOS	£2.67	-0.3%	-4.5%	5.8%	£1,603,059,349	1,031,823,339	£2,760,778,318	
☆ 8	Binance Coin	BNB	£15.63	-0.1%	-2.2%	0.6%	£159,913,933	153,474,825	£2,400,210,021	
☆ 9	Bitcoin SV	BSV	£102.13	-0.1%	-2.8%	2.2%	£477,578,522	18,105,002	£1,846,722,655	
☆ 10	Stellar	XLM	£0.061927	1.0%	0.6%	16.0%	£162,870,990	20,054,779,553	£1,240,726,317	

#	Coin		Price	1h	24h	7d	24h Volume	Circulating Supply	Mkt Cap	Last 7 Days
☆	1	**Bitcoin**	BTC	£6,784.95	0.1%	-3.9%	-4.4%	£19,866,959,638	18,042,200	£122,415,424,890	
☆	2	**Ethereum**	ETH	£143.66	-0.5%	-2.8%	2.3%	£6,545,182,714	108,516,936	£15,588,103,235	
☆	4	**Bitcoin Cash**	BCH	£222.61	-0.2%	-2.9%	-1.2%	£2,084,061,995	18,107,889	£4,031,326,120	
☆	6	**Litecoin**	LTC	£47.98	-1.0%	-3.4%	6.4%	£1,966,174,050	63,659,308	£3,055,157,822	
☆	7	**EOS**	EOS	£2.67	-0.3%	-4.5%	5.8%	£1,603,059,349	1,031,823,339	£2,760,778,318	
☆	9	**Bitcoin SV**	BSV	£102.13	-0.1%	-2.8%	2.2%	£477,578,522	18,105,002	£1,846,722,655	

#	Coin		Price	1h	24h	7d	24h Volume	Circulating Supply	Mkt Cap	Last 7 Days
☆	1	**Bitcoin**	BTC	£6,784.95	0.1%	-3.9%	-4.4%	£19,866,959,638	18,042,200	£122,415,424,890	
☆	2	**Ethereum**	ETH	£143.66	-0.5%	-2.8%	2.3%	£6,545,182,714	108,516,936	£15,588,103,235	
☆	4	**Bitcoin Cash**	BCH	£222.61	-0.2%	-2.9%	-1.2%	£2,084,061,995	18,107,889	£4,031,326,120	
☆	7	**EOS**	EOS	£2.67	-0.3%	-4.5%	5.8%	£1,603,059,349	1,031,823,339	£2,760,778,318	


ethereum


BitcoinCash


bitcoin



- 🖥 Turing-complete
- 💾 Persistent state
- 🎡 Many participants



- ❌ Not Turing-complete
- 💨 Stateless
- 👥 Few participants



ethereum



💾 Effects dependent on state

➡️ Transaction order important

〰️ Parallelisation more difficult

💨 Effects independent of state

🔀 Transaction order not important

📏 Parallelisation easier

The Case for



May 2018

Word	OpCode	Hex	Input	Output	Description
OP_CAT	126	0x7e	x1 x2	out	Concatenates two byte sequences
OP_SPLIT	127	0x7f	x n	x1 x2	Split byte sequence *x* at position *n*
OP_AND	132	0x84	x1 x2	out	Boolean *AND* between each bit of the inputs
OP_OR	133	0x85	x1 x2	out	Boolean *OR* between each bit of the inputs
OP_XOR	134	0x86	x1 x2	out	Boolean *EXCLUSIVE OR* between each bit of the inputs
OP_DIV	150	0x96	a b	out	*a* is divided by *b*
OP_MOD	151	0x97	a b	out	return the remainder after *a* is divided by *b*
OP_NUM2BIN	128	0x80	a b	out	convert numeric value *a* into byte sequence of length *b*
OP_BIN2NUM	129	0x81	x	out	convert byte sequence *x* into a numeric value

November 2018

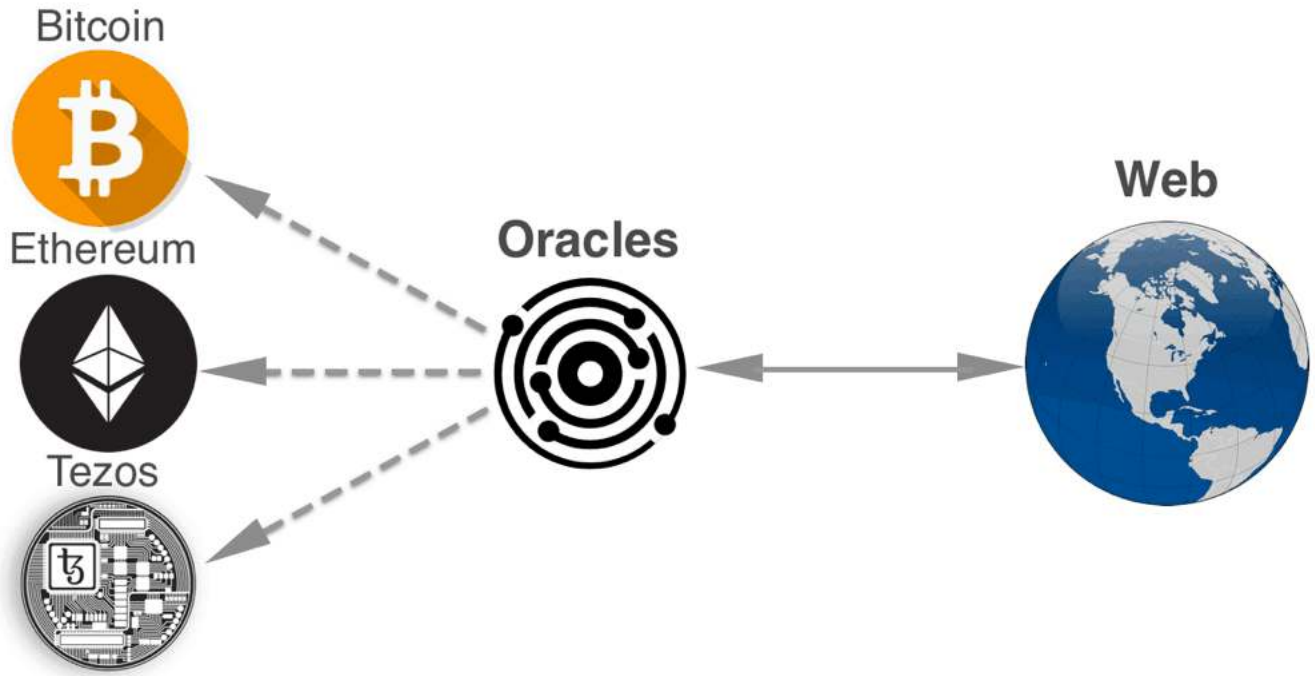
OP_CHECKDATASIG

OP_CHECKDATASIG and OP_CHECKDATASIGVERIFY check whether a signature is valid with respect to a message and a public key.

OP_CHECKDATASIG permits data to be imported into a script, and have its validity checked against some signing authority such as an "Oracle".

OP_CHECKDATASIG and OP_CHECKDATASIGVERIFY are designed to be implemented similarly to OP_CHECKSIG [1]. Conceptually, one could imagine OP_CHECKSIG functionality being replaced by OP_CHECKDATASIG, along with a separate Op Code to create a hash from the transaction based on the SigHash algorithm.

Oracles



Oracles

```
pragma cashscript ^0.2.0;

contract HodlVault(pubkey ownerPk, pubkey oraclePk, int minBlock, int priceTarget) {
    function spend(sig ownerSig, datasig oracleSig, bytes oracleMessage) {
        // message: { blockheight, price }
        int blockHeight = int(oracleMessage.split(4)[0]);
        int price = int(oracleMessage.split(4)[1]);

        // Check that blockHeight is after minBlock and not in the future
        require(blockHeight >= minBlock);
        require(tx.time >= blockHeight);

        // Check that current price is at least priceTarget
        require(price >= priceTarget);

        // Handle necessary signature checks
        require(checkDataSig(oracleSig, oracleMessage, oraclePk));
        require(checkSig(ownerSig, ownerPk));
    }
}
```

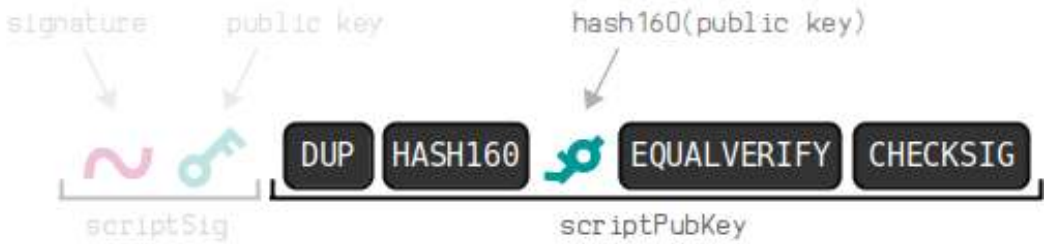
Covenants

```
pragma cashscript ^0.2.0;

contract Covenant(bytes20 pkh) {
    function spend(pubkey pk, sig s, bytes preimage) {
        require(hash160(pk) == pkh);
        require(checkSig(s, pk);
        require(checkDataSig(datasig(sig), sha256(preimage), pk);
    }
}
```

Covenants

1. nVersion of the transaction (4-byte little endian)
2. hashPrevouts (32-byte hash)
3. hashSequence (32-byte hash)
4. outpoint (32-byte hash + 4-byte little endian)
5. scriptCode of the input (serialized as scripts inside CTxOuts)
6. value of the output spent by this input (8-byte little endian)
7. nSequence of the input (4-byte little endian)
8. hashOutputs (32-byte hash)
9. nLocktime of the transaction (4-byte little endian)
10. sighash type of the signature (4-byte little endian)



6 PICK 4 SPLIT SWAP BIN2NUM BIN2NUM SWAP DUP 5 ROLL GREATERTHANOREQUAL VERIFY CHECKLOCKTIMEVERIFY DROP 3 ROLL GREATERTHANOREQUAL VERIFY 3 ROLL 4 ROLL 3 ROLL CHECKDATASIGVERIFY CHECKSIG

```rust
fn script(&self) -> Script {
    use crate::script::OpCodeType::*;
    Script::new(vec![
        Op::Code(Op2Dup),
        Op::Code(OpDup),
        Op::Code(OpHash160),
        Op::Push(self.address.bytes().to_vec()),
        Op::Code(OpEqualVerify),
        Op::Code(OpSwap),
        Op::Push(vec![0x41]),
        Op::Code(OpCat),
        Op::Code(OpSwap),
        Op::Code(OpCheckSigVerify),
        Op::Code(OpRot),
        Op::Code(OpSha256),
        Op::Code(OpSwap),
        Op::Code(OpCheckDataSig),
    ])
}
```

CashScript

```
pragma cashscript ^0.2.0;

contract TransferWithTimeout(
    pubkey sender,
    pubkey recipient,
    int timeout
) {
    // Require recipient's signature to match
    function transfer(sig recipientSig) {
        require(checkSig(recipientSig, recipient));
    }

    // Require timeout time to be reached and sender's signature to match
    function timeout(sig senderSig) {
        require(checkSig(senderSig, sender));
        require(tx.time >= timeout);
    }
}
```

CashScript

 Accessibility without Bitcoin Script knowledge
 Integration within JavaScript applications
 Familiarity for Ethereum developers

CashScript

- 🧷 Fewer errors
- ⚡ Faster development
- 🔰 Easier integration
- 🤔 Covenants are hard

Bitcoin Script

- ⚖️ Optimised code
- 🔧 More low-level control
- ✨ Complete functionality
- 🤔 Covenants are hard

The future of smart contracts

 Compiler optimisation
 Covenants abstraction
 Bigger smart contracts