Some Important BIPs:
BIP 32, BIP 39, BIP 44, BIP 45, BIP 70-74, BIP 65

Ryan X. Charles
Blockchain University
Tokyo, Dec. 19, 2015

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

Outline

BIP 16: P2SH

BIP 32: HD Keys

BIP 39: Mnemonics

BIP 44: HD Wallets

BIP 45: Multisig Wallets

BIP 70 – 74: Payment Protocol

BIP 65: OP_CHECKLOCKTIMEVERIFY

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

Overview

- BIP = Bitcoin Improvement Proposal
- “Official” BIPs located on GitHub here:
  - https://github.com/bitcoin/bips
- People invent standard, discuss on mailing list and IRC, ultimately culminates in a BIP
- Many BIPs are important bitcoin standards
- Some are obsolete and were never widely used (BIP 10, BIP 12)

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

Overview

- BIP 16: P2SH: Makes it easier to use advanced scripts
- BIP 32, 39, 44, 45: Standard ways to use deterministic keys and deterministic wallets
- BIP 70 – 74: Payment protocol – makes it easier to send and receive payments
- BIP 65: CHECKLOCKTIMEVERIFY – makes it possible to lock funds (in a practical way)

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

BIP 16: P2SH

- Normally, can spend money to an address using normal output:
  - OP_DUP OP_HASH160 <address> OP_EQUALVERIFY OP_CHECKSIG
- ...Since everything is always the same, except the address, this template lets a wallet spend directly to an address, which short and easy to copy+paste

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

BIP 16: P2SH

- However, paying to a complicated script, where you can't copy+paste an address, is cumbersome, such as multisig:
  - OP_2 <pubkey> <pubkey> <pubkey> OP_3 OP_CHECKMULTISIG
- Problems: Very long, cannot easily copy+paste, easy to make errors copying

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

BIP 16: P2SH

- One solution: Encode entire output in base58check
  - ...would work, but would lead to very long strings
- Another solution: Add "**OP_EVAL**":
  - Output: OP_DUP OP_HASH160 <hash> OP_EQUALVERIFY OP_EVAL
  - Input: <signature1> <signature2> ... <serialized script>
- Up: Can pay to "hash" just like normal address
- Down: Requires hard-fork of bitcoin protocol
- Abandoned BIP 12

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

BIP 16: P2SH

- Better solution: Match output to template – does not require hard fork – soft fork to script interpreter
- Another solution:
  - Output: OP_HASH160 <hash> OP_EQUALVERIFY
  - Input: <signature1> <signature2> ... <serialized script>
- Up: Can pay to “hash” just like normal address
- Down: None!
- Accepted and standardized BIP 16
- **P2SH = Pay To Script Hash**

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

BIP 16: P2SH

- Example P2SH “redeemScript” (same as normal multisig output):
  - OP_2 <pubkey> <pubkey> <pubkey> OP_3 OP_CHECKMULTISIG
  - Or, with some pubkeys filled in,
  - OP_2 33 0x029cf97e1052008852da9d107411b2d47aad387612558fa864b723c484f8931176 33 0x02f23ab919b3a4795c75552b3985982f54c4164a26948b9fe87625705f694e7aa9 OP_2 OP_CHECKMULTISIG
- Corresponding address:
  - 3AhyGwmdvYZHnmBKx5pry5Zxv8VEVq1kGY
  - Hash redeemScript, prepend with 0x05 version byte, then base58check encode

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

BIP 16: P2SH

- How to pay to 3AhyGwmdvYZHnmBKx5pry5Zxv8VEVq1kG
- Base 58 check decode, and gather hash:
  - 62e7565d9fc50d1e5f6001d7f1f0a7467b9e6709
- Now place hash in output template:
  - OP_HASH160 <hash> OP_EQUALVERIFY
  - ...and get:
  - OP_HASH160 62e7565d9fc50d1e5f6001d7f1f0a7467b9e6709 OP_EQUALVERIFY
- Now place that script in your output, paying as much as you want

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

BIP 16: P2SH

- How to spend from 3AhyGwmdvYZHnmBKx5pry5Zxv8VEVq1kG
- Recover known redeemScript:
  - OP_2 33 0x029cf97e1052008852da9d107411b2d47aad387612558fa864b723c484f8931176 33 0x02f23ab919b3a4795c75552b3985982f54c4164a26948b9fe87625705f694e7aa9 OP_2 OP_CHECKMULTISIG
- ...convert to binary version to get serialized script:
  - 5221029cf97e1052008852da9d107411b2d47aad387612558fa864b723c484f89311762102f23ab919b3a4795c75552b3985982f54c4164a26948b9fe87625705f694e7aa952ae
- Now add signatures and serialized script to get input script:
  - OP_0 <signature1> <signature2> <serialized script>
- ...extra OP_0 is due to famous multisig bug
- Now you have the input script ... add other inputs and outputs

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

BIP 16: P2SH

- How to validate spend from 3AhyGwmdvYZHnmBKx5pry5Zxv8VEVq1kG
- Run input script (scriptSig) as normal
- Run output script (scriptPubkey) as normal
- *If output script matches P2SH template*, which it does in this case, then pop last item from the stack (the serialized script), deserialize it, and run it
- i.e., execute: scriptSig → scriptPubkey → redeemScript
- ...if at no point did the script return false and a true value is left on the top of the stack at the end, then this script does not invalidate the transaction

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

BIP 16: P2SH

- **Summary**: P2SH adds “serialized script”, or redeemScript, to end of scriptSig
- If the output script matches the P2SH template, then the redeemScript is popped and executed after the other scripts are executed
- The P2SH output template is:
  - OP_HASH160 <hash> OP_EQUALVERIFY
- P2SH inputs look like this:
  - <push1> <push2> ... <serialized script>

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

BIP 32: HD Keys

- For privacy, you should always use a new bitcoin address for new payments
- This is difficult to manage – must always backup wallet after every new key is generated
- What if we could deterministically generate new keys?
- Enter: Deterministic Keys

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

BIP 32: HD Keys

- Simplest deterministic key scheme:
- Master seed is a random 256 bit buffer, such as:
- 69ee755ffa5a5f7a9692ca0495108c6a7502ef221607b1ead4158badd37314d1
- Then, append a number to the end:
- 1st: 69ee755ffa5a5f7a9692ca0495108c6a7502ef221607b1ead4158badd37314d100000001
- 2nd: 69ee755ffa5a5f7a9692ca0495108c6a7502ef221607b1ead4158badd37314d100000002
- ...
- Then, hash to produce valid private key buffer:
- 1st: 263b61338ce5b8e167fcf6d178a07346e4915397d174378ddca2739809a59527
- 2nd: 5fc4923126816ce3e710f9314914be80fa290781221926b3591d2373b94d0904
- ...

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

BIP 32: HD Keys

- Now can produce privkey/pubkey from privkey buffers:
- Then, append a number to the end:

- Privkey 1: KxW2d8rp989g1nR36mU7R8nWjCvhSmVe7WzFv7Yc2PGEmnnWTYZE
- Pubkey 1: 03cdc1fe2afc3a59ce4d17d908aa3a37b6993a3f72b7183d592d0fd0def483b6e2
- Addr 1: 1M3w58u3pkMs46AnuSFqjV1LUyeCgakjRW

- Privkey 2: KzRsUZW5gJ7hVERuQt8EHDWjrg1aKXJmn5FTfm5PbYTi7TT1qjCJ
- Pubkey 2: 02db9d3ae6254e5b403423e0f2b4ef94176c2fdfe631f44fd7051fd20e1d960cd7
- Addr 2: 18gVXQ1voex5brB33zR6LZnj2ibHSjghnG

- ...
- Only need to backup master seed. All keys can be derived.

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

BIP 32: HD Keys

- Problems with simple approach:
- Would like a standard that everyone agrees to.
- Would like to be able to share public keys and let people derive new public keys to which you have privkey.
- Would like to be able to partition groups of keys into “accounts”
- …all with one master key that needs to be backed up once.
- **BIP 32** solves all of these.

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

BIP 32: HD Keys

- Neat elliptic curve math trick:
- Keypair: P = pG
- Another keypair: A = aG
- Can add: (P + A) = (p + a)G
- Can multiply: (P + 2A) = (p + 2a)G
- Thus, can share public key P and “derivation value” A, and anybody can derive new public keys (P + nA) to which only you have the private key (p + na)
- … this mathematical fact is used by BIP 32 to make it possible for others, or even yourself, to **derive new public keys without knowing the private keys**

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

BIP 32: HD Keys

- **BIP32**:
- 1) Start with some entropy of at least 128 bits
- 2) sha512 hash to get “Master Extended Private Key” - first 256 bits are “private key”, and second 256 bits are “chain code”
- 3) Chain code used for “derivation value” so other people can derive new public keys without knowing the private key
- 4) Is hierarchical – by allowing “derivation of derivation” can find a “path” such as “m/0/3/4/3/5/5000”
- 5) Allows “hardened” keys, where private key is used in derivation so that knowledge of privkey is necessary for those, such as “m/6h/4” or “m/6'/4” - that requires knowledge of “m/6” privkey to derive “/4”

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

BIP 32: HD Keys

- **BIP32**:
- 6) To encode, use special version bits so that every “extended private key” starts with letters “xprv” in base58, and every “extended public key” starts with “xpub” in base58.
- Example Master Entropy: 000102030405060708090a0b0c0d0e0f
- Master extended private key: xprv9s21ZrQH143K3QTDL4LXw2F7HEK3wJUD2nW2nRk4stbPy6cq3jPPqjiChkVvvNKmPGJxWUtg6LnF5kejMRNNU3TGtRBeJgk33yuGBxrMPHi
- Master extended public key: xpub68Gmy5EdvgibQVfPdqkBBCHxA5htiqg55crXYuXoQRKfDBFA1WEjWgP6LHhwBZeNK1VTsfTFUHCdrfp1bgwQ9xv5ski8PX9rL2dZXvgGDnw

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

BIP 32: HD Keys

- **m/0h** xpriv:
  xprv9uHRZZhk6KAJC1avXpDAp4MDc3sQKNxDiPvvkX8Br5ngLNv1TxvUxt4cV1rGL5hj6KCesnDYUhd7oWgT11eZG7XnxHrnYeSvkzY7d2bhkJ7
- **m/0h** xpub:
  xpub68Gmy5EdvgibQVfPdqkBBCHxA5htiqg55crXYuXoQRKfDBFA1WEjWgP6LHhwBZeNK1VTsfTFUHCdrfp1bgwQ9xv5ski8PX9rL2dZXvgGDnw
- **m/0h/1** xpriv:
  xprv9wTYmMFdV23N2TdNG573QoEsfRrWKQgWeibmLntzniatZvR9BmLnvSxqu53Kw1UmYPxLgboyZQaXwTCg8MSY3H2EU4pWcQDnRnrVA1xe8fs
- **m/0h/1** xpub:
  xpub6ASuArnXKPbfEwhqN6e3mwBcDTgzisQN1wXN9BJcM47sSikHjJf3UFHKkNAWbWMiGj7Wf5uMash7SyYq527Hqck2AxYysAA7xmALppuCkwQ

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

BIP 32: HD Keys

- BIP 32 Summary:
- Can backup one Master Private Key
- Can derive in paths for public or private keys: m/5/6/5/5/2
- Can derive **hardened** that require private key for that path: m/5h
- Can derive public key from public key: m/5/6/7 = m/5/6 → m/7
- Can derive private key from private key: m/5/6/7 = m/5/6 → m/7

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

BIP 39: Mnemonics

- BIP 32 is awesome, but it has one problem: It's difficult and error-prone to backup the master extended private key
- We want:
  - To be able to easily **write down** master key
  - To be able to easily **memorize** master key
  - To be reasonably sure that we did **not have errors** when writing it

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

BIP 39: Mnemonics

- BIP 39 is a standard way to convert entropy into a "mnemonic" with a built-in checksum
- Properties:
  - Easy to write down list of words
  - Easy to memorize list of words
  - Included checksum makes errors less likely

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

BIP 39: Mnemonics

- BIP 39:
- 1) Start with random entropy
- 2) Hash the entropy for the checksum
- 3) Convert checksum and seed into words using a standard wordlist (wordlist should be common words with no easily-confused duplicates)
- 4) Hash words to get master seed – to be used by BIP 32 to get master xprv

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

BIP 39: Mnemonics

- BIP 39:
- Entropy: f76c442bf7847df1a6c1a859043eb02e
- Mnemonic: “wash giraffe april upper elephant web only crush flip capable project front”
- Seed:
  c134ba00badd038b9f7bc8506c5a6245c0762e1d2fb65e73606f353298c3014b1c748baa9b9e6d0cedcf6d11fa192cf707d6e85370180d5d95274ba09e72e279
- Master xprv:
  xprv9s21ZrQH143K4GKGLy7cuGD7dqsXC8Sy82FQkeABdzaZ7otVpbrMZyK6CmSjZcRiaYRnNrGa9GxRUqz6mzasQiq3QmdHwNcqgsFbBoNba7G

BIP 39: Mnemonics

- BIP 39 Summary:
- Mnemonic is easier to memorize than a “seed”
- Multiple wordlists standardized: English, Japanese, French, Spanish
- Mnemonic includes checksum, so that you can know you've made an error – cannot simply get one character wrong

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

BIP 44: HD Wallets

- BIP 32 is awesome, but it doesn't specify how to derive the accounts
  - e.g., “m/4/6/1000” ? Or “m/9/9/9/9/4000” ? Or what?
- Want a standard way to derive accounts that every bitcoin wallet can support, so you can port a key from one wallet to another
- BIP 44: A standard bitcoin wallet account structure

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

BIP 44: HD Wallets

- BIP 44: m / purpose' / coin_type' / account' / change / address_index
- purpose' = 44, in our case
- coin_type' = 0 (for bitcoin)
  account' = 0, 1, 2, 3 ... for your accounts
- change = 0 (not change) or 1 (change)
- address_index = 0, 1, 2, 3, 4, ... for your actual addresses

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

BIP 44: HD Wallets

- Properties:
- Given a **master account pubkey**, m/44'/0'/0', anyone can derive new public keys of yours (but they can't if they only have the derived pubkeys)
- Always derive new addresses in order, 0, 1, 2, 3, and you can scan out the index to see if you've been paid – saving the requirement of remembering what index you last used

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

BIP 44: HD Wallets

- Example:
- mnemonic: actual aunt rural miss lumber anger extend inquiry theme creek jar boring
- master xprv: xprv9s21ZrQH143K3x4uq1VayJp8owEFR2bU9iVDrqyuApveA3zGFV5wH6FMWU2viZWJD5mM2aMrjAAndfxXUzBatse2PfAeStNxx6wqm2fL2o1
- account 0, address 0: 1JwbGXAAFoxT5SGPLWjhZoB5nK5h212qQx
- account 0, change address 0: 1EzrZJnLmkx4qCJciBiFmkHHiQbQsXg3Y1
- account 0, address 1: 1AEYnTMBWTHLE1b4M7kTgANjMApw2UvMu4
- account 0, change address 1: 17nuXdATjNq4obqcYueRtaSDn1RVB7YLem
- account 1, address 0: 14r8v1YM3qwobZnnuwmFT2oqwrdfamRMh9
- account 1, change address 0: 19JEJMzfixRABsZVxqNdxSwXuHWgmx3g7
- account 1, address 1: 1FoiJytRCNkTBNJZbgEwhW2t41nYRQCcqY
- account 1, change address 1: 1M3ALRiMrV8ut1kbqyyPyn7NyPn3ty9zYj

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

BIP 32, BIP 39, BIP 44

- Summary:
- Derive master mnemonic using BIP 39
- Derive xprvs and xpubs using BIP 32
- Derive actual addresses for each “account” using BIP 44
- ...standard bitcoin wallet format used by many (but not all) single-sig bitcoin wallet software

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

BIP 45

- BIP 44 has one problem – only designed for single-sig
- Multisig requires knowing other parties' public keys in order to create redeemScripts and p2sh addresses
- Need a standard for how to combine pubkeys into p2sh multisig addresses

BIP 45

- BIP 45: m / purpose' / cosigner_index / change / address_index
- **Cosigner:** a person on an m-of-n wallet
- Each cosigner can compute each other's public keys (thanks BIP32 32)
- To compute address, first compute each party's public key.
- Then arrange then lexicographically (i.e., alphabetically)
- Then compute redeemScript
- Then address

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

BIP 45

- Summary:
- Solves the problem of collaboration in multisig wallet
- Unfortunately, BIP 45 is only used by Copay
- Other multisig wallets do this differently – AFAIK there is no “standard” way to collaborate on multisig wallets (meaning each person cannot choose their own wallet software)
- (fun fact: I coauthored BIP 45)

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

BIP 70 – 73

- Bitcoin addresses are hard to copy+paste for normal people
- They look weird
- Would be preferable to pay directly to a **person** rather than
- BIP 70 (et al.): Your wallet software asks, “Do you wish to pay 1 BTC to Blockchain University?”

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

BIP 70

- BIP 70 stages from user point-of-view:
  - User clicks a payment button on a website, or somehow else alerts their wallet software about the payment
  - Wallet software asks, “Do you wish to pay X bitcoin to Y merchant?”
  - User clicks “yes” or “no”
  - User is alerted that payment was accepted

BIP 70

- BIP 70 stages from technical point-of-view:
  - Merchant alerts user's wallet to existence of payment request and gives the **payment URI**
  - Wallet software retrieves **payment request** from merchant
  - Wallet software **validates signature on payment request**
  - If user agrees to payment, wallet sends **payment** directly to merchant
  - Merchant sends back a **payment ack**

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

BIP 70

- BIP 70 technical details:
  - Data serialized using **Google Protocol Buffers** – a compact data format that can be reproducibly hashed, making it suitable for signing/verifying
  - Data structures:
  - **Payment Request** (Output, Payment Details)
  - **Payment** (contains bitcoin transaction)
  - **Payment ACK**

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

BIP 70

- Output:

```
message Output {
    optional uint64 amount = 1 [default = 0];
    optional bytes script = 2;
}
```

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

BIP 70

- Payment Details:

```
message PaymentDetails {
    optional string network = 1 [default = "main"];
    repeated Output outputs = 2;
    required uint64 time = 3;
    optional uint64 expires = 4;
    optional string memo = 5;
    optional string payment_url = 6;
    optional bytes merchant_data = 7;
}
```

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

BIP 70

- Payment Request:

```
message PaymentRequest {
    optional uint32 payment_details_version = 1 [default = 1];
    optional string pki_type = 2 [default = "none"];
    optional bytes pki_data = 3;
    required bytes serialized_payment_details = 4;
    optional bytes signature = 5;
}
```

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

BIP 70

- Payment:

```
message Payment {
    optional bytes merchant_data = 1;
    repeated bytes transactions = 2;
    repeated Output refund_to = 3;
    optional string memo = 4;
}
```

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

BIP 70

- PaymentACK:

```
message PaymentACK {
    required Payment payment = 1;
    optional string memo = 2;
}
```

BIP 71

- BIP 71: BIP 70 MIME Types for HTTP transmission

Message	Type/Subtype
PaymentRequest	application/bitcoin-paymentrequest
Payment	application/bitcoin-payment
PaymentACK	application/bitcoin-paymentack

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

BIP 72

- BIP 72: Bitcoin URI Extensions
- bitcoin:mq7se9wy2egettFxPbmn99cK8v5AFq55Lx?amount=0.11&r=https://merchant.com/pay.php?h%3D2a8628fc2fbe
- ← includes not just address and amount, but also merchant URI for retrieving payment request
- ← FYI, this is the thing that is encoded in a bitcoin QR code

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

BIP 73

- BIP 73: Accept Headers
- Wallet can send an “accept” header to a server to indicate it is capable of understanding BIP 70 payment requests
- Allow encoding a normal http URI into a QR code, rather than a full bitcoin URI – the user gets a normal webpage if they don't support BIP 70, or their wallet software gets a payment request if they do support BIP 70

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

BIP 70 – 73 Summary

- User's POV: “Do you want to pay X BTC to Company Y?”
- Tech POV: **Payment Request → Payment → Payment ACK**
- Much more user-friendly than copy+pasting bitcoin addresses
- Signed – can know you're paying the right merchant
- Can use “proof of payment” in disputes
- Comforting to know merchant has received payment
- Refund addresses in case something goes wrong

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

BIP 65: CHECKLOCKTIMEVERIFY

- Bitcoin has 10 “NO OP” codes for “future expansion”: NOP1, NOP2, … NOP10
- nlocktime is barely useful … specifying it means your transaction is not valid until some future date … but can be easily double spent
- CHECKLOCKTIMEVERIFY: re-use NOP2 to check nlocktime, and cannot spend unless time has passed
- Makes nlocktime much more useful

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

BIP 65: CHECKLOCKTIMEVERIFY

- nlocktime: Transaction not valid until block height or date
- CHECKLOCKTIMEVERIFY: Checks top item on stack and compares against nlocktime of tx – tx is invalid unless nlocktime has passed
- In other words, CHECKLOCKTIMEVERIFY lets you lock funds **on the blockchain** until a certain block height or time has passed.

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt

Summary

- BIP 32, BIP 39, BIP 44: How to derive keys for a wallet
- BIP 45: Good way to do make a multisig wallet
- BIP 70 – 73: Make it easier to pay for stuff
- BIP 65: Make nlocktime more useful

Ryan X. Charles
Founder of Datt (datt.co)
twitter.com/ryanxcharles
github.com/ryanxcharles

Code Samples and Slides at:
github.com/ryanxcharles/blockchain-university

Datt