* 1. Table of Contents

1. Module 1: Introduction to Bitcoin SV
2. Lesson 1.1: History and Evolution of Bitcoin SV
3. Lesson 1.2: Core Principles of Bitcoin SV
4. Module 2: Getting Started with Bitcoin SV
5. Lesson 2.1: Setting Up a Bitcoin SV Wallet
6. Lesson 2.2: Sending and Receiving Bitcoin SV Transactions
7. Module 3: Exploring Bitcoin SV Blockchain
8. Lesson 3.1: Understanding the Blockchain Structure
9. Lesson 3.2: Smart Contracts and Scripting on Bitcoin SV
10. Module 4: Security and Best Practices for Bitcoin SV
11. Lesson 4.1: Securing Your Wallet and Private Keys
12. Lesson 4.2: Staying Informed and Protecting Your Investment
13. Capstone Project Idea: Creating a Simple Smart Contract
14. Assessment Types:
15. Lesson 1.1: History and Evolution of Bitcoin SV
16. Overview
17. Sub-Lesson 1: Origins and Milestones
18. Sub-Lesson 2: Key Figures and Events
19. Sub-Lesson 3: Unique Selling Points and Role within Crypto Ecosystem
20. Real-World Example
21. Sub-Lesson 4: Discussion Questions
22. Hands-On Exercises
23. Lesson 1.2: Core Principles of Bitcoin SV
24. Overview
25. 1. Scalability and Microtransaction Vision
26. 2. Miner's Incentive and Satoshi's White Paper
27. 3. Enterprise Adoption and BSV Support
28. 4. Smart Contracts and Data on the BSV Blockchain
29. 5. Witness-enabled Transactions
30. Discussion Questions
31. Hands-on Exercises
32. Lesson 2.1: Setting Up a Bitcoin SV Wallet
33. 2.1.1 Understanding Bitcoin SV Wallets
34. 2.1.2 Setting Up a BSV Wallet
35. 2.1.3 Sending and Receiving BSV Transactions
36. 2.1.4 Securing Your BSV Wallet
37. 2.1.5 Discussion Questions:
38. 2.1.6 Hands-on Exercises:
39. Lesson 2.2: Sending and Receiving Bitcoin SV Transactions
40. Overview
41. Sending Bitcoin SV Transactions
42. Receiving Bitcoin SV Transactions
43. Verifying and Decoding Transactions
44. Discussion Questions
45. Hands-On Exercises
46. Learning Outcomes
47. Lesson Overview
48. Understanding Blockchain Components
49. The Block
50. The Blockchain
51. Real-world BSV Example
52. Discussion Questions
53. Hands-On Exercises
54. Overview
55. Smart Contracts
56. BSV's Approach to Smart Contracts
57. Discussion Questions
58. Hands-On Exercises
59. Lesson 4.1: Securing Your Wallet and Private Keys
60. Overview
61. Concept Explanation
62. Discussion Questions
63. Hands-on Exercises
64. Lesson 4.2: Staying Informed and Protecting Your Investment
65. Learning Outcomes
66. Monitoring BSV Market Trends
67. Securing Your BSV with Hardware Wallets
68. Multi-Signature Wallets for Enhanced Security
69. Blockchain Analytics Tools for Securing Investments
70. Discussion Questions
71. Hands-On Exercises
72. Overview
73. Proof-of-Work (PoW)
74. Explanation
75. Real-world Example
76. Proof-of-Burn (PoB)
77. Explanation
78. Real-world Example
79. Discussion Questions
80. Hands-on Exercises
81. Multiple-Choice Questions (3)
82. Short-Answer / Definition Questions (2)
83. Practical Coding Exercise
84. Starter Code
85. Exercise
86. Introduction to Bitcoin SV Block Chain

Duration: 30 minutes  
Difficulty: Beginner  
Target Audience: Newcomers to BSV

# Bitcoin SV Fundamentals on the Bitcoin SV Blockchain  
  
## Module 1: Introduction to Bitcoin SV  
\*Understanding the unique features and benefits of Bitcoin SV\*  
  
### Lesson 1.1: History and Evolution of Bitcoin SV  
- Overview of Bitcoin's evolution from its inception to the creation of Bitcoin SV  
- Discussing the hard forks that led to the formation of Bitcoin SV  
- Key players and their roles in the Bitcoin SV ecosystem  
  
### Lesson 1.2: Core Principles of Bitcoin SV  
- Explanation of Satoshi Vision, the guiding philosophy behind Bitcoin SV  
- Discussion on scalability, stability, and the potential for enterprise adoption  
- Differences between Bitcoin SV and other cryptocurrencies in terms of design and functionality  
  
## Module 2: Getting Started with Bitcoin SV  
\*Learning the essential tools and processes for working with Bitcoin SV\*  
  
### Lesson 2.1: Setting Up a Bitcoin SV Wallet  
- Recommending wallet options suitable for beginners  
- Step-by-step guide on setting up and securing a wallet  
- Understanding private keys, addresses, and transactions  
  
### Lesson 2.2: Sending and Receiving Bitcoin SV Transactions  
- Walkthrough of sending and receiving Bitcoin SV using a chosen wallet  
- Explanation of transaction fees, confirmations, and blockchain explorers  
- Discussing the simplicity and speed of Bitcoin SV transactions compared to other cryptocurrencies  
  
## Module 3: Exploring Bitcoin SV Blockchain  
\*Exploring the intricacies of the Bitcoin SV blockchain\*  
  
### Lesson 3.1: Understanding the Blockchain Structure  
- Explanation of the block structure and how it functions in the Bitcoin SV network  
- Discussing block size, blocks per hour, and the impact on scalability  
- Demonstrating how to access and analyze blockchain data using a block explorer  
  
### Lesson 3.2: Smart Contracts and Scripting on Bitcoin SV  
- Introduction to simple smart contracts and their implementation on Bitcoin SV  
- Overview of OP\_Codes, scripts, and their role in ensuring security and versatility  
- Walkthrough of creating and deploying a basic smart contract using a development environment  
  
## Module 4: Security and Best Practices for Bitcoin SV  
\*Ensuring the safe handling and storage of Bitcoin SV\*  
  
### Lesson 4.1: Securing Your Wallet and Private Keys  
- Discussing best practices for securing your wallet and private keys  
- Explanation of hardware wallets, multi-signature wallets, and paper wallets  
- Tips for protecting yourself from common security threats in the cryptocurrency world  
  
### Lesson 4.2: Staying Informed and Protecting Your Investment  
- Importance of staying updated on Bitcoin SV news and developments  
- Discussion on risk management strategies for investing in cryptocurrencies  
- Introduction to reliable resources for learning more about Bitcoin SV and the broader blockchain ecosystem  
  
## Capstone Project Idea: Creating a Simple Smart Contract  
\*Applying the knowledge gained throughout the course to create a simple smart contract on the Bitcoin SV blockchain.\*  
  
### Assessment Types:  
1. Practical exercises and quizzes to test understanding of key concepts  
2. Peer reviews and discussions to reinforce collaboration and critical thinking  
3. Final project presentation and code review to assess mastery of smart contract development on Bitcoin SV

# Module: Bitcoin SV  
## Lesson 1.1: History and Evolution of Bitcoin SV  
  
### Overview  
In this lesson, we will delve into the history and evolution of Bitcoin SV (BSV), understanding its roots, significant milestones, and current status as a leading cryptocurrency. Upon completion, you should be able to:  
  
1. Understand the origin and evolution of Bitcoin SV.  
2. Recognize key figures and events in BSV's history.  
3. Identify BSV's unique selling points and its role within the crypto ecosystem.  
4. Analyze real-world examples of BSV transactions.  
5. Apply Python or JavaScript to demonstrate BSV transactions.  
  
### Sub-Lesson 1: Origins and Milestones  
Bitcoin SV was born from a contentious hard fork in November 2018, aimed at restoring the original vision for Bitcoin as described in Satoshi Nakamoto's whitepaper.  
  
```python  
# Example Python code demonstrating BSV genesis block transaction  
import hashlib  
block\_header = {  
 'version': 0x20000000,  
 'previous\_block': '000000000019d668', # Genesis Block previous hash  
 'merkle\_root': '4a5e17f7',  
 'time': 1231006505,  
 'bits': '1a010500207f',  
 'nonce': 391872  
}  
genesis\_hash = hashlib.sha256(json.dumps(block\_header, sort\_keys=True).encode()).hexdigest()  
print("Genesis Block Hash:", genesis\_hash)

* + 1. Sub-Lesson 2: Key Figures and Events

1. Craig Wright claimed to be Satoshi Nakamoto in 2016.
2. The Bitcoin Cash hard fork occurred in August 2017, giving birth to Bitcoin ABC and Bitcoin SV.
3. Nchain, a blockchain research and development firm, plays a significant role in BSV's development.
   * 1. Sub-Lesson 3: Unique Selling Points and Role within Crypto Ecosystem
4. Scalability: BSV aims for mass adoption by handling high transaction volumes with low fees.
5. On-Chain scaling enables smart contracts and micropayments.
   * 1. Real-World Example

BSV's scalability allows for the creation of projects like HandCash, a mobile wallet enabling peer-to-peer payments and tipping content creators.

* + 1. Sub-Lesson 4: Discussion Questions
  1. How has the history and evolution of BSV influenced its current status?
  2. What role does scaling play in the success of BSV as a cryptocurrency?
  3. Can you predict the future implications of on-chain smart contracts for BSV?
     1. Hands-On Exercises
  4. Write Python code to send and receive a BSV transaction using a compatible wallet API (e.g., Electrum SV or simpleledger).
  5. Research and write a brief report on a real-world project built on the BSV blockchain, detailing its purpose and benefits.

```markdown  
# Module: Bitcoin SV (BSV)  
  
## Lesson 1.2: Core Principles of Bitcoin SV  
  
### Overview  
This lesson delves into the fundamental principles that define Bitcoin SV (BSV). Upon completion, you will be able to:  
  
1. Explain the Scalability and Microtransaction Vision of BSV  
2. Discuss the role of the Miner's Incentive and Satoshi's White Paper in BSV  
3. Understand how BSV supports Enterprise Adoption  
4. Recognize the importance of Smart Contracts and Data on the BSV blockchain  
5. Explore the benefits of Witness-enabled transactions for BSV  
  
### 1. Scalability and Microtransaction Vision  
  
Bitcoin SV was created to restore the original vision of Bitcoin as a scalable, peer-to-peer electronic cash system. This involves:  
  
> "...[S]caling Bitcoin to its original Satoshi's whitepaper specification, with no artificial block size limits, [and making it] the global public accounting blockchain for enterprise." - Craig S. Wright, nChain CEO  
  
\*\*Real-world BSV example:\*\* The BSV-powered peer-to-peer marketplace OpenBazaar allows users to buy and sell goods without intermediaries. It can handle thousands of transactions per second, making it a scalable solution for microtransactions.  
  
```python  
# Python code example demonstrating creating a new transaction (note: this is not actual BSV code)  
from bitcoin\_sv import CBitcoinClient, CBTC  
  
client = CBitcoinClient()  
tx = client.create\_new\_transaction(0.01, "receiver\_address", "change\_address")  
signed\_tx = tx.sign(CBTC("private\_key"))  
broadcasted\_tx\_id = client.sendrawtransaction(signed\_tx.hex())  
print("Transaction ID:", broadcasted\_tx\_id)

* + 1. 2. Miner's Incentive and Satoshi's White Paper

The Bitcoin SV network follows the original whitepaper's principles, prioritizing the Miner's Incentive to secure the network. This incentivizes miners to include transactions with higher fees in their blocks.

Real-world BSV example: With high transaction fees, miners are incentivized to process more transactions quickly and efficiently, ensuring a stable and reliable network.

// JavaScript code example demonstrating creating a new transaction (note: this is not actual BSV code)  
const BitcoinSV = require("bitcoinsv-node");  
const client = new BitcoinSV();  
const tx = client.createNewTransaction({  
 changeAddress: "change\_address",  
 outputs: [{  
 address: "receiver\_address",  
 value: 0.01,  
 scriptPubKey: BitcoinSV.Scripts.payToAddress(client.getDefaultNetwork())  
 }],  
 feeRate: 24 // 24 satoshis per virtual byte (vByte)  
});  
const txHex = client.serializeTransaction(tx);  
const privateKey = BitcoinSV.Wallet.ECPrivKey.fromBase58("private\_key");  
const signedTx = client.signTransaction([privateKey], txHex);  
const broadcastedTxId = client.sendRawTransaction(signedTx.toString('hex'));  
console.log("Transaction ID:", broadcastedTxId);

* + 1. 3. Enterprise Adoption and BSV Support

BSV's focus on scalability, low fees, and smart contracts makes it an attractive choice for enterprise adoption. Many companies are already using BSV for various applications, including supply chain management and digital identity solutions.

Real-world BSV example: The BSV-powered digital identity platform Onesmart allows users to securely store, verify, and share their personal data with businesses.

* + 1. 4. Smart Contracts and Data on the BSV Blockchain

BSV supports smart contracts through its scripting language, enabling more complex transactions and business logic on the blockchain. Additionally, BSV stores vast amounts of data on its blockchain, making it a powerful platform for decentralized applications (DApps).

* + 1. 5. Witness-enabled Transactions

Witnesses allow larger transaction data to be included in blocks without increasing the size limit. This feature is crucial for smart contracts and enterprise solutions that require more data on the blockchain.

* + 1. Discussion Questions
  1. What advantages does Bitcoin SV have over other cryptocurrencies in terms of scalability and microtransactions?
  2. How does prioritizing the Miner's Incentive impact the security and reliability of the BSV network?
  3. Why is enterprise adoption important for the future growth of BSV, and how does it differ from consumer adoption?
     1. Hands-on Exercises
  4. Write Python or JavaScript code to create and broadcast a simple BSV transaction with a custom output script.
  5. Research and present on an existing BSV DApp that demonstrates the benefits of smart contracts and data storage on the BSV blockchain.

```markdown  
# Module: Bitcoin SV  
## Lesson 2.1: Setting Up a Bitcoin SV Wallet  
  
This lesson aims to equip you with the essential skills required to set up a Bitcoin SV (BSV) wallet. By the end of this lesson, you should be able to:  
  
1. Understand the concept and importance of a BSV wallet.  
2. Set up a BSV wallet using an appropriate wallet application.  
3. Send and receive BSV transactions through the wallet.  
4. Secure your BSV wallet for safekeeping.  
  
### 2.1.1 Understanding Bitcoin SV Wallets  
  
A Bitcoin SV (BSV) wallet is a digital storage that holds private keys to Bitcoin transactions. It allows you to send, receive, and store BSV securely.  
  
#### Real-world Example:  
Sending BSV for online purchases or donations requires a wallet. For instance, a nonprofit organization accepts BSV donations, and you'd need a wallet to transfer your funds to them.  
  
### 2.1.2 Setting Up a BSV Wallet  
  
There are several BSV wallet options available, such as Electron Cash and the Bitcoin SV Wallet app. This lesson demonstrates using Electron Cash for Windows.  
  
#### Installing Electron Cash:  
  
- Download Electron Cash from https://electroncash.org/download  
- Follow the installation instructions provided for your operating system  
  
#### Creating a New BSV Wallet:  
  
- Open Electron Cash, select "Create a new wallet" and follow the onscreen instructions to create a new wallet with a secure password.  
  
### 2.1.3 Sending and Receiving BSV Transactions  
  
Once your wallet is set up, you can send and receive BSV:  
  
#### Sending BSV:  
  
- Select "Send Coins" from the Electron Cash menu  
- Enter the recipient's address, amount, and click "Send"  
  
#### Receiving BSV:  
  
- Share your wallet address with others so they can send you BSV  
  
### 2.1.4 Securing Your BSV Wallet  
  
To secure your BSV wallet, ensure to store your seed words in a safe location, and never share your private keys or password.  
  
#### Real-world Example:  
Storing your seed words offline using a hardware wallet provides an extra layer of security against hackers.  
  
### 2.1.5 Discussion Questions:  
  
1. Why is it important to secure your BSV wallet?  
2. What measures can you take to ensure the safety of your wallet and its contents?  
3. How does a hardware wallet improve the security of your BSV wallet?  
  
### 2.1.6 Hands-on Exercises:  
  
1. Send a small amount of BSV to another address (either yours or a friend's) using Electron Cash.  
2. Research and set up a hardware wallet for additional security.

# Module 2: Bitcoin SV Transactions  
  
## Lesson 2.2: Sending and Receiving Bitcoin SV Transactions  
  
### Overview  
In this lesson, you'll learn how to send and receive Bitcoin SV (BSV) transactions using Python or JavaScript. By the end of this lesson, you'll be able to:  
  
1. Send BSV transactions using a wallet API.  
2. Receive BSV transactions to a new address.  
3. Verify and decode BSV transactions.  
  
### Sending Bitcoin SV Transactions  
  
To send BSV transactions, we will use the [Bitcoin SV Web SDK](https://bsv-web-sdk.readthedocs.io/). Here's a simple example of sending a transaction using Python:  
  
```python  
from bsvpy import \*  
  
# Initialize Bitcoin SV instance with your node URL and credentials  
bitcoinsv = BSV(url='http://your-node-url', username='your-username', password='your-password')  
  
# Define input and output addresses for the transaction  
input\_address = 'bsv1q50gefk8yljts Bundled Address'  
output\_address = 'bsv1q3w942h5u4c90 Bundled Address'  
amount = 5000  
  
# Get UTXOs for the input address  
utxos = bitcoinsv.get\_utxo\_for\_address(input\_address)  
  
# Find a suitable UTXO to cover the output amount  
utxo = sorted(utxos, key=lambda utxo: utxo['amount'], reverse=True)[0]  
  
# Construct the transaction with the input and output scripts  
tx = TxBuilder()  
tx.add\_input(utxo)  
tx.add\_output(output\_address, amount)  
tx.sign(private\_key=PrivateKey(key='your-private-key'))  
  
# Send the transaction to the Bitcoin SV network  
txid = bitcoinsv.send\_raw\_transaction(str(tx))  
print("Transaction sent with ID:", txid)

Replace the placeholders with your node URL, credentials, input address, output address, and private key for signing.

* + 1. Receiving Bitcoin SV Transactions

To receive BSV transactions, you can generate a new address using Python:

from bsvpy import \*  
  
# Initialize Bitcoin SV instance with your node URL and credentials  
bitcoinsv = BSV(url='http://your-node-url', username='your-username', password='your-password')  
  
# Generate a new address for receiving BSV  
address = bitcoinsv.new\_address()  
print("New receiving address:", address)

* + 1. Verifying and Decoding Transactions

To verify and decode transactions, we can use the bsvpython library in Python:

from bsvpy import \*  
  
# Initialize Bitcoin SV instance with your node URL and credentials  
bitcoinsv = BSV(url='http://your-node-url', username='your-username', password='your-password')  
  
# Get the raw transaction hex data  
raw\_transaction = '01000000...0000'  
  
# Decode the transaction into objects  
tx\_decoded = bitcoinsv.decode\_tx(raw\_transaction)  
  
# Verify the transaction on the Bitcoin SV network  
txid = bitcoinsv.get\_tx\_hash(tx\_decoded)  
block\_height = bitcoinsv.get\_block\_count() - bitcoinsv.get\_mempool\_size() - 1  
result = bitcoinsv.verify\_tx(tx\_decoded, block\_height)  
print("Transaction verified:", result)

* + 1. Discussion Questions
  1. How does the Bitcoin SV network ensure that double-spending is avoided when multiple transactions are competing for the same inputs?
  2. What are some potential challenges when sending and receiving BSV transactions, and how can they be addressed?
  3. What role do blockchain nodes play in validating and broadcasting transactions to the Bitcoin SV network?
     1. Hands-On Exercises
  4. Write a Python script to send a transaction using the Bitcoin SV Web SDK with a different output amount and address.
  5. Implement a simple JavaScript script that interacts with the Bitcoin SV blockchain, e.g., generate a new address, receive BSV, or verify transactions.

```markdown  
# Lesson 3.1: Understanding the Blockchain Structure (BSV)  
  
This lesson provides an in-depth exploration of the blockchain structure within the Bitcoin SV (BSV) network.  
  
## Learning Outcomes  
  
Upon completion of this lesson, you will be able to:  
  
1. Describe the fundamental components of a BSV blockchain.  
2. Identify the role and significance of each component in maintaining the integrity of the network.  
3. Explain how transactions are processed and verified on the BSV blockchain.  
4. Understand the relationship between blocks and the blockchain.  
5. Recognize the importance of block size for scalability in BSV.  
  
## Lesson Overview  
  
This lesson delves into the intricate structure of the BSV blockchain, focusing on its components, transaction processing, and scalability. By understanding these concepts, you will gain a comprehensive grasp of how the BSV network functions.  
  
## Understanding Blockchain Components  
  
### The Block  
  
A block is a collection of transactions that have been verified and added to the blockchain. It contains a header and a body.  
  
```python  
class Block:  
 def \_\_init\_\_(self, index, previous\_hash, timestamp, data\_transactions):  
 self.index = index  
 self.previous\_hash = previous\_hash  
 self.timestamp = timestamp  
 self.data\_transactions = data\_transactions  
 self.hash = self.calculate\_hash()  
  
 def calculate\_hash(self):  
 # Calculate the hash of the block using a suitable hashing algorithm (e.g., SHA-256)  
 pass

* + 1. The Blockchain

The blockchain is a linear sequence of blocks, each one linked to its predecessor by its unique hash. This creates an unalterable and secure chain of information.

* 1. Real-world BSV Example

Let's consider the following transaction:

Alice sends 50 BSV to Bob on the BSV blockchain.

* 1. Alice creates a new transaction, specifying that she wants to send 50 BSV to Bob.
  2. The network validates Alice's signature and checks her balance to ensure she has sufficient funds.
  3. Once validated, the transaction is added to a block.
  4. The block is broadcasted across the network for verification by other nodes.
  5. Upon successful validation, the block is added to the end of the blockchain, marking the completion of Alice's transaction.
  6. Discussion Questions
  7. What are the advantages and disadvantages of a linear blockchain structure like BSV?
  8. How does the block size impact the scalability of the BSV network?
  9. Can you propose an alternative method for structuring a blockchain, and discuss its potential benefits and drawbacks?
  10. Hands-On Exercises
  11. Implement theBlockclass in Python or JavaScript, including the hash calculation function.
  12. Simulate a simple BSV network with two nodes and conduct a transaction between them. Verify that the transaction is successfully added to the blockchain.

```markdown  
# Lesson 3.2: Smart Contracts and Scripting on Bitcoin SV  
  
This lesson will delve into the world of smart contracts and scripting on Bitcoin SV (BSV). By the end of this lesson, you should be able to:  
  
1. Understand the concept of smart contracts and their role in blockchain technology.  
2. Explore BSV's unique approach to scripting and its capabilities.  
3. Learn how to write and deploy simple scripts on the BSV network.  
4. Analyze real-world examples of smart contracts on BSV.  
5. Discuss the potential implications and challenges of using smart contracts on BSV.  
  
## Overview  
  
In this lesson, we will discuss smart contracts and scripting on Bitcoin SV, a blockchain network that supports complex scripts, providing an environment for building decentralized applications (dApps).  
  
## Smart Contracts  
  
A smart contract is a self-executing contract with the terms of the agreement between buyer and seller being directly written into lines of code. The code and the agreements contained therein exist on the blockchain network.  
  
### BSV's Approach to Smart Contracts  
  
Bitcoin SV supports complex scripts, which means that it can handle more data within its blocks compared to other blockchain networks. This allows for more robust smart contracts to be built on the BSV network.  
  
#### Example: Implementing a Voting System  
  
A simple voting system can be implemented on the BSV network using smart contracts. In this example, voters can cast their votes by sending a transaction with their vote data encoded within the scriptSig of the transaction. The scriptPubKey of the recipient's address would verify and count the votes.  
  
```python  
import base58  
from bitcoin.utils import sigencode\_der, encode\_base58  
from bitcoin.keys import ECPrivateKey, SECP256k1  
from bitcoin.core import CScript, CTxIn, COutPoint, COIN  
from bitcoin.script import Op  
  
# Create private key and public key  
private\_key = ECPrivateKey(bytes.fromhex('your\_private\_key'))  
public\_key = private\_key.pub  
  
# Voting options  
options = ['Option A', 'Option B']  
  
# Encode voting options for use in scriptSig  
encoded\_options = [encode\_base58(option.encode('ascii')) for option in options]  
  
# Function to create a signature for a vote  
def sign\_vote(index, private\_key):  
 message = f'Vote {index}'.encode('utf-8')  
 signature = sigencode\_der(private\_key, message)  
 return signature  
  
# Cast a vote  
index = 0  
signature = sign\_vote(index, private\_key)  
vote\_tx = CTxIn(COIN, COutPoint(0, 0))  
vote\_script = [Op.OP\_0, Op.OP\_PUSHDATA1, len(signature), signature, Op.OP\_CHECKSIG]  
vote\_out = COutPoint(1, 0)  
voting\_script = CScript([Op.OP\_IF, vote\_script, Op.OP\_ELSE, Op.OP\_DROP, Op.OP\_ENDIF])  
voted\_tx = CTxIn(vote\_tx, voting\_script)

* 1. Discussion Questions
  2. What are the advantages of using smart contracts on the BSV network compared to other blockchain networks?
  3. How can smart contracts improve the efficiency and security of traditional business processes?
  4. What potential challenges might arise when implementing smart contracts at scale on the BSV network?
  5. Hands-On Exercises
  6. Write a simple script that allows for the transfer of funds only if the sender's public key matches a specific one.
  7. Implement a smart contract to create a decentralized supply chain system where each step in the production process is recorded on the BSV blockchain.

# Module 4: Advanced Bitcoin SV Transactions  
  
## Lesson 4.1: Securing Your Wallet and Private Keys  
  
### Overview  
  
In this lesson, you will learn about securing your Bitcoin SV (BSV) wallet and private keys to ensure the protection of your digital assets.  
  
#### Learning Outcomes  
  
By the end of this lesson, you should be able to:  
  
1. Understand the importance of securing your BSV wallet and private keys.  
2. Implement best practices for wallet security, including the use of hardware wallets and multi-signature setups.  
3. Generate and manage backup phrases for disaster recovery.  
4. Secure your development environment to prevent unauthorized access.  
5. Understand the role of encryption in BSV transactions and how to encrypt messages using Bitcoin SV's built-in functionality.  
  
### Concept Explanation  
  
#### 1. Importance of Wallet Security  
  
Wallet security is essential for protecting your BSV assets from unauthorized access or theft. A compromised wallet can result in loss of funds, making it crucial to implement robust security measures.  
  
\*\*Real-world example\*\*: In 2011, an estimated 850,000 BTC (worth billions today) were lost due to poor storage practices and weak security measures.  
  
#### 2. Best Practices for Wallet Security  
  
- Use hardware wallets: These offline devices provide enhanced security by storing private keys away from potential threats online. Examples include Ledger Nano S and Trezor Model T.  
- Employ multi-signature setups: This requires multiple signatures (e.g., from multiple devices or parties) to approve a transaction, providing an additional layer of security.  
- Generate strong backup phrases: A backup phrase is a sequence of words used to recover your wallet in case of loss or theft. Use a tool like BIP39 to generate secure backup phrases.  
  
#### 3. Securing Your Development Environment  
  
Secure your development environment by using antivirus software, limiting access to sensitive files, and encrypting your hard drive. Additionally, always keep your operating system and applications up-to-date with the latest security patches.  
  
#### 4. Encryption in BSV Transactions  
  
Bitcoin SV offers built-in encryption functionality for secure communication between parties. The `OP\_ENCODEVERIFY` and `OP\_DROP` opcodes are commonly used to encrypt messages within transactions.  
  
```python  
import bitcoin  
from bitcoin.utils import encode\_base58, decode\_base58  
  
# Encrypt a message using Bitcoin SV's built-in encryption functionality  
def encrypt\_message(msg):  
 tx = bitcoin.core.Transaction()  
 input = bitcoin.core.TxIn(prev\_out=bitcoin.core.OutPoint(), script\_sig=bitcoin.script.CScript([bitcoin.opcodes.OP\_DUP, bitcoin.opcodes.OP\_HASH160, msg.hash, bitcoin.opcodes.OP\_EQUALVERIFY, bitcoin.opcodes.OP\_CHECKSIG]))  
 output = bitcoin.core.TxOut(nValue=0, pkScriptPubKey=bitcoin.script.CScript([bitcoin.opcodes.OP\_DUP, bitcoin.opcodes.OP\_HASH160, msg.hash, bitcoin.opcodes.OP\_EQUAL]) + bitcoin.script.ToWitness(msg))  
 tx.add\_input(input)  
 tx.add\_output(output)  
 txid = tx.hash  
 return encode\_base58(txid.serialize())  
  
# Decrypt a message using the encrypted transaction hash and private key  
def decrypt\_message(encrypted\_txid, privkey):  
 tx = bitcoin.core.Transaction().from\_hex(bitcoin.core.deserialize\_bch(decode\_base58(encrypted\_txid)))  
 signature\_hash = list(map(lambda txin: txin.signature\_hash(), tx.inputs))[0]  
 recovered\_public\_key, \_ = bitcoin.ecrecover(signature\_hash, privkey, recover=True)  
 public\_key = bitcoin.pubkey\_to\_address(recovered\_public\_key)  
 output\_script = tx.vout[0].scriptPubKey  
 message\_hash = output\_script[3][::-1]  
 message = bitcoin.hash160\_to\_bytes(message\_hash).hex()  
 return message.decode('utf-8')

* + 1. Discussion Questions
  1. What are some common reasons for wallet security breaches, and how can they be prevented?
  2. Why is it important to encrypt messages when conducting BSV transactions?
  3. How do hardware wallets contribute to improved wallet security compared to software wallets?
     1. Hands-on Exercises
  4. Research different hardware wallet options available on the market and compare their features, security protocols, and price points.
  5. Implement the provided Python code for encrypting and decrypting messages using Bitcoin SV's built-in encryption functionality. Experiment with different messages to understand the process better.

# Module 4: Bitcoin SV Transactions and Security  
  
## Lesson 4.2: Staying Informed and Protecting Your Investment  
  
This lesson focuses on the importance of staying informed about market trends and securing your Bitcoin SV (BSV) investment.  
  
### Learning Outcomes  
1. Understand the importance of monitoring BSV market trends.  
2. Learn how to securely store BSV using hardware wallets.  
3. Grasp the concept of multi-signature wallets for enhanced security.  
4. Recognize the role of Blockchain analytics tools in securing investments.  
5. Comprehend the implications of regulatory changes on BSV investment.  
  
### Monitoring BSV Market Trends  
Monitoring market trends is crucial to making informed decisions about your BSV investment. You can use various online resources, such as [CoinMarketCap](https://coinmarketcap.com/currencies/bitcoin-sv/) and [Blockchain.com](https://www.blockchain.com/btc/bsv), to track the price, volume, and market capitalization of BSV.  
  
\*\*Example:\*\* By following the price fluctuations of BSV on CoinMarketCap, you can identify opportunities for buying or selling your investment to maximize profits.  
  
### Securing Your BSV with Hardware Wallets  
Hardware wallets are physical devices designed to securely store cryptocurrencies like BSV. Examples include Ledger and Trezor. These devices provide an additional layer of security by storing the private keys offline, making them less susceptible to hacking or theft.  
  
```python  
from bsvpython import Key, HDKey, Wallet  
  
# Create a new HD wallet  
hd\_key = HDKey()  
wallet = Wallet(private\_key=hd\_key.to\_bytes())

Example: Connect your Ledger Nano X to your computer and use the Ledger Live application to manage your BSV holdings securely.

* + 1. Multi-Signature Wallets for Enhanced Security

Multi-signature wallets require multiple signatures (keys) to authorize transactions, providing an additional layer of security. This means that even if one key is compromised, the attacker would still need access to the other keys to steal your funds.

const bsv = require('bsv');  
const { HDKey } = bsv;  
  
// Create a multi-sig wallet with 2 of 3 signatures required (privateKeys[0], privateKeys[1], and privateKeys[2])  
const multiSigScript = bsv.scripts.multiSignature(2, [privateKeys[0].toPublicKey(), privateKeys[1].toPublicKey(), privateKeys[2].toPublicKey()]);

Example: Implement a 2-of-3 multisig wallet using the BSV Python library to secure your investment.

* + 1. Blockchain Analytics Tools for Securing Investments

Blockchain analytics tools help you monitor transactions on the blockchain and identify potential security threats, such as illegal activities or fraudulent transactions. Examples include Chainalysis and CipherTrace.

Example: Use Chainalysis to track the movement of BSV on the blockchain and identify any suspicious activity related to your investment.

* + 1. Discussion Questions
  1. What are some advantages of using a hardware wallet for storing BSV?
  2. How might multi-signature wallets help secure your BSV investment?
  3. What role do blockchain analytics tools play in ensuring the security of BSV investments?
     1. Hands-On Exercises
  4. Implement a 2-of-3 multisig wallet using the BSV Python library and test it by sending transactions to different addresses.
  5. Use Chainalysis or CipherTrace to monitor the movement of BSV on the blockchain and identify any suspicious activity related to your investment.

```markdown  
# Assessment Types  
  
This lesson provides an overview of various types of assessments in the context of Bitcoin SV (BSV) blockchain. By the end of this lesson, you should be able to:  
  
1. Understand different types of assessment methods used in BSV.  
2. Explain how each assessment type contributes to the overall security and transparency of the BSV network.  
3. Demonstrate the use of Python or JavaScript for creating and verifying various types of assessments on the BSV blockchain.  
4. Evaluate the trade-offs between different assessment methods in terms of scalability, privacy, and ease of implementation.  
  
## Overview  
  
Assessments play a crucial role in ensuring the integrity and security of transactions on the BSV blockchain. This lesson covers four main types of assessments: proof-of-work (PoW), proof-of-burn (PoB), proof-of-existence (PoX), and oracles.  
  
## Proof-of-Work (PoW)  
  
### Explanation  
  
Proof-of-work is the original consensus mechanism used by Bitcoin and is still employed by BSV. Miners compete to solve complex mathematical problems to validate transactions and add new blocks to the blockchain. The first miner to find a solution receives a reward in the form of newly minted coins.  
  
### Real-world Example  
  
In a BSV mining pool, miners collaborate to increase their chances of finding the next valid block. When a member of the pool successfully solves the PoW problem, the rewards are distributed according to each miner's contributed computing power.  
  
```python  
# Python example for calculating the mining reward based on mining power (hashrate)  
mining\_reward = 6.25 # BSV reward per block  
hashrate = 1000000000000 # Hashes per second  
reward\_per\_hash = mining\_reward / hashrate  
print(f"Mining reward per hash: {reward\_per\_hash} satoshis")

* 1. Proof-of-Burn (PoB)
     1. Explanation

Proof-of-burn is a consensus mechanism where participants "burn" (destroy) a certain amount of coins to prove their stake in the network and earn the right to create new blocks. The destroyed coins are sent to an unspendable address, effectively removing them from circulation.

* + 1. Real-world Example

To participate in PoB, users send a specified number of BSV coins to a specific "null" address, thus burning those coins. As a reward, they can then compete for creating the next block and receive newly minted coins as compensation.

// JavaScript example demonstrating the cost of burning 1000 BSV using an unspendable address  
const bsv = require('bsv');  
const address = bsv.Address.fromHash160('SHA-256:<YOUR\_NULL\_ADDRESS\_HASH>');  
const tx = new bsv.Transaction();  
tx.addOutput(address, 1000 \* Math.pow(10, 8)); // Add output of 1000 BSV to the null address  
const privateKey = bsv.PrivateKey.fromWIF('<YOUR\_PRIVATE\_KEY\_WIF>');  
tx.sign(privateKey);  
const serializedTx = tx.serialize();  
// Broadcast the transaction to the network

* 1. Discussion Questions
  2. How do PoW and PoB compare in terms of energy consumption and computational power requirements?
  3. What are the potential advantages and disadvantages of implementing PoB as a consensus mechanism on BSV?
  4. Can you think of any real-world applications for PoB beyond participating in the BSV blockchain?
  5. Hands-on Exercises
  6. Write Python or JavaScript code to calculate the number of BSV coins burned when sending 1000 BSV to a null address with your chosen implementation's specific hash.
  7. Implement a simple Python script that simulates a mining pool and distributes rewards based on individual miner hashrates.

# Bitcoin SV Fundamentals Assessment  
  
## Multiple-Choice Questions (3)  
  
1. Which of the following statements about Bitcoin SV is true?  
 - a) Bitcoin SV was created as a result of a hard fork from Bitcoin Cash in 2018.  
 \*correct answer\*  
 - b) Bitcoin SV is an implementation of the original Bitcoin protocol with increased block size limits.  
 - c) Bitcoin SV stands for "Secure Verification System."  
 - d) Bitcoin SV was created before Bitcoin Cash.  
  
2. Which of the following is not a unique feature of Bitcoin SV?  
 - a) Increased block size limit, allowing more transactions per second.  
 \*incorrect answer\*  
 - b) Implementation of the original Satoshi's Nakamoto consensus and scripting rules.  
 \*correct answer\*  
 - c) Support for on-chain scaling solutions for micropayments and enterprise solutions.  
 \*correct answer\*  
 - d) A focus on preserving the original vision of Bitcoin as a peer-to-peer electronic cash system.  
 \*incorrect answer\*  
  
3. Which of the following is the correct Bitcoin SV block size limit?  
 - a) 8 MB  
 - b) 1 GB  
 \*incorrect answer\*  
 - c) 2 GB (temporarily increased in Nov 2018)  
 \*incorrect answer\*  
 - d) 128 MB  
 \*incorrect answer\*  
 - e) 4 GB  
 \*correct answer\*  
  
## Short-Answer / Definition Questions (2)  
  
1. What is the significance of increasing the block size limit in Bitcoin SV?  
  
2. Explain how Bitcoin SV aims to preserve the original vision of Bitcoin as a peer-to-peer electronic cash system.  
  
## Practical Coding Exercise  
  
### Starter Code  
  
```python  
import requests  
from bs4 import BeautifulSoup  
  
def get\_bsv\_blocks():  
 url = "https://api.bitcoinsv.io/public/v1/blockchain/stats"  
 response = requests.get(url)  
 soup = BeautifulSoup(response.text, 'xml')  
 total\_blocks = int(soup.find('totalblocks').text)  
 return total\_blocks  
  
total\_bsv\_blocks = get\_bsv\_blocks()  
print(f"Total Bitcoin SV blocks: {total\_bsv\_blocks}")

* + 1. Exercise

Write a Python script that calculates the average block time for Bitcoin SV. You can find the timestamp of each block in the Bitcoin SV API. Make sure to handle edge cases, such as when no data is available or when there are not enough blocks to calculate a meaningful average.

Hint: Python's datetime library can be useful for handling timestamps.