kraken INTELLIGENCE
ORDO AB CRYPTO

# Taproot Primer
# An Upgrade For The Ages

September 2021

# Executive Summary

*Kraken Intelligence would like to give a special thanks to Murch from Chaincode Labs for his generous review of this note and feedback.*

Many would contend that Bitcoin is the most important invention since the Internet's inception in 1983. Despite its nascency, Bitcoin has some limitations that have hitherto hampered mass adoption, including scalability and privacy concerns. Historically, optimizing Bitcoin's protocol has been a continuous process of exploring new ideas amongst community members to reach a consensus. However, achieving a consensus is onerous and infrequent because there has been significant controversy amongst Bitcoin's vast and diverse user base about how Bitcoin should evolve. Consider that Bitcoin's latest upgrade was the 2017 Segregated Witness (SegWit) soft fork, which led to a split in the community and a hard fork that created Bitcoin Cash (BCH).

Bitcoin is once again set to undergo a significant upgrade in November 2021. The upgrade, known as "Taproot," consists of three major Bitcoin Improvement Proposals (BIP) that will introduce several features to enhance the network's privacy, security, and scalability. The first BIP will upgrade Bitcoin's core cryptography by integrating the more secure and lightweight Schnorr digital signature scheme into Bitcoin in parallel with its existing scheme, Elliptic Curve Digital Signature Algorithm (ECDSA). The second BIP, also known as "Taproot," builds on the SegWit upgrade to improve privacy, lower transaction fees, and expand upon scaling solutions such as Bitcoin's Lightning Network (LN). Finally, the third BIP, known as "Tapscript," integrates a reformed version of Bitcoin's scripting language to enable schnorr/taproot transactions and make future Bitcoin upgrades easier.

Together, these improvements will allow standard single-signature (single-sig) transactions, multi-signature (multisig) transactions, and complex smart contracts all look identical on the blockchain. Thus, Taproot substantially improves user privacy, network scalability, and the fungibility of all bitcoins. Users will enjoy cheaper transaction fees and the ability to conduct multisig transactions and create complex smart contracts with the same efficiency, low fees, and privacy as single-sig transactions. Taproot is arguably the most significant Bitcoin upgrade to date, setting the stage for massive innovation and adoption. Those who understand its ins and outs will walk away with a deep understanding and appreciation for what this revolutionary technology has to offer the world. In this report, Kraken Intelligence provides a comprehensive analysis of the Taproot upgrade, how it works, and why it matters.

## Table of Contents

# 1.

# Introduction

## What Is Taproot

Taproot is a soft fork upgrade to the Bitcoin protocol that bundles various technical improvements to make transactions more private, flexible, and scalable. It is one of the most highly anticipated upgrades in all of Bitcoin's history and consists of the following three Bitcoin Improvement Proposals (BIPs):

### *BIP340 (bip-schnorr)*

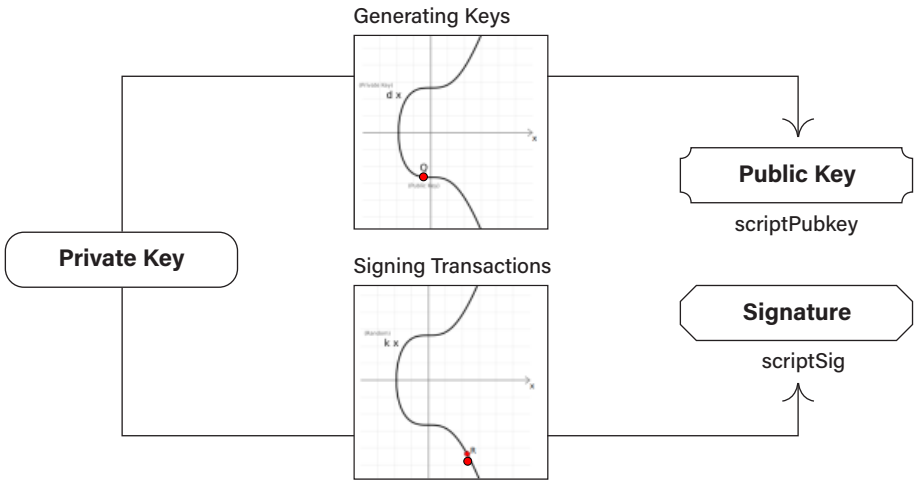
Bitcoin has traditionally used an Elliptic Curve Digital Signature Algorithm (ECDSA) scheme, an application of public-key cryptography, to authenticate transactions. Put simply, a digital signature system allows users to generate a public and private key pair where the private key can generate a digital signature that proves ownership of the public key without revealing the private key.

BIP340 will outfit Bitcoin with the "Schnorr signatures" scheme, in addition to the network's existing ECDSA scheme. Like ECDSA, Schnorr uses a mathematical structure called an “elliptic curve” as the basis for its digital signature system. Elliptic curves, derived from algebraic geometry, are applied in various public-key cryptography systems to provide some of the most secure encryption protocols. An elliptic curve consists of all the points that satisfy a simple equation in the form of “y2 = x3 + ax + b.” Functions defined on an elliptic curve are easy to perform but nearly impossible to invert.

Figure 1 shows that public keys and digital signatures are points on an elliptic curve, though only a part of the signature is a point on the curve. If both of these points are created from the same private key, a geometric connection between them proves that the person who created the digital signature also owns the public key. In essence, ECDSA is a mathematical way to prove bitcoin ownership without giving away access to the bitcoin(s) or any other private information.

*Figure 1*

**Elliptic Curve Digital Signature Algorithm (ECDSA)**




Source: Kraken Intelligence

Schnorr allows for key and signature aggregation, a feature that effectively reduces the network's data load, improves privacy, and allows for a faster transaction verification process, among other benefits. Multisig transactions, or transactions that require multiple signatures, are easily identifiable on the blockchain because they expose multiple public keys and signatures, while single-sig transactions only expose one key and signature per counterparty. Additionally, the amount of block space consumed by multisig transactions increases as more participants become involved.

Schnorr's key aggregation feature combines all the public keys involved in a multisig transaction, requiring participants to perform signature aggregation to produce a single signature for the aggregated key. This technique effectively obscures that the transaction involves more than two users and reduces the size of the transaction from a larger multisig transaction to a smaller single-sig transaction. Schnorr also enables bip-taproot (BIP341), meaning bip-taproot is infeasible without Schnorr.

### *BIP341 (bip-taproot)*

Adds "Taproot," a privacy solution that utilizes Schnorr signatures to enable Merklized Alternative Script Trees (MAST). MAST uses Merkle trees, or a data structure used for data verification and synchronization, to make smart contracts more efficient and private by solely revealing the condition of the contract that was met, not any of the other conditions that weren't met.[2]

### *BIP342 (bip-tapscript)*

Introduces "Tapscript," an upgraded scripting language that will complement Schnorr and Taproot by improving signature hashing to validate Taproot scripts. Bip-tapscript introduces flexibility to enhance Bitcoin's smart contract capability and changes some limits for resource requirements, including the removal of the 10,000-byte size limit that exists in legacy scripts. Tapscript provides a forward-compatibility mechanism, known as "tagged public keys," that makes it easy for future soft forks to extend the signature-checking opcodes with new sighash types or other changes.[3]

All in all, the soft fork's main feature will allow multisig transactions and complex smart contracts to be indistinguishable from standard transactions via key aggregation. As a result, even the most complex smart contracts will be virtually identical to regular transactions on the blockchain. Thus, Taproot's signature aggregation technique can improve the privacy of the Lightning Network (LN), an application built upon Bitcoin to improve scalability, by making LN-related transactions look like single-sig transactions.

The LN allows users to conduct "off-chain" transactions to reduce network congestion by opening multisig smart contracts known as "channels" between other users. The users can transact with each other in these channels without needing to broadcast every transaction to the blockchain, meaning users can send bitcoin in these channels instantly with nearly zero fees. The only LN-related transactions that are broadcast to the network are the opening and closing of channels. Blockchain sleuths can quickly identify these transactions as complex smart contracts due to public on-chain data such as the number of public keys and signatures included in the channel or because of the script being used. Not only does this significantly reduce Bitcoin user privacy, but it also adds unnecessary network congestion that weighs on network scalability and could potentially affect bitcoin fungibility if a sleuth discovers a blacklisted wallet in a channel.

However, Taproot will allow the opening and closing of channels to look like standard single-sig bitcoin transactions. By packaging signatures, the details of a multisig transaction can remain private, and users can pay lower transaction fees. As we discuss later in the report, the significance of key-aggregation should not be overlooked because it effectively enables complex layer 2 networks on Bitcoin to create massive multisig vaults (e.g., requiring more than 1,000 signatures) that were previously not viable.

## Origins

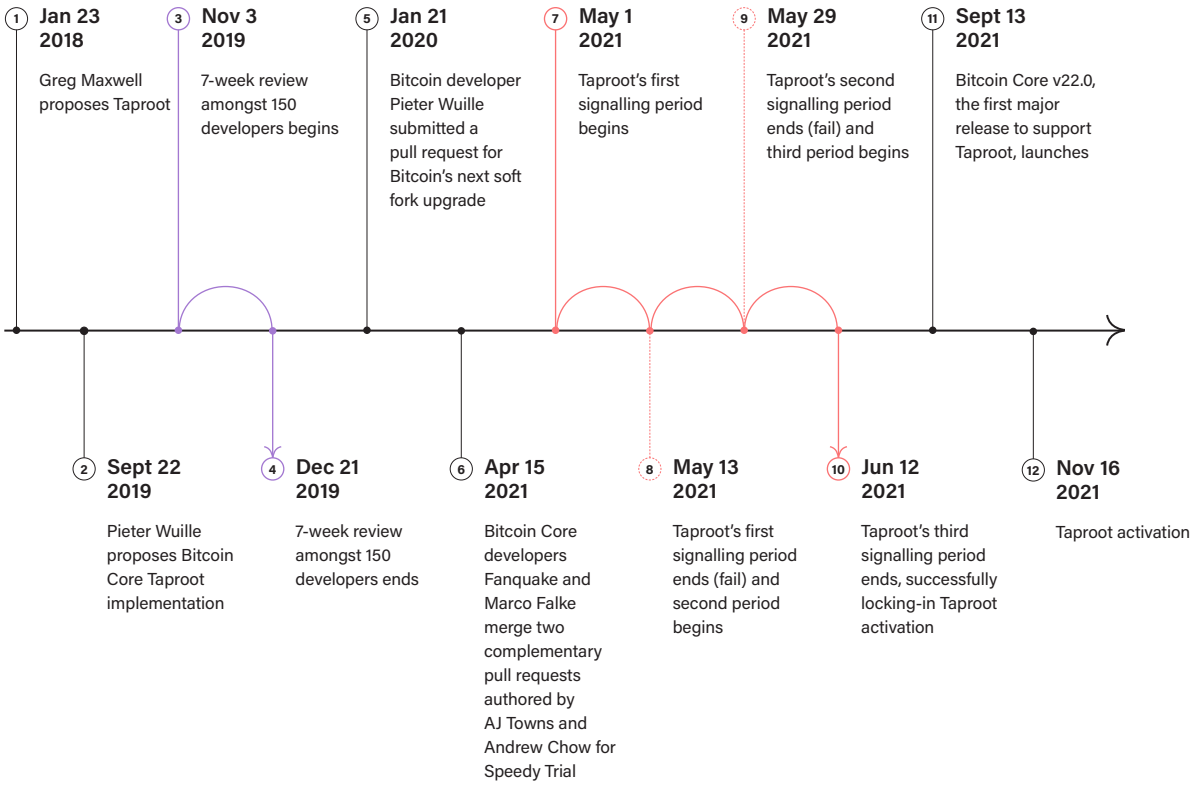
Taproot's development began nearly four years ago after former Blockstream CTO and prominent Bitcoin core developer Gregory Maxwell proposed the concept in late **January 2018** to address Bitcoin's scalability, privacy, and security concerns.[4] It was not until late **September 2019** that Bitcoin developer Pieter Wuille proposed implementing Taproot into Bitcoin Core.[5] Shortly thereafter, Taproot began gaining significant momentum. The proposal was reviewed and tested by 150 developers between **November 3rd, 2019** and December 21st, 2019.[6]

## Activation Timeline

The upgrade reached consensus from an overwhelming majority of stakeholders on June 12th, 2021, at block 687,284 at the end of the third signaling period. Because miners have six months to prepare for the upgrade, Taproot is expected to activate via soft fork at block 709,632 on November 16th, 2021.

*Figure 2*

**Taproot Timeline**




Source: Kraken Intelligence, GitHub, CoinDesk

## Why It Matters

Upgrades to the protocol are rare because Bitcoin is an iterative technology that comes with financial risks—the underlying code in any upgrade can contain bugs or unintended consequences. Consensus on upgrades also generally requires that the upgrade is backward-compatible from the perspective of existing Bitcoin node software and that the upgrade has negligible trade-offs with regards to node software resource consumption, such as electricity usage and required disk space. Therefore, Bitcoin upgrades must remain infrequent to give developers ample time to extensively test the proposed upgrade's code.

Additionally, Bitcoin's decentralization requires meticulous coordination between participants worldwide to ensure an upgrade is properly implemented. Making any amendments to Bitcoin is onerous because there is no central authority to decide what changes get pushed through, and many key stakeholders often disagree on what is best for the protocol.

SegWit, which paved the way for the development of the Lightning network, was an extremely controversial upgrade because organizations attempted to bundle it with a hard fork proposal.[7] The initial purpose of SegWit was to fix Bitcoin's transaction malleability problem, an attack that allows a bad actor to change a bitcoin transaction's unique ID (TXID) before confirmation on the blockchain. This issue made it possible for a bad actor to pretend that a transaction did not happen, allowing them to make a double withdrawal on a poorly-programmed crypto exchange. The attack consisted of a bad actor that would make a withdrawal request on an exchange, prompting the exchange to process the requested withdrawal and save the transaction ID (TXID) in their database. The attacker would then malleate the transaction and resubmit it with a different TXID, which gets mined on the blockchain. The exchange never detects that the original transaction was mined because the attacker modified the TXID. Afterward, the bad actor would reach out to the platform to say their withdrawal was never confirmed even though it was. Thus, the exchange processes another withdrawal for the attacker because they cannot confirm that the initial withdrawal went through.

In coming up with a fix for Bitcoin's transaction malleability, developers moved digital signatures (often called a witness) from the part of the transaction data that is hashed to make a TXID, to another area of the transaction where malleated signatures would not change the TXID. In doing so, they realized they could discount this witness data when accounting for the size of a transaction, implicitly freeing up block space. In the end, the secondary focus of SegWit was to scale the network. However, the community's attempt at scaling resulted in a heated philosophical debate about whether Bitcoin's block size should increase to allow for more transaction throughput or if this would damage the network's decentralization by materially raising the costs of running a node to data center levels.

The debate resulted in a "civil war" amongst the community and a hard fork that created "Bitcoin Cash" for those who wanted a version of Bitcoin with bigger blocks. Unlike SegWit, there is no contemporaneous hard-fork proposal with Taproot. The support from miners was profuse with little-to-no conflict, implying the community strongly agrees that Taproot will substantially improve Bitcoin. The considerable challenge was choosing the optimal activation method, having learned from the last onerous upgrade.

Though it may only seem like an incremental improvement to Bitcoin, Taproot revolutionizes the Bitcoin network by massively enhancing privacy, making bitcoins more fungible, improving network scalability, and setting up foundational infrastructure to deploy future changes more easily. At a high level, Taproot will enhance Bitcoin’s efficacy as a medium of exchange by allowing for more transaction throughput, upgrade the network’s core cryptography that ensures network security, and inspire more developers to work on Bitcoin by enabling complex DeFi smart contracts, among others. Ultimately, Taproot will translate to an overall smoother user experience and roll out the red carpet for vast innovation on the Bitcoin network.

# 2.

# How Bitcoin Transactions Work

Before breaking down the inner workings of Taproot, a foundational understanding of how bitcoin transactions work is required. Readers with a fundamental understanding of Bitcoin's Unspent Transaction Output (UTXO) model should skip over this section to "Taproot Explained" on page 19.

## Public-Key Cryptography

Bitcoin wallets function similarly to traditional bank accounts in that both an "account number" and "password" are required to access the funds held in the wallet. When a user creates a wallet, they randomly generate a unique cryptographic key pair of one public key and one private key to allow the user to send or receive bitcoin. The public key acts as the "account number," and the private key like the "password." This is known as public-key cryptography, whose advent in 1976 set the stage for the creation of the Internet and now cryptoassets.

Figure 3 illustrates the **four basic components of a Bitcoin wallet**. The following example is for illustrative purposes; please do not send funds to this wallet or claim it as your own—everyone can use it as they have the private key!

*Figure 3*

**Components of a Bitcoin Wallet**

## Seed Phrase

Any Bitcoin wallet starts by creating a seed phrase. The seed phrase is usually converted into a list of 12–24 words that can be used to back up and restore the wallet. Anyone who knows the seed can drain the wallet, so it is critical **never** to type the seed into any computer.

***Example of a Bitcoin seed:***
valley pulp iron unique pen tired energy crash topic business happy feel

## Private Key

From that seed, the Bitcoin wallet generates a public and private key. A Bitcoin private key is a randomly generated, large number that acts as the wallet's password to **spend** bitcoin. However, this number is so large that it is displayed in hexadecimal format (a combination of letters A-F and numbers 1-9) to make it more compact and easier to use. A private key can create a unique digital signature that's used to move funds into the custody of another digital wallet.

***Example of a Bitcoin private key:***
10d05ad23efa7e3962bab3954b353f6ab00ecf865c28b437ccdb74ab72d4f87b

## Public Key

A Bitcoin public key is a hexadecimal number derived from the private key. Public keys act as the account number to **receive** bitcoin.

***Example of a Bitcoin public key:***
0292e91a485dc4d817693545305eca83fa19d5ee1d3bbd9bbc5351e65ddca3e0f3

## Address

The Bitcoin wallet creates addresses by hashing the public key to shorten it for easier usage. The applied hash function takes the public key data, scrambles it, and returns a unique, shortened output. This result acts as a digital fingerprint for the public key data that was inputted in the function. A wallet address can then be shared with others to "receive" bitcoin.

***Example of a Bitcoin address:***
19GJ727x4TQ74WWb8WpGQUAapX38wwV1iD

## The Unspent Transaction Output (UTXO) Model

> "We define an electronic coin as a chain of digital signatures. Each owner transfers the coin to the next by digitally signing a hash of the previous transaction and the public key of the next owner."
>
> – *Satoshi Nakamoto explaining the UTXO model in the Bitcoin Whitepaper (2008)*

Bitcoin utilizes the UTXO model to account for the distribution of coins amongst users. Put differently, the UTXO model is Bitcoin's method for organizing the blockchain's ledger such that no coins are spent twice. At its core, each UTXO is a quantity of bitcoin linked by an address to a private key. Bitcoin transactions consist of inputs and outputs; inputs refer to UTXOs consumed by a transaction (i.e., UTXOs spent), while outputs are the UTXOs created by a transaction (i.e., UTXOs received). Because only unused outputs can be used as inputs to a transaction, UTXOs can only be spent once. If someone were to try to use an output that was previously spent in another transaction, nodes would reject it.

In a bitcoin transaction, the sender spends a UTXO(s) by using it as an input while the recipient obtains a newly-created UTXO(s) that they can spend in future transactions. A digital signature—the result of cryptographically applying a private key to data that proves UTXO ownership without exposing the private key—is required to unlock and spend UTXOs.

Moreover, UTXOs are not spent partially under any circumstances. It is helpful to think of UTXOs as a receipt that shows the balance amount or change in a given transaction. For example, imagine someone wants to buy a pair of shoes that cost $35. Because $35 bills do not exist, the purchaser would pay with a $50 bill or several bills that add up to $35 (e.g., paying with a $20, $10, and $5 bill). UTXOs function in a similar way in that they are always spent in their entirety.

If somebody wanted to purchase a car that costs ₿1, but their wallet only has a single UTXO that contains ₿2, they would send the full ₿2 and receive ₿1 back as change. The recipient would also receive a single UTXO associated with ₿1. However, if that user had various UTXOs of ₿0.4, ₿0.1, ₿0.2, and ₿0.3, all four UTXOs would be used as the inputs of the transaction. The inputs in this transaction would then be added up to a single ₿1 UTXO.

**Example of a Standard Bitcoin Transaction**

1. Alice owes Bob ₿0.02 for fixing her car.
2. Bob sends Alice his public wallet address to receive payment.
3. Alice creates a digital signature using her private key to unlock and send ₿0.02 from UTXOs associated with her wallet to Bob's public wallet address.
4. Assuming Alice's wallet only has two UTXOs worth ₿0.01 and ₿0.015, she would send both UTXOs as the transaction's inputs.
5. Two UTXOs are created as the unused outputs of the transaction: a UTXO associated with ₿0.02 is sent to Bob's wallet, and a UTXO worth ₿0.005 is sent back to Alice as change.

*Figure 4*

**Advantages and Disadvantages of the UTXO Model**

| Pros | Cons |
|---|---|
| Solves the double-spending problem | Not user-friendly |
| Encourages privacy as users are highly recommended to create a new address for every incoming transaction, making it difficult to link different coins to a single owner | Limited smart contract capability (i.e., Turing incomplete), meaning complex computations are impractical |
| Allows transactions to be verified in parallel | Data-intensive |

Source: Kraken Intelligence

## Bitcoin Script

UTXOs are coded in Bitcoin Script, a deliberately constrained programming language that gives the Bitcoin software instructions on how to spend coins in a UTXO. When sending bitcoin(s), users must execute a locking script on each output in the transaction (newly-created UTXOs). The recipient must then execute an unlocking script to spend the bitcoin tied to the UTXOs. Scripts include two elements:

### *Data*

Critical information needed for the transaction to occur, such as a digital signature and public key.

### *Operation Codes (Opcodes)*

Simple functions (i.e., commands) that operate on the script's data. Opcodes determine the instructions recorded with each complex transaction, enabling users to set spending conditions on their UTXOs.

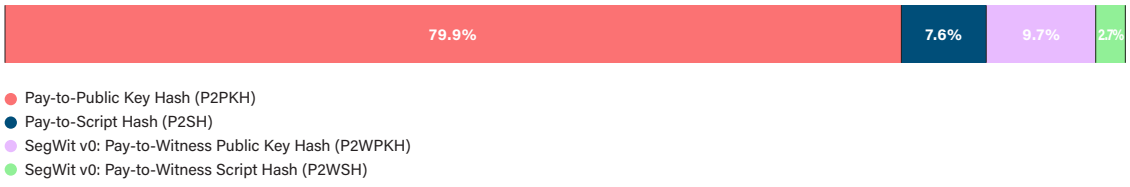
When nodes on the network agree that a complete script is valid (i.e., locking and unlocking conditions are met), the UTXO is unlocked and can be spent. While every transaction requires signature checks to prove UTXO ownership, some optional spending conditions include:

- **Multisig**: Requires multiple private keys to provide digital signatures.
- **Hashlocks**: Requires a particular piece of data to be publicly revealed.
- **Timelocks**: Requires a block height or date to be reached.

Several conditions can be utilized for each transaction to create complex types of smart contracts. For instance, when Alice sends Bob ₿0.02 for car repairs, Bob can lock the UTXOs with spending conditions (multiple signatures or a certain date) before Bob can spend the UTXOs Alice sent him. Although there are many different locking scripts with various combinations of opcodes, most nodes only relay a handful of "standard scripts." These common scripts include Pay-to-Public Key Hash (P2PKH), Pay-to-Script Hash (P2SH), and the SegWit versions of these scripts, Pay-to-Witness Public Key Hash (P2WPKH) and Pay-to-Witness Script Hash (P2WSH).

*Figure 5*

**Distribution of Script Usage (June - September 2021)**




Source: Clark Moody's Dashboard

## Pay-to-Public-Key Hash (P2PKH) Script

Most standard bitcoin transactions utilize the P2PKH script to lock bitcoin to the hash (i.e., condensed version) of a public key. Basic transactions sent to an address contain P2PKH scripts that are resolved by sending the public key and a corresponding digital signature. However, because all spending conditions are publicly revealed during P2PKH transactions, complex transactions with various opcodes require more advanced scripting to preserve privacy.

## Pay-to-Script Hash (P2SH)

P2SH is a standardized smart contract on Bitcoin that is more flexible than P2PKH because it allows users to construct arbitrary scripts and multisig transactions. Complex transactions with various opcodes typically utilize P2SH to preserve privacy as the transaction's spending conditions are not publicly visible at first. Unlike P2PKH, which locks bitcoin to the public key's hash, P2SH locks bitcoin to the hash of the script itself. This means that UTXOs locked using P2SH can only be spent if the hash of the script is provided. Because only a hash of the script is included on the blockchain initially, only the new UTXO owner knows the conditions in which they can be spent.

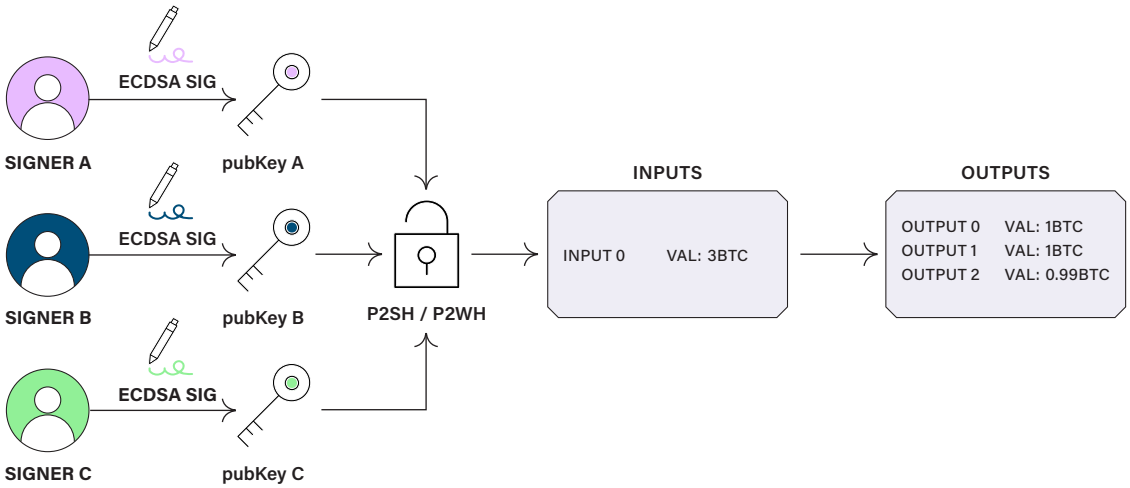
Though the spending conditions are initially hidden from the public, the entire script and associated spending conditions are revealed as soon as the new UTXO owner spends the coins. Furthermore, P2SH inefficiently requires public key knowledge of all participants of a multisig transaction. Because the initial hash of the script can be used to verify that the script was valid, this public reveal is a feature and not a bug. Nevertheless, revealing all possible conditions that could have been met every time coins are spent is both data-intensive and a breach of privacy.

Blockchain sleuths can see private information upon further examination of these scripts, such as what type of wallet was used to conduct a transaction. Moreover, the script's exposed conditions enable these blockchain sleuths to easily distinguish standard P2PKH transactions from multisig transactions. A savvy bad actor can use this private information to scam or hack exposed users in the worst-case scenario. This excess of opcode data also impedes network scalability because it eats up more block space.

P2SH transactions are easily identifiable because they require different addresses that begin with "3," as discussed in BIP13.[8] Not only does this allow P2SH transactions to be identified, but it also narrows down the identities of participants in the multisig transaction. P2SH is a poor long-term solution in terms of operational efficiency due to its privacy and security concerns.

*Figure 6*

**P2SH Transaction Example**




Source: Kraken Intelligence

# 3.

# Taproot Explained

## BIP340 (bip-schnorr)

Bitcoin uses ECDSA digital signatures to enable the transfer of coins. However, the first phase of the Taproot soft fork will enable Schnorr signatures in addition to ECDSA signatures, both of which can utilize elliptic curve cryptography (ECC). ECC is a type of cryptography that uses elliptic curves to encrypt data such that the authenticity of the encrypted data is easily verifiable, but the data is nearly impossible to access without a "decryption key." For perspective, a Universal Security study found that breaking a standard 228-bit ECC key would require more energy than it would take to boil all the water on earth.[9]

Bitcoin utilizes ECC to allow users to prove private key ownership with a digital signature that does not expose their private key to the public. While public-key cryptography fundamentals have barely changed since inception, dozens of open-source signature schemes are now available to cryptographers worldwide.

Not only would Taproot be impractical without Schnorr, but cryptographers and Bitcoin veterans generally agree that Schnorr is better than ECDSA in almost every way with virtually no trade-offs. Some of these benefits over ECDSA include:

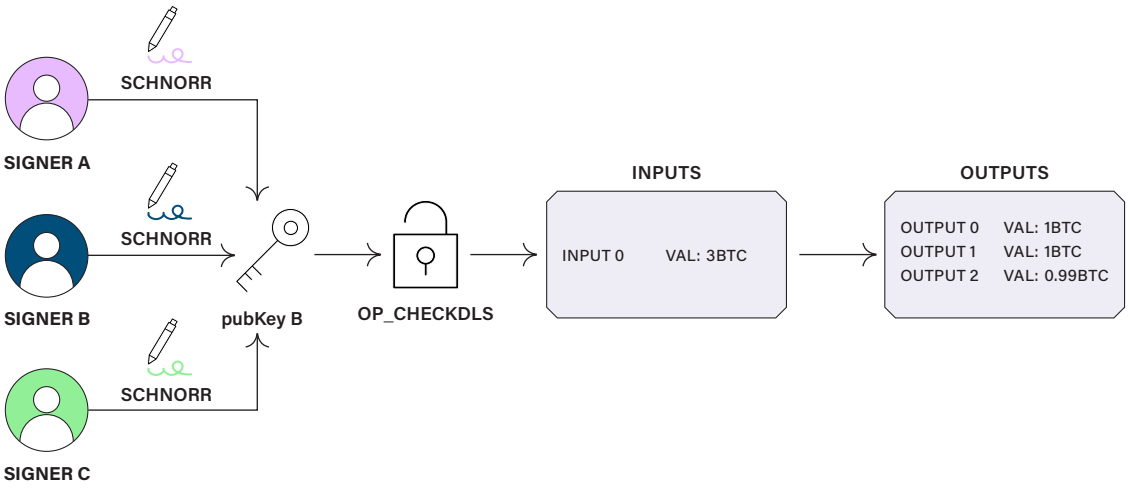
### *Linear Verification Algorithm:*

Schnorr's most significant advantage over ECDSA is its linearity, which is the foundation for higher-level smart contracts with improved efficiency and privacy. This feature allows multiple parties of a multisig transaction to collaborate to combine their public keys and produce a single signature that is valid for the

sum of their public keys, known as "key aggregation." By aggregating public keys and signatures into "threshold public keys" and "threshold signatures," Schnorr's linearity shortens the transaction verification process. Namely, the network only needs to verify one key rather than multiple, significantly decreasing the size of multisig transactions and making them indistinguishable from any regular transaction on the blockchain.

*Figure 7*

**Key Aggregation Example**




Source: Kraken Intelligence

### *Faster Verification*

In addition to the verification speed benefits from Schnorr's linearity, the scheme will also modify an opcode for Tapscript (OP_CHECKSIG) to expect a Schnorr signature instead of an ECDSA signature. Moreover, Schnorr transactions will replace ECDSA's opcode family used to verify multisig transactions (OP_CHECKMULTISIG and OP_CHECKMULTISIGVERIFY) for a single, more efficient opcode (OP_CHECKSIGADD). The new opcode improves signature verification efficiency because it allows transactions to be verified in batches rather than individually and requires fewer opcodes for multisig transactions. As a result, Bitcoin users will enjoy increased block space that will allow for more transactions processed per second.

### *Greater Economic Density*

Schnorr signatures and public keys are roughly -11% and -3% smaller than ECDSA signatures and public keys, respectively. Schnorr's lightness allows the network to free up block space and scale more efficiently. As such, users can pay less fees per transaction due to the decrease in network congestion.

### *Backward Compatibility*

Schnorr signatures will not cause any standardization issues on Bitcoin because they are compatible with ECDSA private keys. Thus, Schnorr users will not experience any problems where they cannot conduct a transaction because the counterparty's wallet was created with ECDSA and vice-versa. Upgrades to the Bitcoin network must be backward compatible if network participants adopt them because it will take some time for exchanges, custodians, and wallet services to implement the change. If bip-schnorr were backward incompatible, it would be impractical for Bitcoin service providers to adopt the development because it would make Bitcoin more complex to use. For instance, users looking to conduct a transaction would have to know the wallet software the counterparty uses to avoid accidentally burning coins. Luckily, bip-schnorr mitigates this problem by allowing for ECDSA private keys to sign transactions.

### *Provable Security*

Schnorr signatures are provably secure because they include an easily verifiable security proof. In contrast, ECDSA signatures are only secure in practice. Though no attacks have successfully cracked ECDSA, a formal security proof is yet to be discovered.

David Pointcheval and Jacques Ster proved in a 1998 paper that Schnorr signatures are unforgeable under chosen message attacks (i.e., SUF-CMA), guaranteeing that the only way to break Schnorr signatures is by solving a virtually impossible mathematical equation known as the elliptic curve discrete logarithm problem (ECDLP).[10] On the other hand, the best results for the provable security of ECDSA rely on stronger assumptions rather than a verifiable proof.

Although Schnorr signatures carry many benefits over ECDSA, the scheme is not an innovation within cryptography. German cryptographer and academic Claus-Peter Schnorr invented Schnorr as a Frankfurt professor and researcher in the 1980s. However, Schnorr was possessive over his innovation. Since its inception, the cryptographer encumbered Schnorr signatures behind a patent, restraining its unrestricted use and thus hampering further innovation. The Schnorr patent finally expired in 2008, which was the same year Satoshi Nakamoto invented Bitcoin. Since Schnorr patented the cryptographic signature scheme for nearly 20 years, most innovation revolved around the open-source ECDSA scheme. At the time, Schnorr signatures lacked the utilization, standardization, and testing in existing software to secure a global monetary system, such as Bitcoin. In sum, Satoshi chose ECDSA over Schnorr in 2008 to secure bitcoin transactions for several reasons:

### *Patent-Free*

Unlike Schnorr, ECDSA was not encumbered by patents or copyright, meaning there were no legal restrictions with using it for the Bitcoin network.

### *Standardized*

Because ECDSA was not protected by patent or copyright, most cryptographers

were already accustomed to ECDSA. Also, ECDSA was considered "Lindy" at the time. In other words, because ECDSA's security was sufficiently established by years of rigorous testing, it was also likely to have a longer remaining life expectancy.

### *OpenSSL*

ECDSA was in OpenSSL, a set of open-source encryption tools developed by cypherpunks to improve the privacy of online communications. OpenSSL made Bitcoin's ECDSA implementation at inception a relatively simple process.

Even though ECDSA had known flaws and was regularly criticized by cryptographers, it was open-source, lightweight, established, and trustworthy. Compared to other popular signature schemes, ECDSA was the optimal choice for digital money because it had leaner computational requirements, shorter key lengths, yielded roughly the same level of security, and met industry standards. ECDSA's implementation on Bitcoin has seen improvements since inception, such as the work of Pieter Wuille and colleagues on an improved elliptic curve specific to Bitcoin (known as secp256k1) that made signatures faster and lighter. Nevertheless, ECDSA inevitably has fundamental flaws that justify upgrading the entire system.

## BIP341 (bip-taproot)

While bip-schnorr detailed Schnorr's specifications on Bitcoin, bip-taproot specifies the rules for the new Pay-to-Taproot (P2TR) output type. P2TR is an upgrade from SegWit version 0 (i.e., P2WPKH and P2WSH) to version 1 to allow for Schnorr signatures. Bip-taproot aims to improve the privacy, efficiency, and flexibility of Bitcoin's scripting capabilities without negatively affecting security.

Bip-taproot leverages Schnorr signatures to include more complex transaction types to appear the same as standard single-sig transactions on the blockchain:

- Combines P2PKH and P2WPKH (i.e., single-sig transactions).
- Opening/closing LN channels, atomic swaps, and other complex smart contract

protocols that could result in all parties agreeing on an outcome.

- Transactions using MuSig, a new and improved protocol for aggregating public keys and signatures for the Schnorr digital signature algorithm (similar to P2SH and P2WSH multisig).

Non-P2TR transaction types, such as P2PKH and P2SH, inherently must reveal all the possible conditions that could have been met. As previously mentioned, this problem unnecessarily adds to network load and significantly hinders user privacy. Bip-taproot enables any Bitcoin smart contract (P2TR) to theoretically scale to the size of, for example, a +10,000-member decentralized autonomous organization (DAO) with thousands of bitcoins locked while costing and looking the same as a standard single-sig transaction.

For instance, imagine someone entering a bar needs to show the bouncer their ID to prove their age. However, doing so would expose private information utterly irrelevant to the situation, such as their home address, license number, organ donor status, whether they wear contact lenses or not, and more. Most people would agree that it would be ideal to prove to the bouncer their age without revealing all this unnecessary information. Legacy bitcoin transactions function similarly in that the spending conditions that were not executed add no value by being publicized. Bip-taproot aims to fix this privacy problem with its usage of merklized alternative script trees (MAST).

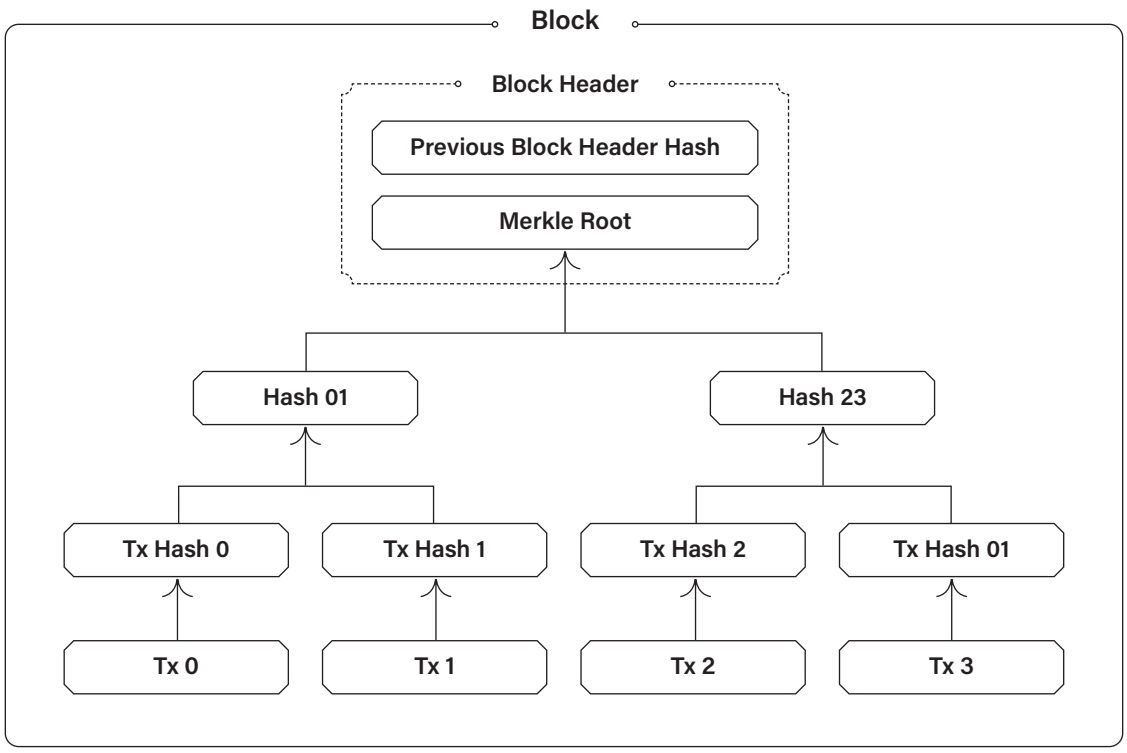
To understand the network storage benefits that Taproot brings, consider that on April 20th, 2021, the average bitcoin transaction fee reached an all-time high of $62.79. Bitcoin's "mempool" (memory pool), which acts as a waiting room for unconfirmed transactions to be picked up by miners and included in a block, became the most congested it has been since the bull run that ended December 2017. The surge in transaction fees came off the back of bitcoin falling -37% over five days, which tells us that the network saw a surge in transactions as market participants moved funds. Had Taproot been active and widely utilized at the time of the price crash, Bitcoin's mempool would have likely seen less congestion and transaction fees would have been lower; this potentially could have allowed for anyone to transact irrespective of the amount of bitcoin being transferred.

## MAST (Merklized Alternative Script Trees)

Bip-taproot implements MAST, a privacy solution that uses Merkle trees to solve the P2SH and P2PKH problem where all spending conditions are revealed during a transaction. Ralph Merkle, one of the inventors of public-key cryptography and the inventor of cryptographic hashing, patented the concept of Merkle trees (also known as hash trees) from 1979 to 2002.[11,12] A Merkle tree is an efficient way of proving that some data exists in a set without knowing the entire set. Specifically, a Merkle tree is a data structure "tree" in which every "leaf" node contains the hash of a data block, and every "non-leaf" node contains the hash of its child nodes. Leaf nodes can have exactly one parent and no children, meaning it is the last item at the end of each “branch.” Non-leaf nodes have two parents and can have any number of children.

*Figure 8*

**Merkle Tree**




Source: Kraken Intelligence

Merkle trees allow efficient and secure verification of the contents of large data structures. Merkle trees are used by major blockchains such as Bitcoin and Ethereum to allow for efficient and secure verification of the contents of large data structures, consistency verification, and data synchronization.[13,14] These major networks utilize Merkle trees instead of other data structures because it:

- Materially reduces the amount of data that a trusted authority has to maintain to prove the integrity of the data.
- Utilizes a small amount of network space, making proofs computationally seamless and fast.
- Requires a minuscule amount of information to be transmitted across a network during proofs as it separates the validation of the data from the data itself.

MAST requires users to reveal only the executed spending condition of a UTXO rather than the entire script, including conditions that weren't met. This solution minimizes the amount of opcode data revealed on-chain at creation or when a UTXO is spent, thus reducing the data load for each transaction and fees while preserving user privacy. Scripts are then stacked together in the MAST data structure for easy access. MAST is based on the realization that even the most complex smart contracts with numerous stakeholders could include a single condition for all participants to agree to settle the transaction.

MAST allows for all the different spending conditions of a UTXO set to be individually hashed—instead of combined into a single hash—and included in a Merkle tree. The Merkle tree is then used to produce a single hash known as the Merkle root, which "locks" UTXOs. If any data is revealed, the Merkle root and the Merkle path (some additional data) can be used to verify that the specific data was included in the Merkle tree while the rest of the conditions in the Merkle tree remain obscured from the public (except the condition that was satisfied to spend the bitcoin). MAST is more private and data-efficient than P2SH smart contracts. Moreover, Schnorr allows for these MAST structures to be indistinguishable from standard bitcoin transactions. As seen in figure 9, multisig

transactions eat up block space in proportion to how many signatories are involved. However, multisig transactions using P2TR will be fixed to allow for virtually any number of signatories while consuming the same amount of block space, costing and looking the same as a single-sig P2TR transaction.

*Figure 9*

**Size Comparison of Common Script Types**

| Epoch | Script Type | Input Size | Witness Size | Output Size |
|---|---|---|---|---|
| Legacy | P2PK | 113.5 B | N/A | 44 B |
| | P2PKH | 147.5 B | N/A | 34 B |
| | P2SH (2-of-2 multisig) | 259 B | N/A | 32 B |
| SegWit | P2SH (2-of-3 multisig) | 296 B | N/A | 32 B |
| | P2WPKH | 41 B | 107.5 B | 31 B |
| | P2WSH (1-of-1 multisig) | 41 B | 112.5 B | 43 B |
| | P2WSH (2-of-2 multisig) | 41 B | 219 B | 43 B |
| | P2WSH (2-of-3 multisig) | 41 B | 253 B | 43 B |
| Taproot | P2TR (key path) | 41 B | 66 B | 43 B |

Source: Coinmonks
Note: Size is recorded in Bytes, which refers to the raw byte length of a transaction's serialized format and is used to measure the data footprint of transactions on the network.

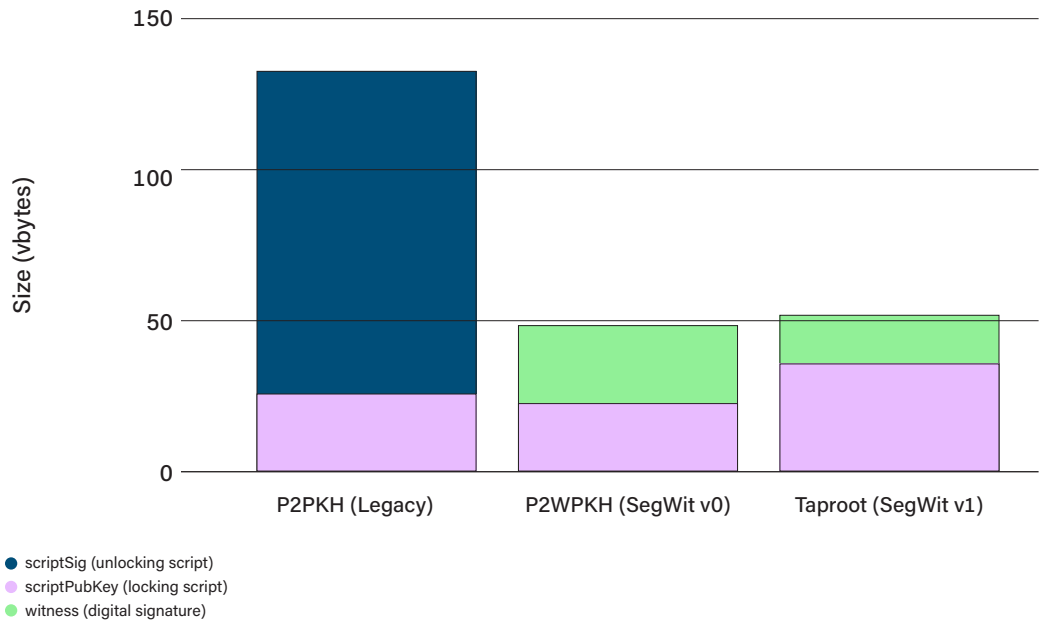
## Pay-to-Taproot (P2TR)

As previously mentioned, bip-taproot introduces a new script type known as Pay-to-Taproot (P2TR). This script type allows outputs to be spent using a single Schnorr signature to satisfy the keypath spending condition or by revealing and satisfying one of the scripts from the Merkle tree. A P2TR script merges the traditionally separate P2SH policies with P2PK, which is similar to P2PKH but differs in that it locks bitcoin to the public key instead of the hash of the public key; as a result, all outputs are spendable by either a public key or a script and indistinguishable from each other. P2TR is a type of native SegWit version 1 output. Bip-taproot specifically defines rules for v1 outputs with a 32-byte witness program, meaning other v1 output types could theoretically be introduced in the future.

To grasp bip-taproot's potential impact, one should look at the most common way wealth is transferred on the Bitcoin network: single-sig transactions. The cost to create and spend a Taproot single-sig output (SegWit v1) is about 5% more expensive than doing so with P2WPKH (SegWit v0). However, the cost to create a Taproot output is almost the same as to create a P2WSH output (multisig), and the cost to spend a single-key Taproot is 40% cheaper than P2WPKH. Though P2WPKH transactions are a bit more efficient than single-sig P2TR transactions, the latter allows for multisig transactions that look like single-sig transactions.

*Figure 10*

**Comparison of Single-Signature Transactions Using Legacy Software, SegWit, and Taproot**




Source: Bitcoin Optech

## BIP342 (bip-tapscript)

The third phase of the soft fork, known as Tapscript (BIP342), specifies a slight variant of Bitcoin's existing Script language used in P2TR transactions. As explained earlier, Bitcoin's legacy Script language is used to define the spending policies of coins. Similarly, Tapscript defines the semantics of Taproot's spending conditions to make Schnorr signatures and Taproot technology available to users.

As an upgraded version of Bitcoin's Script programming language, Tapscript enables a more seamless introduction of new features and builds on the batch verifiability of Schnorr signatures. Specifically, Tapscript improves signature hashing to validate taproot scripts, adds new opcodes to allow users to create more complex smart contracts, and changes some limits for resource requirements, including the removal of Bitcoin's Script 10,000-byte data size limit and the limit on the number of opcodes per transaction. Furthermore, Tapscript shares most opcodes with legacy and SegWit transactions with a couple of exceptions:

- TapScript's signature check opcode verifies Schnorr signatures rather than ECDSA signatures.
- Two multisig opcodes used to verify the validity of signatures in a multisig transaction are replaced by a single opcode that enables signature batch verification with Schnorr and reduces the number of opcodes needed to execute existing tasks, such as a signature check.
- A new opcode used to upgrade the Script system allows for a more seamless introduction of new opcodes in the future.
- Signature hashes (SIGHASH), which indicate what parts of the transaction is signed by the digital signature, are calculated differently than in legacy Script or SegWit to make it easier for future soft forks to extend the signature-checking opcodes with new sighash types or other changes.

Though Tapscript may seem like a minor boost to Schnorr and Taproot, it is an essential

piece of the puzzle. Signature verification is the most computationally intensive operation in a script because signatures are verified individually. By allowing for signature batch verification, Bitcoin can realize its efficiency improvements associated with Schnorr and Taproot. This newfound smart contract flexibility on Bitcoin, thanks to Taproot, permits more complicated multisig contracts and opens up the possibility of Bitcoin-based Decentralized Autonomous Organizations (DAOs) and other Decentralized Finance (DeFi) applications.

# 4.

# Taproot's Impact

As a soft fork, nodes running outdated software will still work. However, users who want to verify Taproot transactions have until approximately November 16th, 2021, to update their node software to Bitcoin Core v0.21.1—the newest version of Bitcoin Core containing activation logic for the Taproot soft fork. Because most wallet service providers will continue to use ECDSA even after Schnorr is activated due to its standardization, the power and capabilities of Taproot will likely not be realized immediately. Also, blocks will continue to have both Schnorr signatures and ECDSA signatures, limiting the benefits of batch verification in the short term. In due time, the broader crypto community will adopt the technology widely so that most transactions are conducted via Taproot, even without the Bitcoin user's knowledge of it. Nonetheless, complete adoption will not be realized overnight, and individual users are more likely to adopt Taproot before businesses.

With Taproot having been a work-in-progress for years, some market participants argue that the positive impact Taproot will have on bitcoin is already reflected in its price. These skeptics may point to bitcoin's near 10x increase since March 2020 as being partly attributed to the Taproot upgrade since it has been on the industry's radar for quite some time. Not to mention, given the complexity of the upgrade, some believe that the value Taproot brings to Bitcoin may go unnoticed by its users, if not take a substantial amount of time to be fully appreciated and reflected in overall network usage.

On the other hand, others will argue that the impact of Taproot will not be immediately reflected in price until shortly after its activation, as evidenced by bitcoin soaring +50% in the week following SegWit's activation. Additionally, these

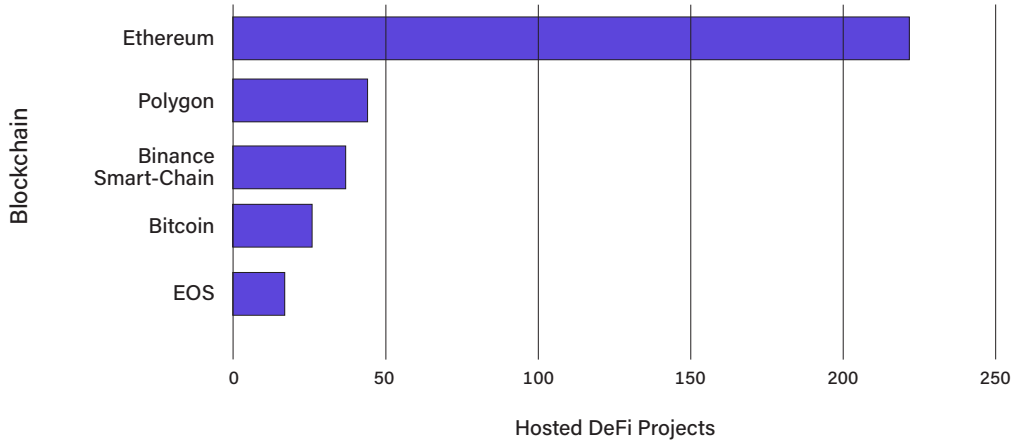
bulls will likely point to the potential mass adoption the network will see due to Taproot as a macro bullish indicator. In reality, it is most likely that other factors are the primary drivers of price action in the short term, with upgrades like Taproot not impacting price for years to come.

However, despite short-term price action, the technological upgrades make the Bitcoin network significantly more attractive for building DeFi protocols that could draw incremental demand for bitcoin. Specifically, key aggregation allows Bitcoin to compete with higher-throughput blockchains such as Ethereum, which is the primary breeding ground for DeFi projects. Users speculate that Taproot will ultimately enable DeFi networks (i.e., sidechains) on Bitcoin like Sovryn, Thorchain, and Portal, among others, to create massive multisig vaults (e.g., requiring over 1,000 signatures) that lock bitcoin up for usage in the sidechain while costing the same as a single-sig transaction. Before Taproot, operating these sidechains with this many signatories was prohibitively expensive.

Because large multisig smart contracts with hundreds or even thousands of signatories are too costly, Bitcoin has seen little interest from DeFi protocols. According to DeFi Prime, Ethereum hosts over 64% of DeFi projects, while Bitcoin is home to roughly 7.5% of DeFi protocols.[15]

*Figure 11*

**Number of DeFi Protocols by Blockchain**




Source: DeFi Prime

Furthermore, SegWit transactions and LN channels are positioned to become more prevalent now that a notably better option will be available. Still, following Taproot activation, this growth may be challenging to measure as the creation and closure of Taproot-compliant LN channels will look like standard transactions.

Lastly, Taproot has the potential to create a strong fee market for miners once the last bitcoin is mined around 2140. This theory assumes that users may want to passively engage in CoinJoin transactions where their wallet balances are combined between multiple senders into a single transaction if there is material demand for privacy. On-chain transaction fees could see a boost that may retain mining demand even after the last bitcoin is released if adoption arises. Either way, Taproot kills two birds with one stone by improving Bitcoin's viability as a store of value and a medium of exchange, both of which have been contended by critics due to Bitcoin's scalability concerns and highly volatile market price.

# 4.

# Conclusion

Taproot is set to be the biggest update to Bitcoin thanks to its focus on optimizing network scalability, transaction privacy, and smart contract functionality. As discussed, Schnorr signatures (BIP340) change Bitcoin's core cryptography to allow for key and signature aggregation, making multisig transactions look like regular transactions. Meanwhile, Taproot (BIP341) builds on Schnorr to create an upgraded SegWit transaction type with spending instructions based on Taproot, Schnorr signatures, and MAST; this means that the amount of spending condition information publicly revealed on-chain and allows for more complex spending conditions. Finally, we noted that Tapscript (BIP342) will enable an upgraded version of Bitcoin's Scripting language to integrate the new spending conditions made available by Taproot and make future network upgrades easier to conduct. Not only does Tapscript integrate new technologies into Bitcoin, but it also enhances the privacy and efficiency of existing features, such as SegWit and the LN. These features can help scale Bitcoin to a higher transaction throughput and attract a larger user base, but it could be that this potential will not be achieved without significant adoption. While some see Taproot as likely having little impact on the price of bitcoin because it has been widely expected for years and it may go unappreciated by most users following activation, others will say price will rise following activation because the upgrade will invite significantly more users to the network and Bitcoin's latest upgrade sparked a +50% price increase.

Bitcoin's mass adoption is debatably dependent upon the quality of its privacy and scaling guarantees, meaning the need for better privacy and scalability solutions will continue to grow in the long term. However, thanks to the technological upgrades enabled by the Taproot soft fork, Bitcoin will be a step closer to addressing these

overarching concerns. This upgrade is significant because, while many mature cryptographic privacy mechanisms exist and are ready for implementation, they often carry significant trade-offs that most stakeholders would disagree on. On the other hand, Taproot received a nearly perfect consensus as it seemingly carries no significant trade-offs. Looking ahead, Schnorr, Taproot, and TapScript will jointly empower users and pave the way for a new epoch for Bitcoin following activation in November 2021.

## *Endnotes*

1. Soft Fork - A software upgrade that is backwards-compatible with outdated versions of the software.
2. https://bitcoinops.org/en/topics/mast/
3. Opcode (Operation Code) - An instruction defined by a user in a bitcoin transaction.
4. https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2018-January/015614.html
5. https://github.com/bitcoin/bitcoin/pull/19997
6. https://github.com/ajtowns/taproot-review
7. Hard fork - A software upgrade that is backwards-incompatible with outdated versions of the software, requiring all nodes to upgrade to the latest version of the protocol software.
8. https://github.com/bitcoin/bips/blob/master/bip-0013.mediawiki
9. https://eprint.iacr.org/2013/635.pdf
10. https://www.di.ens.fr/david.pointcheval/Documents/Papers/2000_joc.pdf
11. https://en.wikipedia.org/wiki/Ralph_Merkle
12. https://www.google.com/patents/US4309569
13. https://bitcoin.org/en/developer-guide#block-chain-overview
14. https://blog.ethereum.org/2015/11/15/merkling-in-ethereum/
15. https://defiprime.com/bitcoin

We appreciate your feedback! Please visit **https://surveys.kraken.com/jfe/form/SV_2gJix9LD6BtAwTQ** to participate in a brief survey. For all future Kraken Intelligence content, sign up **here**. For comments, suggestions, or questions related to this article or future topics you'd like to learn more about, you may also direct your communication to intel@kraken.com or to your account manager.

Kraken provides access to more than 85 cryptocurrencies spanning nearly 390 markets with advanced trading features, industry leading security, and on-demand client service. With the acquisition of Crypto Facilities, Kraken now offers seamless access to regulated derivatives on 5 cryptocurrencies with up to 50x leverage. Sign up for a free account in minutes at **www.kraken.com/sign-up**. We look forward to welcoming you.

For multi-exchange charting, trading, portfolio tracking, and high resolution historical data, please visit **https://cryptowat.ch**. Create a free Cryptowatch account today at **https://cryptowat.ch/account/create**.

For OTC-related execution services or inquiries, please direct your communication to **otc@kraken.com** or to your account manager.

## *Disclaimer*