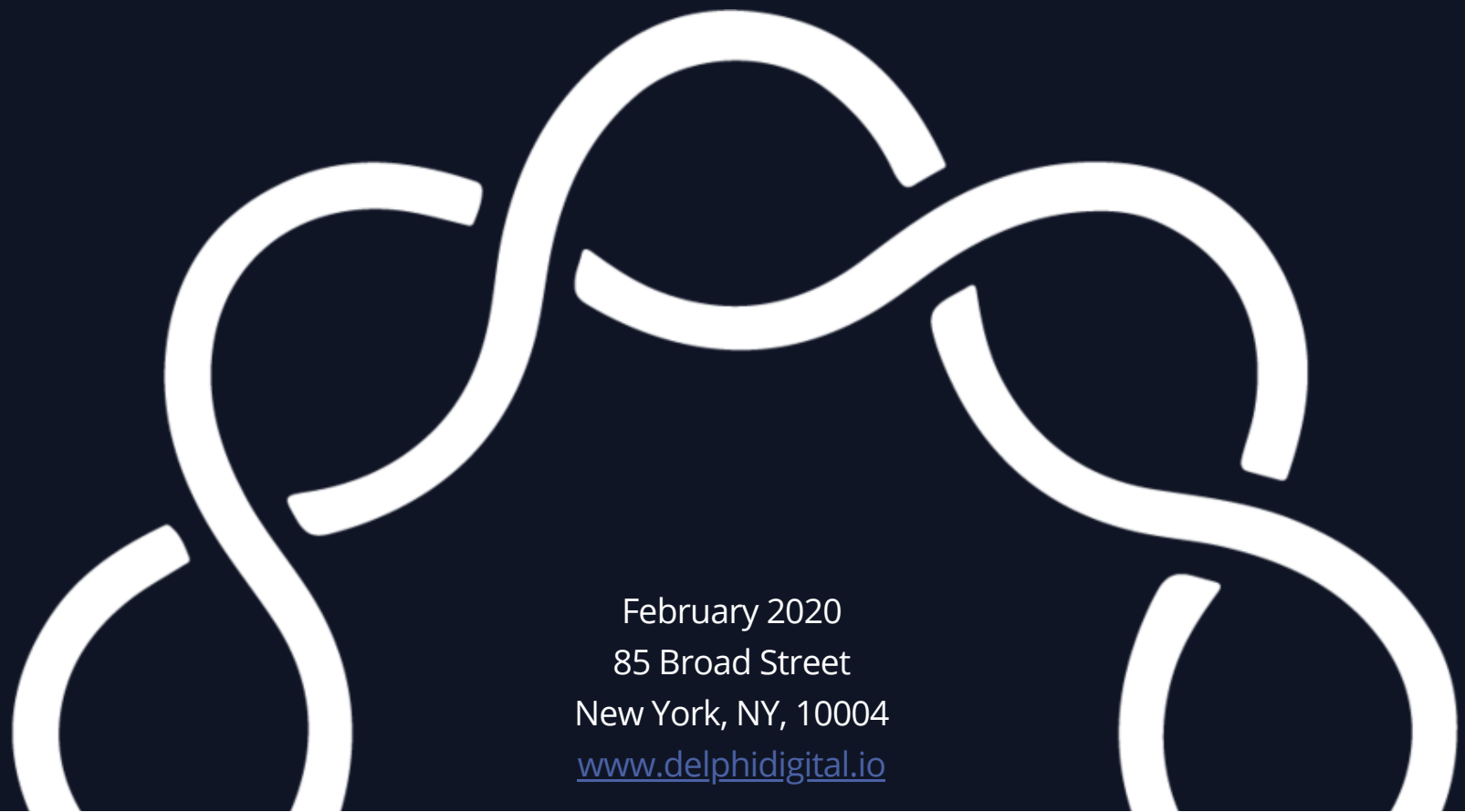
DELPHI DIGITAL

Digital Signature Schemes

Thematic Insights



February 2020
85 Broad Street
New York, NY, 10004
www.delphidigital.io

Table of Contents

Analysts

Medio Demarco
medio@delphidigital.io

Paul Burlage
paul@delphidigital.io

Preface

Digital signature schemes are **cryptographic primitives** (i.e. building blocks) that underpin the security of distributed networks. While the nuances of this topic can be difficult to grasp given how technically complex it is, learning the basics are crucial for understanding how these networks actually function.

Beyond even that, however, there have been exciting developments on this front worth paying attention to that will usher in significant changes and benefits such as enhanced privacy, security, functionality and scalability.

There are new projects deploying new schemes (i.e. THORChain and tBTC using Threshold ECDSA) and prominent projects set to finally adopt old, yet superior schemes (i.e. Bitcoin adding Schnorr). While there are many unique digital signature schemes in use today, we'll focus on three of the most important ones - **Threshold ECDSA, Schnorr and BLS.**

Rather than diving too deeply into the math and computer science involved, we'll provide a high-level overview of how this technology functions, why it matters, and touch on the projects either currently using these schemes or expected to in the future.

Threshold ECDSA

Schnorr

BLS

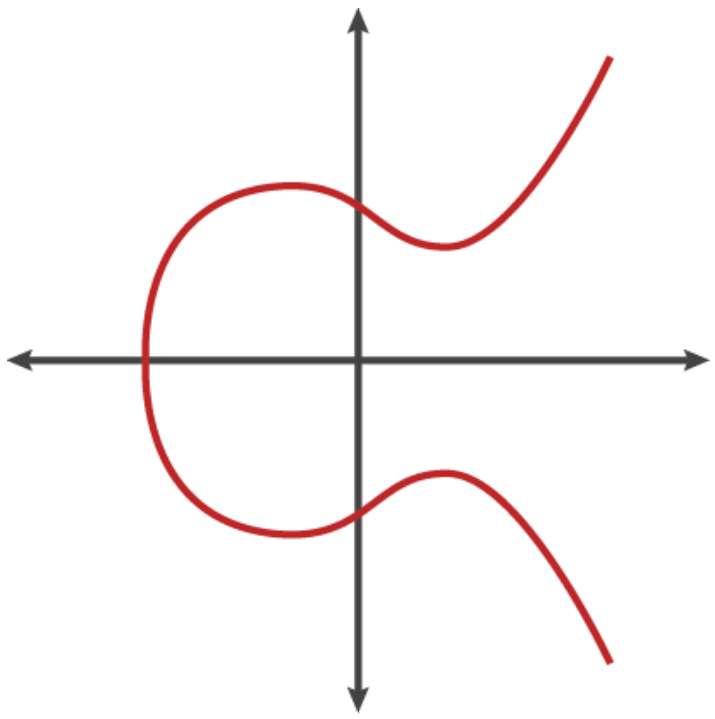
Elliptic Curve Cryptography

DELPHI DIGITAL

Before we dive into the different signature schemes, let's start with a quick overview on elliptic curve cryptography ("ECC"), since the vast majority of cryptocurrencies use some form of it as the basis of their security. In ECC, there are two components at work to be aware of - 1) the elliptic curve in use and 2) the signature scheme algorithms. For example, Bitcoin uses the secp256k1 curve and a compatible ECDSA algorithm. Elliptic curves represent parameters that allow people to securely arrive at a shared secret through a mathematical process. In other words, an elliptic curve is a set of points that answer a mathematical equation, such as the formula below.

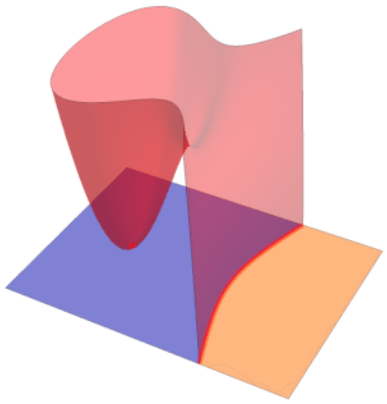
$$y^2 = x^3 + Ax + B$$

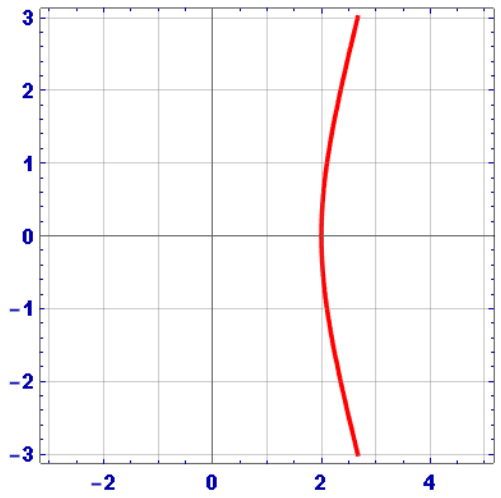
While these curves are often visualized in 2 dimensions, as seen in the picture to the right, this 2D image is actually just a slice of its 3 dimensional form, shown in Figure A. While this is close to an accurate visualization, the graph should actually wrap around itself. If we skip past some complex math and properly adjust the graph, you end up with the true visualization in Figure B. The colored dots in Figure B represent points on the elliptic curve. By plugging in the right input data, the algorithm will move around the points on the elliptic curve until it reaches a final point. For example, plugging in the private key will cause the algorithm to hop around the curve until a valid signature has been generated. We'll explain this in greater detail on the following slide.



To view the animated version click here

Figure A





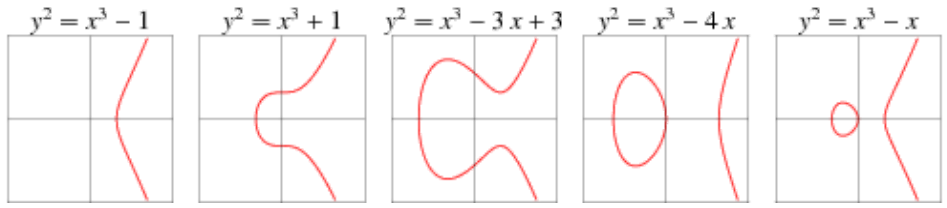
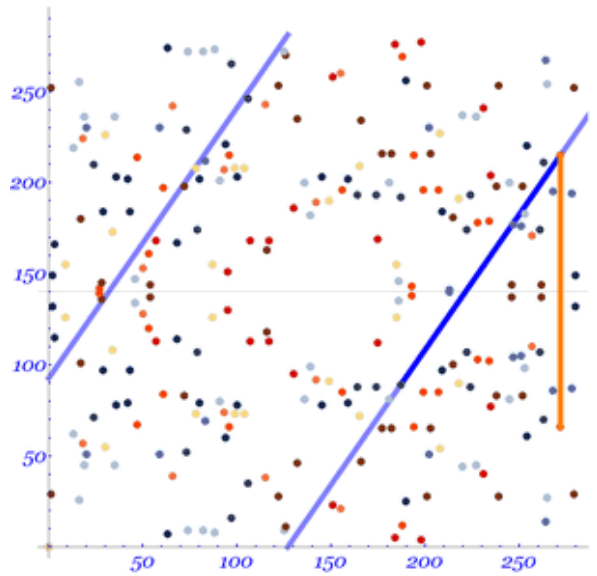
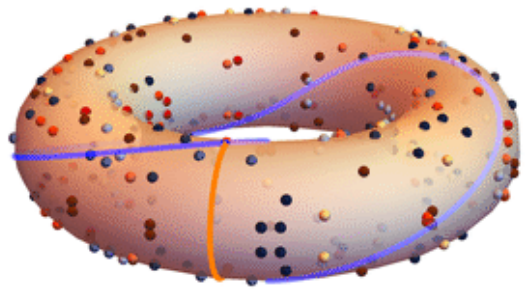


Figure B





Source: All About Circuits, Hans Knutson, Trustica, Cloudflare Blog, Wolfram MathWorld, Bitcoin StackExchange, An Introduction to Bitcoin, Elliptic Curves and the Mathematics of ECDSA, Guide to Elliptic Curve Cryptography

Public-Key Cryptography

Distributed networks leverage elliptic curves for their security through public-key cryptography (a.k.a asymmetric cryptography). These systems utilize both a public key and a private key. As the names imply, the public key can be openly shared while the private key is kept secret by the owner. In the context of cryptocurrencies, a user signs (sends) a transaction using their private key. Anyone can then prove that this transaction is valid by using the sender's public key. This is how cryptographic networks ensure the validity of transactions while preventing incorrect persons from spending coins they do not own. Ownership of a private key effectively acts as a form of identity on the network.

Conceptually, this might make sense but what exactly are these keys? A public key is simply a pair of X and Y coordinates for a point on the elliptic curve, while the private key is a piece of secret information necessary to generate those coordinates. You can use a private key to generate a public key but the process will not work in reverse. Let's walk through the visualizations to the right since they illustrate how this works in practice.

Alice wants to send a transaction to Bob so she plugs her private key into the signing algorithm. Remember, the private key is just a secret piece of information used as an input. After a few rounds of calculations moving around the curve, a valid signature is generated by the algorithm. To verify the signature, Bob plugs the information from the public key and signature into the verification algorithm. After a few rounds of calculations moving around the curve, Bob arrives at a secret point on the curve, confirming the validity of the signature. He wouldn't have been able to arrive at this point on the curve if the inputs had been incorrect. To watch the full detailed process on the 3D visualization you can click here.

While this is a simplified explanation of a complex mathematical process, understanding the factors at play here is important base knowledge which we'll build on throughout the rest of this report.

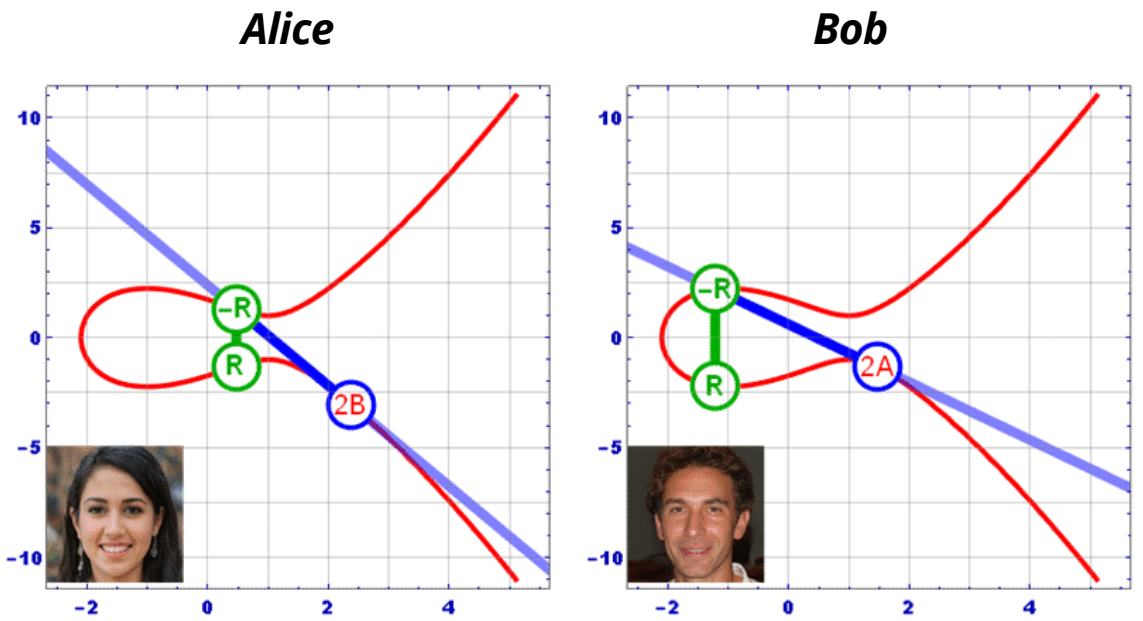
Source: Haseeb Qureshi, Trustica, Cloudflare Blog, All About Circuits, Learn Me A Bitcoin

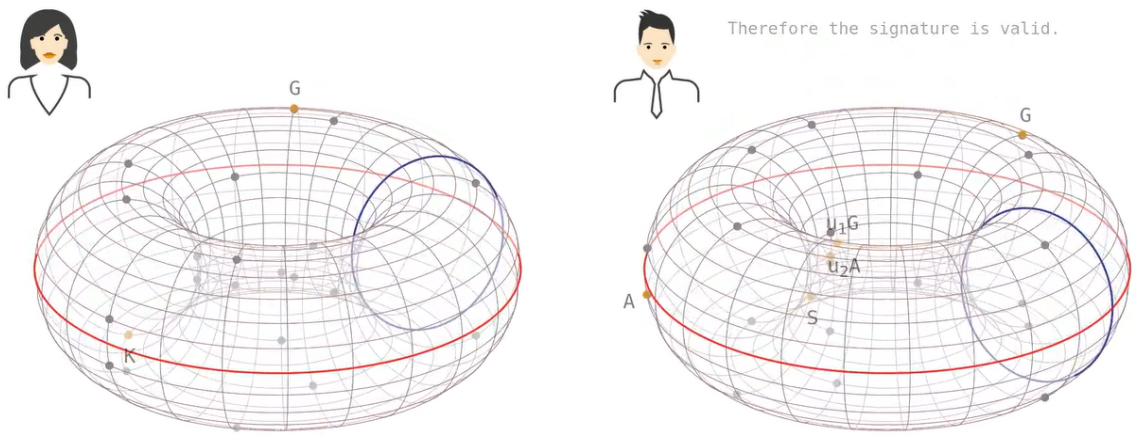
Digital Signature Schemes are generally comprised of 3 algorithms:

1) Key Generation Algorithm: Outputs a private key and its corresponding public key. Some schemes incorporate distributed key generation ("DKG") to ensure no single party ever has possession of the secret key.

2) Signing Algorithm: Outputs a signature from a private key and message input.

3) Signature Verifying Algorithm: Output validates or rejects a transaction based on the public key and signature.





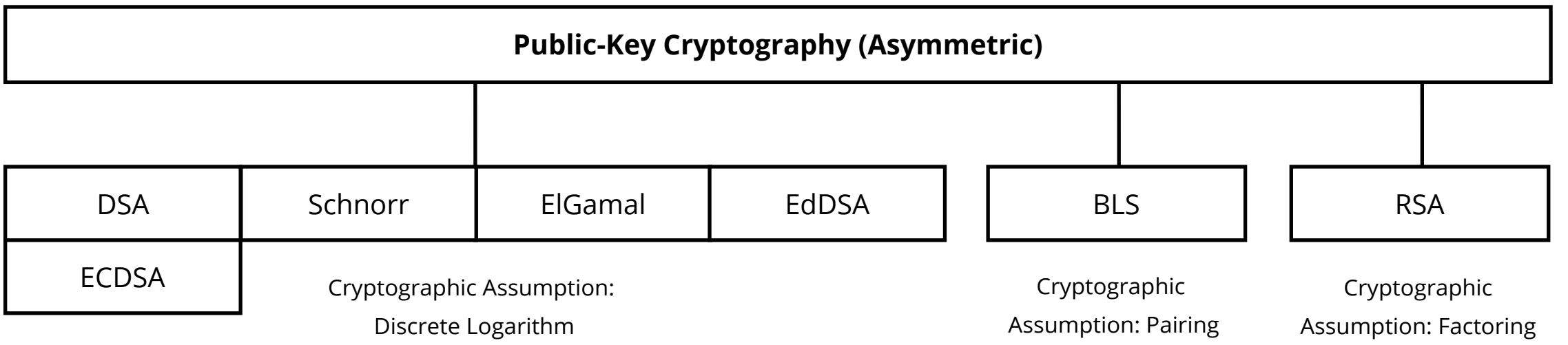
Digital Signature Algorithms Overview

DELPHI DIGITAL

A network can implement or be compatible with different signature schemes as long as they use the same elliptic curve. Bitcoin, for example, can already support Threshold ECDSA signatures and could add Schnorr and BLS through a softfork. The only time a hardfork would be required is if the new signature scheme would be allowed to spend existing BTC outputs secured by a different signature scheme. If you're interested in reading more on this topic, here's a great tweet thread by Pieter Wuille.

The diagram below is a small snapshot of the digital signature scheme landscape. As you can see, different schemes rely on different cryptographic assumptions. This refers to the assumption that it is difficult to compute certain mathematical processes. For example, Bitcoin and Ethereum currently use ECDSA signatures, whose security hinges upon the difficulty of computing discrete logs. ECDSA is standardized and can achieve the same relative security as RSA with smaller key sizes. How stark is the difference between the two approaches? By applying the concept of Universal Security, you can compute how much energy is needed to break a cryptographic algorithm, and compare that with how much water that energy could boil. This is a kind of cryptographic carbon footprint. By this measure, breaking a 228-bit RSA key requires less energy than it takes to boil a teaspoon of water. Comparatively, breaking a 228-bit elliptic curve key requires enough energy to boil all the water on earth. For this level of security with RSA, you'd need a key with 2,380-bits (source).

This is important as it impacts the on-chain data footprint and operational efficiency of the network. With that said, ECDSA does not come without its trade-offs. The verification equation is non-linear (unlike Schnorr and BLS) and, as a result, ECDSA multi-signature schemes have distinct disadvantages to alternative signature schemes. Methods such as Threshold ECDSA alleviate some deficiencies but bring new challenges of their own. We'll compare the characteristics of these different schemes on the following slide.



Source: Cloudflare Blog

Signature Scheme Properties

DELPHI DIGITAL

Properties		ThresholdECDSA	Schnorr	BLS
Preimage	The presence of a private key that needs to be split	No	No	No
Trusted Setup	Single party generating keys vs distributed key generation	No	No	No
Multisig Obfuscation	The transaction address does not reveal itself as multi-sig	Yes	Yes and No	Yes
Signer Privacy	Signers of a transaction can not be revealed from observation	Yes	Yes	Yes
Signature Size	How large is the signature size, and how does this change with an increase in m and n signers?	Static	Static	Static + Smaller
KeyGen Time	How much time it takes to generate keys and how does this change with an increase in m and n participants.	Dynamic	Dynamic	Dependent (n-n/ m-n)
KeyGen Rounds	The number of times participants must interact with one another to generate keys	Dynamic	Dynamic	For DKG and m-of-n
Verification Time	How much time it takes to verify a signature, and how does this change with an increase in m and n signers.	= ECSDA Speed	< ECSDA	>>> ECDSA
Signing Time	How much time it takes to sign a message, and how does this change with an increase in m and n.	Dependent	Dependent	Dependent
Signing Rounds	The necessity and number of times participants must interact with one another to sign a transaction	Yes (dependent)	Yes (dependent)	Aggregation only

This table is meant to illustrate the general tradeoffs/properties of the particular signature schemes. These signature schemes have numerous constructions, and therefore, not every property is 100% attributable.

Source: Leland Lee

Project List

The projects below either plan to implement, have discussed implementing or have contributed to the development of these signature schemes. This is a non-exhaustive list and many of the projects use different adaptations of the associated signatures.

Threshold ECDSA

OFFCHAIN LABS · KEEP · THORCHAIN · Ren · BINANCE · TORUS · ZenGo · SKALE

Schnorr

bitcoin · Blockstream · zilliqa · Bitcoin Cash · Grin · BEAM

BLS

chia · sigma prime · SKALE · celo · Algorand · Harmony · DFINITY · ethereum 2.0

Source 1, Source 2

MultiSig vs Signature Aggregation

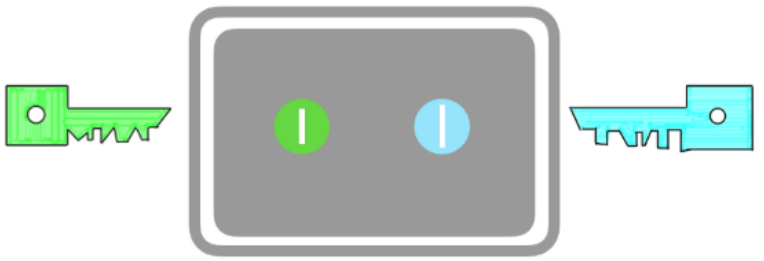
DELPHI DIGITAL

Threshold ECDSA, Schnorr and BLS offer a variety of benefits over current solutions, but they really shine in situations where multiples parties (M-of-N) must come together to sign a transaction (e.g. spend a BTC output). This functionality, often referred to as MultiSig, is already available but has some relative drawbacks.

With MultiSig, each of the parties has their own private key that corresponds to a different lock. For example, if there were 5 parties (N=5), there would also be 5 locks for each key. It may require at least 3 of the parties (M=3) to use their private key to sign an outgoing transaction. These private keys are static, stored locally by each party and the signing process happens entirely on-chain. It's also very easy to identify a MultiSig address on-chain and observe the signers. Using Bitcoin MultiSigs to illustrate this point, in the table to the bottom right, you can clearly see how much BTC is held in each type of MultiSig*. MultiSig transactions also have higher fees (more data per signature).

Compared to MultiSig, signature aggregation schemes offer benefits such as improved privacy and lower transaction fees. In addition, while MultiSig designs are unique to each chain, schemes such as Threshold ECDSA are chain-agnostic, working with any network that supports ECDSA signatures (almost all of them). This is one of the reason's why THORChain is leveraging Threshold ECDSA for its cross-chain liquidity pools. A full report on THORChain's technical and economic design is now available to our Institutional members. Let's now dive into each scheme in greater detail.

MultiSig Vault



Threshold ECDSA Vault

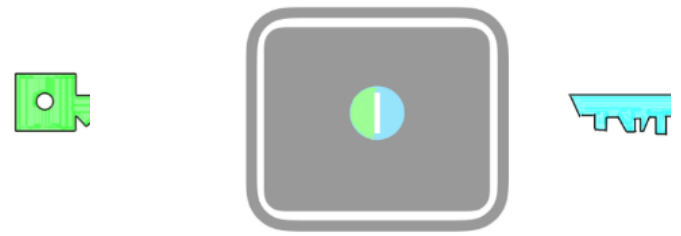


Image modified; Source: ZenGo

MultiSig Tradeoffs:

Strengths	Drawbacks
• Additional signers can improve security	• Multiple private keys exist and are stored locally • Higher transaction fees (more data per signature) • Reduced privacy since addresses, transactions and signers are easily identified on-chain • Implementations are chain specific • Less flexible than other schemes

BTC in P2SH Addresses by Type

Type of P2SH Address	Amount of BTC	Value of BTC
UNSPENT	3,527,000	$26047 M
P2WPKH	1,018,000	$7518 M
2 OF 3	314,000	$2319 M
OTHER MULTISIGS	290,000	$2142 M
3 OF 4	264,000	$1950 M
3 OF 6	77,000	$569 M
3 OF 5	72,000	$532 M
2 OF 6	69,000	$510 M
sw 2 OF 3	63,000	$465 M
2 OF 2	58,000	$428 M
2 OF 4	16,000	$118 M
OTHER NON-MULTI	10,000	$74 M
3 OF 3	852	$6 M
TOTAL	**5,778,852**	**$42677 M**

Data as of: December 29th, 2019
Source: txstats.com,

**The UNSPENT category represents MultiSigs that haven't spent an output yet, which would reveal what type (M-of-N) of lock it is*

Threshold ECDSA

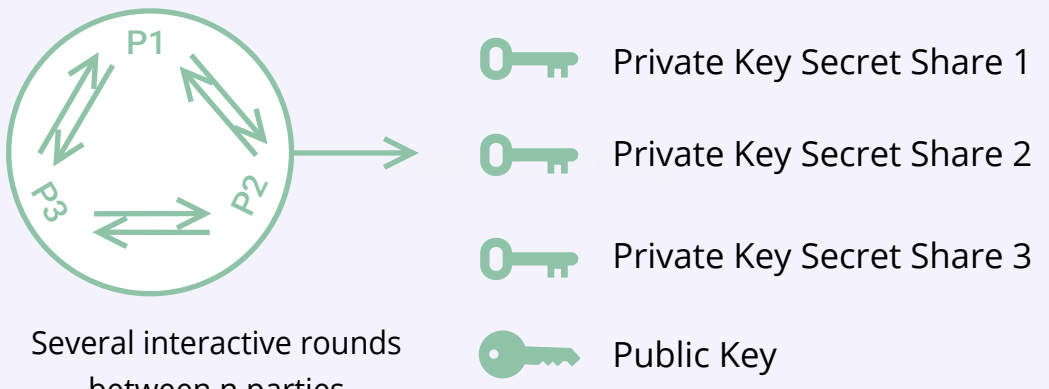
Overview

From a high level, threshold signature schemes alter the key generation and signing algorithms of classical digital signature schemes. This is done by introducing interactive multi-party computation, which in turn gives the scheme distributed key generation and distributed signing functionality.

During the key generation step, a set of n parties generate a set of distinct private key secret shares to each party. The secret is said to be t-of-n, ensuring that a threshold of t +1 parties are needed to reveal any private key information. In TSS, the key generation abstracts local key generation (no preimage, trusted dealer, and or prior existence of private key) and generates the said key through several interactive rounds between multiple parties. The output is multiple private key shares for each party who participated in DKG and one public key.

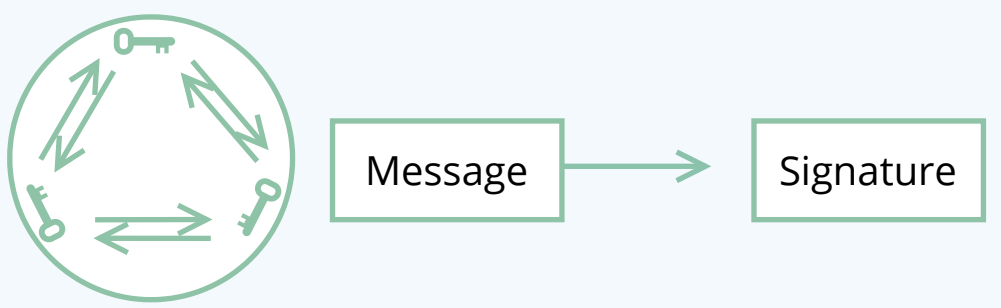
For transaction signing, again, these parties participate in several interactive rounds, but this time to generate a signature. The threshold institutes that there must be t parties that agree on a transaction to sign and output a signature. The verify algorithm remains the same, as still, only one private key exists. Threshold ECDSA is illustrated below.

Distributed Key Generation (DKG)



Several interactive rounds between n parties

Distributed Signing



Several interactive rounds between at least t + 1 parties combined with message

Features

TSS can be applied to various schemes (and therefore is protocol-agnostic) with interactive rounds/computation happening off-chain. Interactive rounds taking place off-chain, in combination with signature aggregation, results in lower transaction fees than classic MultiSig transactions (verifiers do not have to check every signature corresponds to every public key). Due to the above point, TSS signatures are more efficient than MultiSig. A TSS signature output resembles that of regular digital signature, and thus, retains the privacy of all parties involved (obfuscates the presence of multiple signers). Threshold signatures have different security assumptions, as funds/transactions are only secure up to the threshold. While no one party can forge the signature, one still trusts that a threshold level of the parties involved is not malicious. Interactive rounds also add complexity, which brings additional challenges (i.e. signers need to be online). This is further explained on the Schnorr Signature page.

Source

Schnorr Signatures

DELPHI DIGITAL

Overview

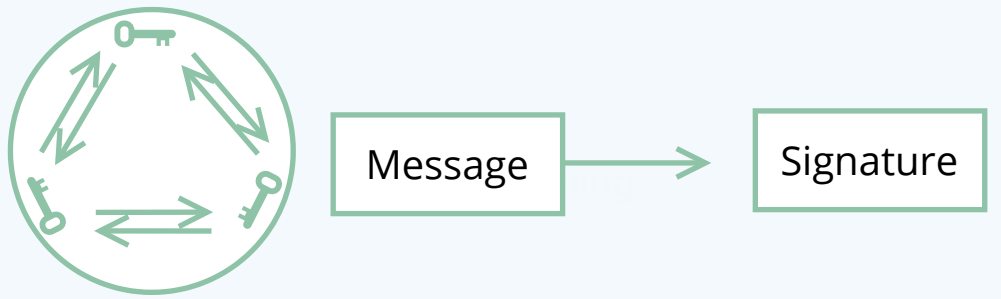
A Schnorr signature is a type of digital signature that relies on the discrete log cryptographic assumption, and therefore can work with various mathematical groups that satisfy this problem (e.g. Bitcoin's sep256k1 elliptic curve). The Schnorr Signature algorithm allows for the aggregation of multiple signatures while retaining a constant signature size. This is unlike P2SH MultiSig (Bitcoin), which grows linearly with the number of private keys (think m, of m-of-n). Schnorr signatures have one signature opposed to a m number of signatures for every m party, and thus they significantly reduce the size of signature data of on-chain transactions.

In the early iterations of Schnorr signatures, each party had distinct private and public keys. Consequently, on-chain proofs were required for each participant to validate the authenticity of their Public Key. There have been numerous iterations of Schnorr signatures that expand on its functionality. For example, MuSig allows for both digital signature and public key aggregation. This allows for only one public key needed for verification. There are many different approaches to m-of-n Schnorr signatures, in addition to MuSig, with various associated trade-offs. These tradeoffs include: foregoing key aggregation, key aggregation with a trusted setup, using Merkle leaves as permitted combinations of keys, the use of a DKG ceremony, etc.

Public Key Aggregation

Multiple private keys output a singular public key

Multi-sig Signing



Dependent on the Schnorr signature implementation (MuSig =3), there is x interactive rounds between at least m parties to sign the message

Features

Compared to ECDSA signatures, Schnorr signatures are smaller and allow for both signature and key aggregation. In return, participant keys and the count of keys are obfuscated, and thus verifiers do not need to know the individual private keys for every signer. The smaller signature and a single key predicate lead to less computational verification costs. Together, Schnorr signatures increase privacy (spending policy privacy, not transaction linkage privacy) and scalability capabilities (batch validation). The verifier does not need to verify every signature separately like in ECDSA - making Schnorr signature highly efficient.

Schnorr signatures incorporating key aggregation, however, does bring some complications. To aggregate signatures, parties need to agree upon a common random number. As a result, the parties must take part in multiple communication rounds dependent on m. Additionally, m-of-n Schnorr requires making a Merkle tree of m public keys. This makes Schnorr signatures scaling benefits not particularly as useful for large sets of m or n. Additional issues arise as interactive rounds are difficult when many keys are in cold storage. While ECSDA also requires a random number (nonce), generating this number via Schnorr is not deterministic. There needs to be a source for this randomness, and this results in some additional attack vectors.

Source

Schnorr on Bitcoin

DELPHI DIGITAL

Bitcoin Core discussions about Schnorr signatures date back to 2014, but only recently has the Taproot/Schnorr soft-fork Bitcoin Improvement Proposal (BIP) officially been published (BIP 340, 341, 342). The BIPs present a standard for 64-byte Schnorr signatures over the elliptic curve secp256k1. Bitcoin, as it stands, uses ECDSA signatures over this curve. BIP 340 outlines the additional traits of Schnorr signatures and its resulting benefits for authenticating transactions.

To summarize, Schnorr signatures are provably secure, and for the same relative security, ECDSA relies on far stronger assumptions. Schnorr signatures, also, are non-malleable, unlike ECDSA. Lastly, the linearity of Schnorr signatures allows for simpler and more efficient multi-party signing (adding up multiple keys to output a single public key that is valid for one signature).

From these qualities, the BIP lists the associated benefits of Schnorr signatures:

1. Signature Encoding: Schnorr allows for a fixed 64-byte signature encoding format (ECDSA requires a variable format that can reach up to 72 bytes).

2. Public Key Encoding: Schnorr public keys are encoded as 32 bytes (ECDSA uses compressed 33-byte encoding of elliptic curve points).

3. Batch Verification: Schnorr signatures allow efficiently verifying signatures in batch, unlike pure ECDSA.

4. Completely Specified: the verification algorithm is completely specified at the byte level, unlike ECDSA. This ensures that all signatures are valid to all verifiers, not just a portion of verifiers.

Notable High-level Effects:

Schnorr improves privacy, however, it is not a privacy solution. Schnorr signatures do not obfuscate transaction linkage, although, they do increase the anonymity set by making multi-signature transactions look like single-signature transactions. This makes the on-chain footprint of various use-cases more obscure. With all anonymity set privacy methods, adoption of the practices (taproot/MuSig) by a wide user base are imperative for significant privacy. Schnorr signatures, due to their more simple construction, ultimately make privacy features more accessible and implementable than ECDSA.

BLS Signatures

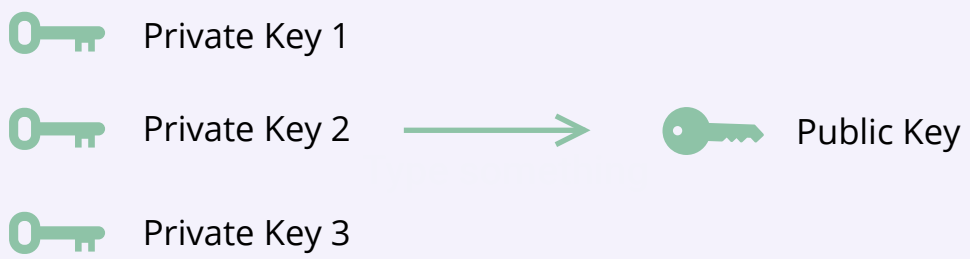
Overview

BLS signature schemes rely on two cryptographic primitives: hashing to the curve and curves pairing. In ECDSA and Schnorr, a message is first hashed and then this output number is signed. Hashing to the curve, on the other hand, involves hashing the message directly to the elliptic curve. Curves pairing is quite complex, and for more information, this is a good resource. These primitives resolve issues of other signature schemes, but come with their drawbacks.

As noted previously, ECDSA makes it difficult to aggregate keys and signatures. BLS, along with Schnorr, improves on this aspect via key aggregation and signature compression. Due to BLS's underlying construction, it fulfills the concept of an 'aggregated signature scheme' - n parties can sign a **different** message, and these n signatures can be combined into a singular signature. This is unique to BLS. Schnorr only allows for aggregation of signatures of the same message and requires this to happen at the time of signing via several private interactive rounds. BLS, on the other hand, allows for miners/users to publicly aggregate signatures of different messages at any time.

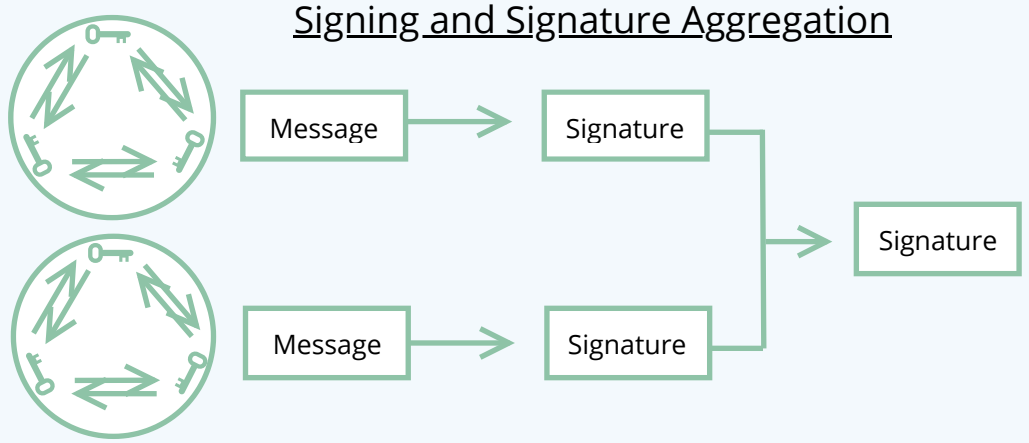
For m-of-n BLS, a single round communication set-up period is required to generate membership keys for every private key. M-of-n BLS can utilize membership keys or rely on other methods, such as Shamir's Secret Sharing. BLS signature length is 2x shorter than Schnorr and ECDSA. All the signatures in the block when compressed can take up to only 33 bytes. Unlike Schnorr, BLS doesn't rely on any randomness. It is an entirely deterministic signature algorithm.

Public Key Aggregation



Multiple private keys output a singular public key. For n-of-n, no interactive rounds required. For m-of-n, an interactive setup period is required to create 'membership keys'.

Signing and Signature Aggregation



Signature aggregation from different messages

Features

Due to BLS's unique cryptographic assumptions, it has differentiated traits to that of Schnorr and ECSDA. For one, BLS signatures are significantly shorter and do not require a random number when combining. M-of-n BLS, additionally, does not require several interactive rounds of communication. **Lastly, BLS allows for all signatures in a single block to be combined.** This is not the case for Schnorr. Bring this all together and you have a signature scheme that is more efficient for large signature sets. The relative time difference in key generation and signing between an m-100 vs m-1000 multi-signature scheme are impressively small. With that said, BLS comes with a significant drawback. Verification times (CPU verification computation for Bitcoin is the main bottleneck) is slower than that of ECDSA and Schnorr. This is due to the inefficiencies of pairing. As a result, BLS can significantly reduce block size by including more transactions, but would still necessitate long verification times. The security of pairing is not as battle-tested as that of discrete logs.

Source

Disclosures

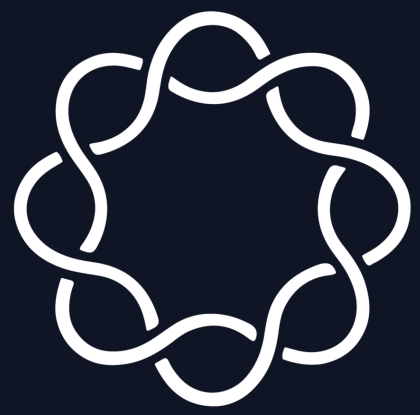
The Research Team may own the tokens represented in this report, and as such this should be seen as a disclosure of any potential conflict of interest. Anyone can contact Delphi Digital for full token disclosures by team member at Team@DelphiDigital.io. This report belongs to Delphi Digital, and represents the opinions of the Research Team.

Delphi Digital is not a FINRA registered broker-dealer or investment adviser and does not provide investment banking services. This report is not investment advice, it is strictly informational. Do not trade or invest in any tokens, companies or entities based solely upon this information. Any investment involves substantial risks, including, but not limited to, pricing volatility, inadequate liquidity, and the potential complete loss of principal. Investors should conduct independent due diligence, with assistance from professional financial, legal and tax experts, on topics discussed in this document and develop a stand-alone judgment of the relevant markets prior to making any investment decision.

Delphi Digital does not receive compensation from the companies, entities, or protocols they write about. The only fees Delphi Digital earns is through paying subscribers. Compensation is not received on any basis contingent upon communicating a positive opinion in this report. The authors were not hired by the covered entity to prepare this report. Delphi Digital did not receive compensation from the entities covered in this report for non-report services, such as presenting at author sponsored investor conferences, distributing press releases or other ancillary services. The entities covered in this report have not previously paid the author in cash or in stock for any research reports or other services. The covered entities in this report are not required to engage with Delphi Digital.

The Research Team has obtained all information herein from sources they believe to be accurate and reliable. However, such information is presented “as is,” without warranty of any kind – whether expressed or implied. All market prices, data and other information are not warranted as to completeness or accuracy, are based upon selected public market data, reflect prevailing conditions, and the Research Team’s views as of this date, all of which are accordingly subject to change without notice. Delphi Digital has no obligation to continue offering reports regarding this topic. Reports are prepared as of the date(s) indicated and may become unreliable because of subsequent market or economic circumstances. The graphs, charts and other visual aids are provided for informational purposes only. None of these graphs, charts or visual aids can and of themselves be used to make investment decisions. No representation is made that these will assist any person in making investment decisions and no graph, chart or other visual aid can capture all factors and variables required in making such decisions.

The information contained in this document may include, or incorporate by reference, forward-looking statements, which would include any statements that are not statements of historical fact. No representations or warranties are made as to the accuracy of such forward-looking statements. Any projections, forecasts and estimates contained in this document are necessarily speculative in nature and are based upon certain assumptions. These forward-looking statements may turn out to be wrong and can be affected by inaccurate assumptions or by known or unknown risks, uncertainties and other factors, most of which are beyond control. It can be expected that some or all of such forward-looking assumptions will not materialize or will vary significantly from actual results.



DELPHI DIGITAL

85 Broad Street
New York, NY, 10004
www.delphidigital.io