Cryptography behind top 20 cryptocurrencies

Items marked with * have some additional notes, which you can display by hovering on them.

Name	Type	Signing alg	Curve	Hash	Address encoding	Addr has
Bitcoin	UTXO	ECDSA	secp256k1	SHA-256	base58, bech32	SHA-256, RIPEM 160
Ethereum	account	ECDSA	secp256k1	Keccak-256 *	none (just hex) *	last 2(of Kecca 256 *
XRP	account	ECDSA *	secp256k1 *	first half of SHA-512	base58 with different alphabet *	SHA-256, RIPEM 160
Litecoin	UTXO	ECDSA	secp256k1	SHA-256 *	base58, bech32	SHA-256, RIPEM 160
EOS	account	ECDSA	secp256k1	SHA-256	none *	none
Bitcoin Cash	Same as Bitcoin *					
Stellar	account	EdDSA	ed25519	SHA-256 and SHA-512 in EdDSA *	base32	none

Binance Coin	Ethereum ERC-20 token *					
Tether	Bitcoin Omni layer / Ethereum ERC-20 token					
TRON	UTXO	ECDSA	secp256k1	SHA-256	base58	last 2(bytes Kecca 256 *
Cardano	UTXO	EdDSA	ed25519	none and SHA-512 in EdDSA *	base58	none
Monero	UTXO *	*it's complicated**	ed25519	Keccak-256 *	base58	Kecca 256 *
IOTA	UTXO	Winternitz one time signature scheme	-	Curl, Kerl *	none	Kerl
Dash	UTXO	ECDSA	secp256k1	SHA-256 *	base58	SHA-256, RIPEM 160
Maker	Ethereum ERC-20 token					
NEO	account	ECDSA	secp256r1	SHA-256	base58	SHA-256, RIPEM 160
Ontology	account	ECDSA	nist256p1	3x SHA-256	base58	SHA-256, RIPEM 160
Ethereum Classic	Same as Ethereum					
NEM	account	EdDSA	ed25519	none and Keccak-256 in	base32	Kecca 256, RIPEM

				EdDSA *		160
Zcash	UTXO	ECDSA, zk-SNARKs *	secp256k1, Jubjub *	SHA-256	base58, bech32	SHA-256, RIPEM 160
Tezos	account	EdDSA, ECDSA *	ed25519, secp256k1, secp256r1	BLAKE2 and SHA-512 in EdDSA *	base58	BLAK

Notes

Items marked with * have some additional notes, which you can display just by hovering on them.

- [1]: The Keccak hash function has won the SHA-3 competition and is thus the underlying hash function in SHA-3. However, some modifications had been introduced in the final stage of standardization, which lead to two *very similar but yet different* hash functions with completely different outputs. Keccak: as introduced by its authors; SHA-3: as standardized in FIPS-202. Many cryptocurrencies (such as Ethereum) chose SHA-3 for their hashing algorithm, but did so before the standard was finalised. This resulted in them using the Keccak function instead of what is now considered as SHA-3.
  Source: Wikipedia, Ethereum SE, Ethereum yellow paper
- [2]: EdDSA hashes the message internally before signing and this function can be chosen. The default is SHA-512. Some cryptocurrencies are leaving this up to the EdDSA algorithm exclusively and do not hash the message beforehand. Others do. This summary tries to list both, the "outter" hash that is used before inserting it into the algorithm and also the EdDSA's internal one.
  Source: RFC 8032

- [3]: Monero transactions are based on Elliptic Curve Cryptography using curve Ed25519, but transaction inputs are signed with so-called Multilayered Linkable Spontaneous Anonymous Group signatures (MLSAG), and output amounts (communicated to recipients via ECDH) are concealed with Pedersen commitments and Borromean ring signatures (later replaced with Bulletproofs).
  Source: From Zero To Monero, Ring confidential transactions, Confidential transactions, Borromean ring signatures, Bulletproofs
- [4]: Vulnerabilities were found in the Curl hash function, causing several dispute between IOTA engineers and some cryptographers.
  Source: Cryptographic vulnerabilities in IOTA, IOTA Vulnerability Report

Columns description

- **Type:** Whether the cryptocurrency is based on the UTXO model or not. Transactions in the UTXO model always spend the whole input, with some of the coins returned as a change. In account-based model, there is no such mechanism and just a simple account book is used.
- **Signing algorithm:** What signing algorithm is being used, all present cryptocurrencies use Elliptic Curve Cryptography.
- **Curve:** What elliptic curve is being used in the underlying signing algorithm.
- **Hash:** What hash function is being used to hash the transaction data that are then signed.
- **Address encoding:** What algorithm is being used to encode the address.
- **Address hash:** What hash function is being used for addresses.

Credits

- Tomas Susanka, @tsusanka

- Found a bug? File a PR on GitHub
- Like it? Tip it using Lightning Network:

tippin me