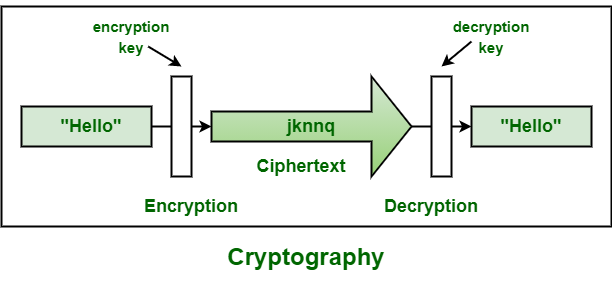
**Cryptography** **i**s a technique of securing communication by converting plain text into unintelligible ciphertext. It involves various algorithms and protocols to ensure data confidentiality, integrity, authentication, and non-repudiation.



The two primary types of cryptography are symmetric key cryptography and asymmetric key cryptography.

**Elliptic Curve Cryptography**

**Elliptic-curve cryptography** (**ECC**) is an approach to [public-key cryptography](https://en.wikipedia.org/wiki/Public-key_cryptography) based on the [algebraic structure](https://en.wikipedia.org/wiki/Algebraic_structure) of [elliptic curves](https://en.wikipedia.org/wiki/Elliptic_curve) over [finite fields](https://en.wikipedia.org/wiki/Finite_field). ECC allows smaller keys compared to non-EC cryptography (based on plain [Galois fields](https://en.wikipedia.org/wiki/Finite_field))[*[vague](https://en.wikipedia.org/wiki/Wikipedia:Vagueness" \o "Wikipedia:Vagueness)*] to provide equivalent security.[[1]](https://en.wikipedia.org/wiki/Elliptic-curve_cryptography#cite_note-:0-1)

Elliptic curves are applicable for [key agreement](https://en.wikipedia.org/wiki/Key_agreement), [digital signatures](https://en.wikipedia.org/wiki/Digital_signature), [pseudo-random generators](https://en.wikipedia.org/wiki/Cryptographically_secure_pseudorandom_number_generator) and other tasks. Indirectly, they can be used for [encryption](https://en.wikipedia.org/wiki/Encryption) by combining the key agreement with a [symmetric encryption](https://en.wikipedia.org/wiki/Symmetric-key_algorithm) scheme. They are also used in several [integer factorization](https://en.wikipedia.org/wiki/Integer_factorization) [algorithms](https://en.wikipedia.org/wiki/Algorithm) that have applications in cryptography, such as [Lenstra elliptic-curve factorization](https://en.wikipedia.org/wiki/Lenstra_elliptic-curve_factorization).

**ECC** implements all major capabilities of the asymmetric cryptosystems: **encryption**, **signatures** and **key exchange**.

The **ECC cryptography** is considered a natural modern **successor of the RSA** cryptosystem, because ECC uses **smaller keys** and signatures than RSA for the same level of security and provides very **fast key generation**, **fast key agreement** and **fast signatures**.

ECC Keys:

The **private keys** in the ECC are integers (in the range of the curve's field size, typically **256-bit** integers). Example of 256-bit ECC private key (hex encoded, 32 bytes, 64 hex digits) is:

0x51897b64e85c3f714bba707e867914295a1377a7463a9dae8ea6a8b914246319.

The **key generation** in the ECC cryptography is as simple as securely generating a **random integer** in certain range, so it is extremely fast. Any number within the range is valid ECC private key.

The **public keys** in the ECC are **EC points** - pairs of integer coordinates {***x***, ***y***}, laying on the curve. Due to their special properties, **EC points** can be **compressed** to just one coordinate + 1 bit (odd or even). Thus the **compressed public key**, corresponding to a 256-bit ECC private key, is a **257-bit** integer. Example of ECC public key (corresponding to the above private key, encoded in the Ethereum format, as hex with prefix 02 or 03) is: 0x02f54ba86dc1ccb5bed0224d23f01ed87e4a443c47fc690d7797a13d41d2340e1a. In this format the public key actually takes 33 bytes (66 hex digits), which can be optimized to exactly 257 bits.

## Curves and Key Length

**ECC** crypto algorithms can use different underlying **elliptic curves**. Different curves provide different level of **security** (cryptographic strength), different **performance** (speed) and different **key length**, and also may involve different algorithms.

**ECC curves**, adopted in the popular cryptographic libraries and security standards, have **name** (named curves, e.g. secp256k1 or Curve25519), **field size** (which defines the key length, e.g. **256-bit**), security **strength** (usually the field size / 2 or less), **performance** (operations/sec) and many other parameters.

**ECC keys** have **length**, which directly depends on the underlying curve. In most applications (like OpenSSL, OpenSSH and Bitcoin) the default **key length** for the ECC private keys is **256 bits**, but depending on the curve many different ECC key sizes are possible: 192-bit (curve secp192r1), 233-bit (curve sect233k1), 224-bit (curve secp224k1), 256-bit (curves secp256k1 and Curve25519), 283-bit (curve sect283k1), 384-bit (curves p384 and secp384r1), 409-bit (curve sect409r1), 414-bit (curve Curve41417), 448-bit (curve Curve448-Goldilocks), 511-bit (curve M-511), 521-bit (curve P-521), 571-bit (curve sect571k1) and many others.

## ECC Algorithms

**Elliptic-curve cryptography** (ECC) provides several groups of algorithms, based on the math of the elliptic curves over finite fields:

* ECC **digital signature** algorithms like [**ECDSA**](https://en.wikipedia.org/wiki/Elliptic_Curve_Digital_Signature_Algorithm) (for classical curves) and [**EdDSA**](https://en.wikipedia.org/wiki/EdDSA) (for twisted Edwards curves).
* ECC **encryption** algorithms and hybrid encryption schemes like the [**ECIES**](https://en.wikipedia.org/wiki/Integrated_Encryption_Scheme) integrated encryption scheme and [**EEECC**](https://cse.unl.edu/~ssamal/crypto/EEECC.pdf) (EC-based ElGamal).
* ECC **key agreement** algorithms like [**ECDH**](https://en.wikipedia.org/wiki/Elliptic-curve_Diffie%E2%80%93Hellman), [**X25519**](https://cryptography.io/en/latest/hazmat/primitives/asymmetric/x25519/) and [**FHMQV**](https://fastd.readthedocs.io/en/v18/crypto/fhmqvc.html).

All these algorithms use a **curve** behind (like secp256k1, curve25519 or p521) for the calculations and rely of the difficulty of the **ECDLP** (elliptic curve discrete logarithm problem). All these algorithms use public / private key pairs, where the **private key** is an integer and the **public key** is a point on the elliptic curve (EC point). Let's get into details about the elliptic curves over finite fields.

## ECC Algorithms

**Elliptic-curve cryptography** (ECC) provides several groups of algorithms, based on the math of the elliptic curves over finite fields:

ECC **digital signature** algorithms like [**ECDSA**](https://en.wikipedia.org/wiki/Elliptic_Curve_Digital_Signature_Algorithm) (for classical curves) and [**EdDSA**](https://en.wikipedia.org/wiki/EdDSA) (for twisted Edwards curves).

 ECC **encryption** algorithms and hybrid encryption schemes like the [**ECIES**](https://en.wikipedia.org/wiki/Integrated_Encryption_Scheme) integrated encryption scheme and [**EEECC**](https://cse.unl.edu/~ssamal/crypto/EEECC.pdf) (EC-based ElGamal).

 ECC **key agreement** algorithms like [**ECDH**](https://en.wikipedia.org/wiki/Elliptic-curve_Diffie%E2%80%93Hellman), [**X25519**](https://cryptography.io/en/latest/hazmat/primitives/asymmetric/x25519/) and [**FHMQV**](https://fastd.readthedocs.io/en/v18/crypto/fhmqvc.html).

All these algorithms use a **curve** behind (like secp256k1, curve25519 or p521) for the calculations and rely of the difficulty of the **ECDLP** (elliptic curve discrete logarithm problem). All these algorithms use public / private key pairs, where the **private key** is an integer and the **public key** is a point on the elliptic curve (EC point). Let's get into details about the elliptic curves over finite fields.