Chapter 4: Public Key Cryptography

Foreword

Symmetric key cryptography uses a shared key for both encryption and decryption. The biggest problem with this is that the shared key needs to be exchanged between participants over a secure channel, which can be quite hard to achieve. It also defeats the objective of encryption if we don't have a secure channel for communication in the first place.

This is where **asymmetric cryptography** comes in.

Asymmetric Cryptography

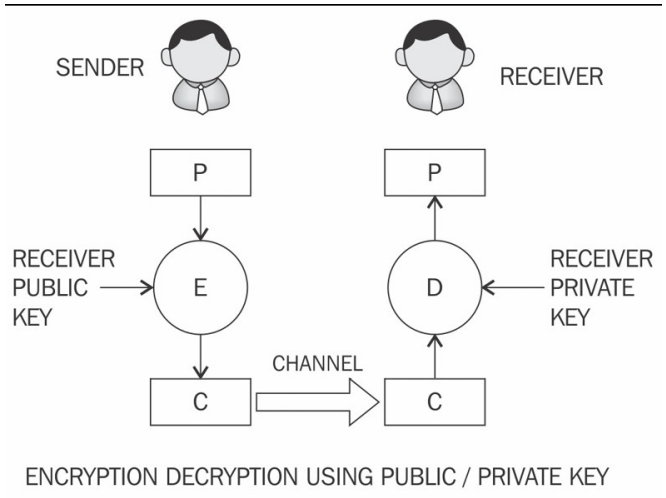
Asymmetric cryptography refers to a type of cryptography where the key that is used to encrypt the data is different from the key hat is used to decrypt the data.

These keys are called private and public keys, respectively, which why asymmetric cryptography is also known as **public key cryptography**. It uses the public and private keys to encrypt and decrypt data, respectively.

- The public key is constructed from the private key and can be freely broadcasted to other users.
- Public-key algorithms enable the creation of a public key from a randomly generated private key.
- The created public key could not be used to infer the private key.
- In other words, the creation of the public key from the private key is a one-way process.
- The public-key algorithm not only performs encryption, but also provides authentication functionality.

Asymmetric Cryptography Cont.

The following diagram shows how the receiver uses public key cryptography to verify the integrity of the received message:



Some asymmetric cryptography schemes currently in use are *RSA*, *DSA* and *ElGamal*.

Asymmetric Cryptography Uses

- Security mechanisms offered by public key cryptosystems include key establishment, digital signatures, identification, encryption, and decryption.
- The public-private key concept in asymmetric cryptography is used in bitcoin and other cryptocurrencies to identify the owner of the asset.
    - Private keys are used to represent the ownership of coins in cryptocurrency
- Key establishment mechanisms are concerned with the design of protocols that allow the setting up of keys over an insecure channel.
- Non-repudiation (*defined in Chapter 3, Symmetric Cryptography, as the assurance that an action, once taken by someone, cannot be denied later*) services, a very desirable property in many scenarios, can be provided using digital signatures.
    - *Sometimes, it is important not only to authenticate a user but also to identify the entity involved in a transaction. This can also be achieved by a combination of digital signatures and challenge-response protocols*.

Asymmetric Cryptography Drawbacks

Compared to symmetric key algorithms, public key algorithms are slower in terms of computation.

- Therefore, they're not commonly used for encrypting large files or data that *actually* required encryption
- Normally, public key algorithms are used for exchanging keys for symmetric algorithms. Once the keys are established securely, symmetric key algorithms can be used to encrypt the data.

How Reliable is Asymmetric Cryptography?

The strength of a public-key cryptography system depends on how feasible it is to infer the private key from the publicly available information about the key. Although it is infeasible, it is **not** impossible, and security relies solely on the key size and key generation mechanism.

- All asymmetric key algorithms are based on a number theory problem that ensures the characteristics required for key generation and the encryption and decryption processes.
- Based on different ways of solving the mathematical problem in number theory, asymmetrical key generation is broadly characterized in three ways: **prime factorization**, **discrete logarithm**, and **elliptic-curve**.
    - All public-private key algorithms are based on these mathematical problems. All these problems are similar in functionality to trapdoor functions.

> A **trapdoor** function is a function where it is easy to compute the values in one way but infeasible to find the inverse. This means that it is difficult to find the original input values supplied to the function from the result. This functionality is widely used in asymmetrical cryptography.

Prime Factorization

> **Prime factorization** is a concept in number theory regarding the decomposition of a number into the product of two prime numbers.

- Prime factorization is a subset of integer factorization, in which a composite number is factored into the product of any two integers.
- It is challenging to find the factors of semi-primes (numbers that result from the product of two prime numbers) because they have only a single pair of factors, and the complexity of finding the factors increases as the size of the prime number used in the product increases.
- There is no known efficient factorization algorithm for finding factors when numbers are of a certain size.
- RSA uses prime factorization, presuming that it's really difficult to find the private key from the exposed product of prime numbers. This presumed difficulty is the reason behind the use of prime factorization in cryptography.

Discrete Logarithm

> A **discrete logarithm** is based on the modular arithmetic settings on a discrete logarithm where the solution is infeasible to find.

- The logarithm $\log_b a$ is a discrete logarithm that has the integer solution x so that $b^x = a$.
  - There is no efficient general method for finding the solution to a discrete logarithm.

- When modular arithmetic is used with a discrete logarithm, it's known as modular exponentiation, and this problem becomes really difficult.
  - This problem is generally used with the Diffie-Hellman key exchange algorithm.

For example, it's easy to calculate the result of $3^3 mod 5 = 2$*, however, it would be almost impossible to find* 3^3.

RSA Cryptosystems

RSA is one of the initial implementations of public-private cryptography.

- It uses the principle of prime factorization to generate a public-private key pair, which acts as a trapdoor function.
- Encryption is performed using the public key, which is distributed to everyone, and decryption is performed using the secretly kept private key.
- The public and private key pair are computed with the help of two large prime numbers.
- The public key is published to the user, and the private key is kept secret. The prime numbers are also kept secret.
- As long as the prime numbers used are large, it is infeasible to compute the private key from the public key.
- The whole RSA cryptosystem is based on the number theory problem of integer factorization, which ensures that the difficulty of prime factorization is proportional to the size of the prime numbers used.

RSA parameter generation

1. Select two distinct large prime numbers; p and q.
2. Compute $n = p * q$ and $\varphi(n) = (p-1) * (q-1)$.
3. Choose a random integer e, such that $1 < e < \varphi(n)$ and $gcd(e, \varphi(n)) = 1$, that is, integer e and $\varphi(n)$, are coprimes.
4. Find $d \equiv e - 1 (mod \varphi(n))$.
   - More clearly, find d such that $d * e \equiv 1 (mod \varphi(n))$, meaning find a value d such that $d * e$ has a remainder of 1 when divided by $\varphi(n)$.
5. The public key is denoted by (e, n) and the private key by (d, p, q). Here, e is called the public exponent, and d the private exponent.

Encryption and Decryption using RSA

Now that we've generated our parameters, we can get started with the **main** encryption and decryption process.

- Encryption is performed in RSA using the distributed public key.
- Message M is converted to integer m such that $0 \leq m < n$. Ciphertext c is computed using the exposed public exponent, as follows:

$$c \equiv m^e mod(n)$$

- Anyone who possesses the public exponent can perform encryption on the message and transmit it to whoever possesses the private exponent.
- Whoever has access to the ciphertext and private exponent can perform decryption as follows:

$$m \equiv c^d mod(n)$$

Encryption and Decryption using RSA Cont.

- Message M could be regenerated from the decrypted integer m.
- This is how RSA makes use of the prime factorization technique to perform encryption and decryption.
- The process could be performed reasonably quickly for small messages, but it is not the preferred way of encryption for large messages.
- This is why RSA is widely used in cryptographic primitives, such as digital signatures, rather than encryption.

Elliptic-curve cryptography

ECC is a public-private cryptography based on the elliptic curve mentioned earlier.

- ECC performs the addition of points on the elliptic curve to compute public-private key pairs.
- ECC requires smaller key sizes than other asymmetric key cryptosystems, such as RSA.
- ECC is widely used in key exchange mechanisms and digital signatures and is rarely used in encryption systems.

ECC vs. RSA

- RSA cryptography uses prime factorization. The factorization of a semi-prime number is really difficult. When used in this domain, it forms a trapdoor (one way) function.
- Similarly, elliptic-curve-based algorithms can use discrete logarithms. Finding the discrete logarithm of a random element on an elliptic curve with respect to a point on the same curve is a severe problem.
- ECC provides the same level of security as RSA, but has a smaller key size. A 256-bit ECC key is equivalent to a 3,072-bit RSA key.
    - Similarly, a 384-bit ECC key provides the same level of security as a 7,680-bit RSA key, and so on. We can clearly see the advantage of less computation time due to the smaller key size in ECC.

Due to its key size advantage compared to RSA, ECC is used in Bitcoin's addressing system, along with transaction signing operations.

It is also popular in other blockchain applications. Other applications of ECC are Tor, iMessages, SSH, and SSL/TLS.

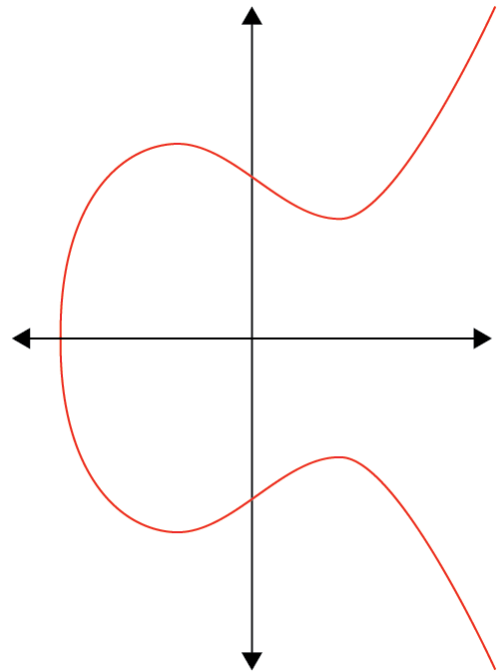
Elliptic-curve

An **elliptic curve** is a plane real algebraic curve with an equation in the form $y^2 = x^3 + ax + bmod(p)$. They are constructed in a finite field.

- The modulus operation on p indicates that the curve is over a finite field of prime numbers of the order p.

A finite field is a field with a finite number of elements defined by parameter p, which is a prime number. Thus, the finite field is $F_p = 0, \ldots, p-1$. It is represented by modulo p in the equation.

- An elliptic curve should be a non-singular curve, meaning no cusps, self-intersections, or isolated points.
- An elliptic curve on a finite field is used in the cryptography system.



Example elliptic curve

Elliptic-Curve Cont.

- All the elements used in ECC must be agreed upon by the cryptography actors.
  - These elements are called elliptical curve domain parameters.

$\{p, a, b, G, n, h\}$ are the parameters used in ECC. These parameters are defined as follows:

> p: The finite field is defined by this prime number.
> a **and** b: These are the constants used in the equation.
> G: The set of all points in the curve is defined by this generator, also known as the base point.
> n: This represents the order of the base point or generator G, a smallest positive number n such that nG = ∞.
> h: This is the cofactor, which is the ratio of the orders of the group and sub group (n), and it must be small (h <= 4), usually h=1.

- Bitcoin's Elliptical Curve Digital Signature Algorithm (ECDSA) curve uses a unique set of domain parameters defined in secp256k1. Technical specifications of this curve are beyond the scope of this class.

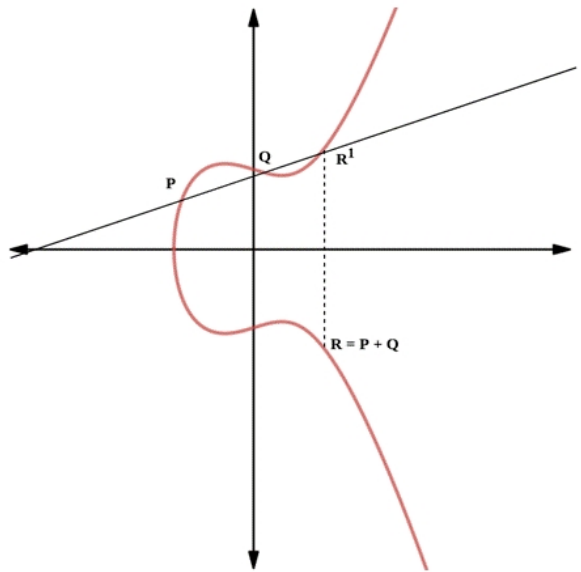
The operations performed on an elliptical curve are called **dot operations**, and they are **point addition** and **point doubling**.

Point Addition

> **Point addition** is a process where two different points are added.

Let's assume that P and Q are two points on the elliptic curve. P is not equal to Q; they are two distinct points on the curve.

Point addition is explained geometrically in the image to the right:



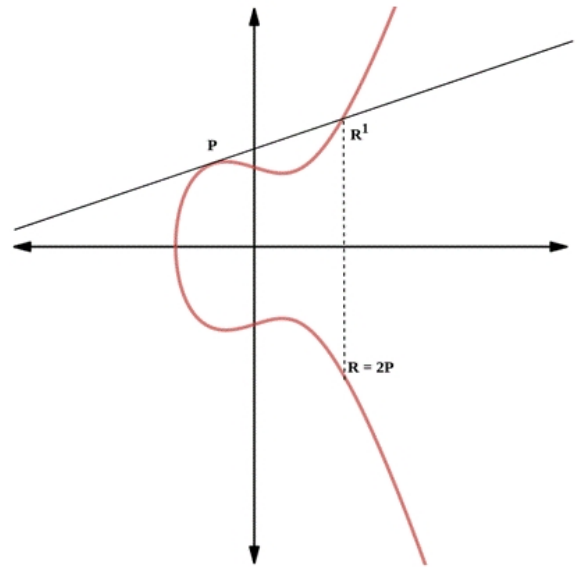
The following steps are performed on the elliptic curve as shown in the image to the side to add two points.

1. Draw a straight line between points $P(x_1, y_1)$ and $Q(x_2, y_2)$
2. The line will intersect the elliptic curve at point R^1
3. A reflection of point R^1 about the x axis gives point $R(x_3, y_3)$, which is the result of the addition of P and Q

Point Doubling

> **Point doubling** means that the same point is added to itself.

Point doubling is a similar operation to point addition, with the exception that point Q is moved to the same location as point P where $P = Q$.



The following steps are performed on the elliptic curve as shown in the image above to compute point doubling:

1. Draw a tangent (since there is only one point) to the curve at point P
2. This line will intersect the curve at point R^1
3. A reflection of point R^1 about the x axis gives point R, and this is point doubling or a multiple of R ($2R$)

- Why is this important?

Point doubling is the concept used in ECC to construct the public key from the private key.

Public Key Generation

Now that we have defined point doubling, we can calculate a point on the curve that is the multiple of the given point generator, point G (for example, $4G = G + G + G + G$), and this could be computed using point doubling.

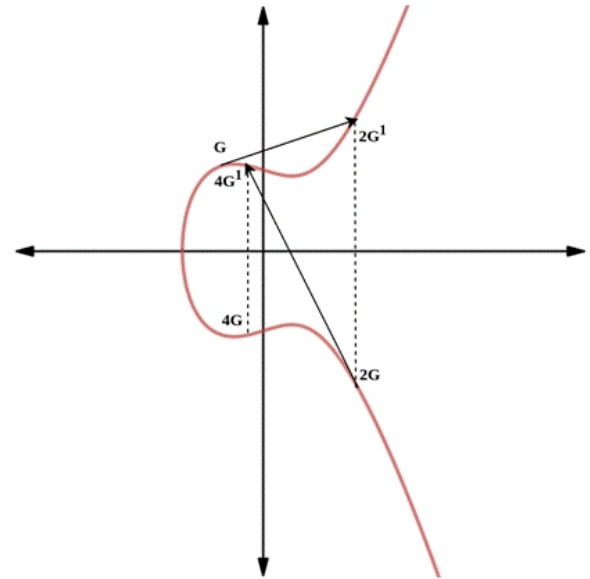
- Let's use this concept to compute a public-private key pair in an asymmetric cryptography system.
- Every curve domain parameter is the same for a given specification according to the technical specifications of *secp256k1*, in the case of Bitcoin and some other blockchain applications.

Public Key Generation

- Let's say k is a randomly chosen private key, and K is the public key to be generated. The generator of the curve, G, has a standard value.
- The public key could be computed by performing the following operation on the curve:

$$K = k * G$$

- We can generate the public key using this equation on an elliptic curve using point doubling.
- Point doubling on an elliptic curve is a one-way operation. It is, therefore, a challenging task to compute the multiplied value k after the required point K has been found.
- In this case, points 2G and 4G are derived using point doubling of G on the given curve.



> This geometrical method could be used to generate the public key, K, by multiplying the generator by the private key k times.

Digital Signatures

> A **digital signature** is a method of providing proof of ownership of digital documents.

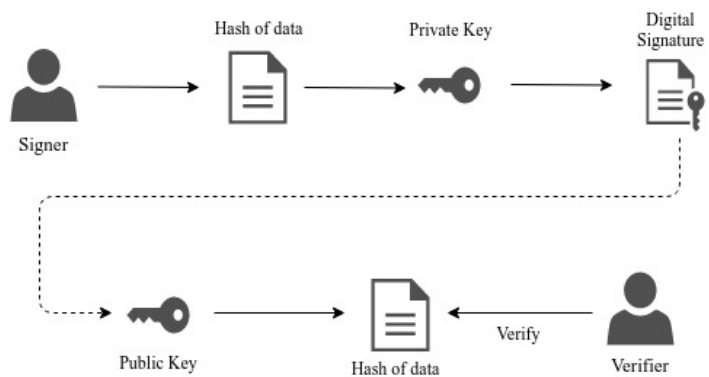
- Public-private key cryptography is widely used in the field of digital signatures due to their asymmetric key property.
    - The owner can use the private key to sign a message or document, and the verifier can verify their ownership using the public key, which is distributed to everyone.
- In other words, the asymmetric key cryptography provides a way to identify an entity. Anyone can prove the identity by owning the private key. Creating an identity for the participants allows them to perform operations such as asset management.

The process is similar to the handwritten signatures used in the real world, where an owner of an asset can use their signature to perform any action on that asset and anyone can verify the signature by comparing it with a signature that was used previously. The digital signature is more secure than the hand-written signature since it is infeasible to forge a signature without owning the private key.

How It’s Done

As shown in this image, the digital signature process consists of two parts: **signing**, and **verification**.

- Unlike with encryption, the digital signature performs the first operation, signing using the private key. Verification uses the public key distributed by the signer.



Signing process

> **The signing operation** is performed by the owner of the message using the private key to *prove their authenticity*.

Let's say Alice is the owner of a document that has a message m, and wants to distribute it to others in the network.

Now, Alice will initially hash the message and use her private key to sign the document.

The signature is created as follows, where F_s is the signature function, F_h is the hashing function, m is the message, and dA is Alice's private key:

$$S = F_s(F_h(m), dA)$$

Alice will now distribute her signature, along with the message, to everyone in the network.

Verification Process

> **Verification** is a process performed by anyone who possesses information that is made public by the owner.

Public information usually has a public key, a message, and the signature of the message.

Let’s assume that Bob possesses all the public information and wishes to verify the message to check its authenticity.

- Bob uses a signature verification algorithm, which requires a hash of the message, the public key, and the signature.
- The algorithm will verify that the message hasn’t been tampered with by anyone.

Elliptical Curve Digital Signature Algorithm (ECDSA)

> Commonly used to sign transactions and events in blockchains (like Bitcoin), ECDSA is a digital signature algorithm that makes use of ECC to create the key pairs used in the signing and verification process of the digital signature.

The ECDSA signature $\{r, s\}$ has the following simple explanation:

- The signing function encodes a random point r (represented by its x-coordinate only) through elliptic-curve transformations using the private key `privKey` and the message hash h into a number s, which is the proof that the message signer knows the private key `privKey`.

- The signature $\{r, s\}$ cannot reveal the private key due to the difficulty of the ECDLP problem.

> The base assumption is that finding the discrete logarithm of a random elliptic curve element with respect to a publicly known base point is infeasible: this is the **"elliptic curve discrete logarithm problem" (ECDLP)**

- The signature verification decodes the proof number s from the signature back to its original point R, using the public key `pubKey` and the message hash h and compares the x-coordinate of the recovered R with the r value from the signature.

> For a more detailed process of what's going on here, check this link.

Asset Ownership in Blockchain

- A blockchain network is a decentralized peer-to-peer network, where nodes communicate with each other to create, exchange, and validate blocks.
- Most of the users in the blockchain network are interested in the application layer of the blockchain where operations can be performed by creating transactions.
- An identity can easily be created in the network using public/private keys, as we had seen in previous slides.

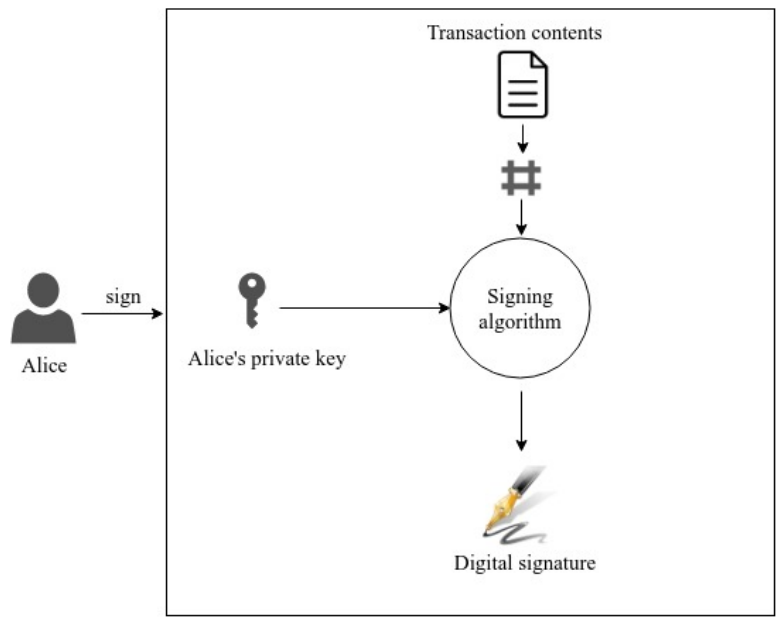
Asset Ownership in Blockchain Cont.

- Nodes can perform operations such as asset creation or asset transfer.
- Each operation that deals with assets is valid if they are approved by the asset owner.
    - The asset owners prove their identity by signing the transactions using their private key.
- *Asset management operations*, such as transferring the asset, can only be performed by the owner, but it can be verified by anyone in the network.
- All the operation details are embedded in transactions, and digital signatures are used by the asset owners to sign those transactions.
    - They then broadcast these transactions to every node in the blockchain so that they are included in the next block to be appended to the blockchain ledger.

Transferring an Asset: An Example

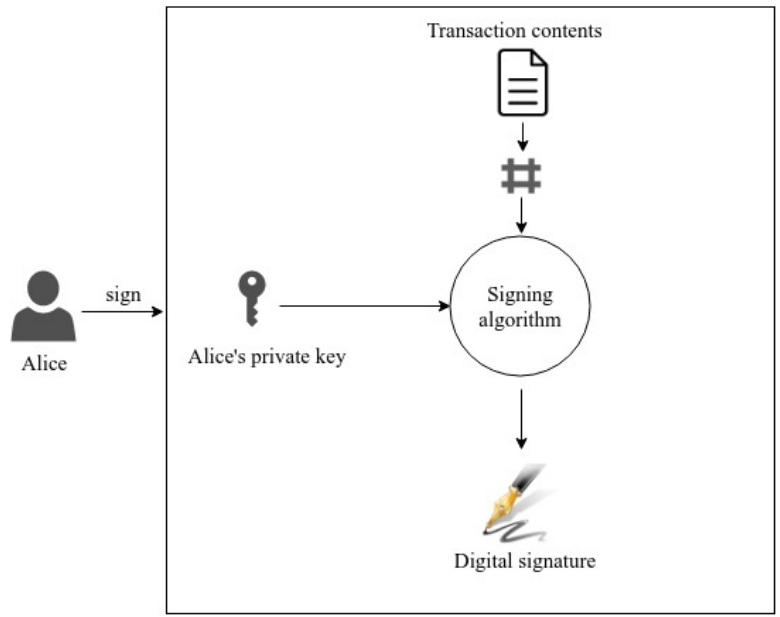
The ownership of an asset can be proven by possession of the private key for an address (public key) to which the asset belongs. Whenever the ownership of an asset needs to be transferred, users use their private key to sign the transaction, firstly proving their ownership, and from there transferring the ownership to the desired user. For example:

- Alice is a user who claims to own an asset in the blockchain network. She wishes to transfer this asset to her friend, Bob.
- She has access to her private key, and she uses it to create a transaction with a digital signature, which will prove that she owns the asset.
- The signing process is performed using the digital signature that is similar to what we used earlier in this chapter.
- When ECDSA was used as a signing algorithm, which also made use of an ECC key pair.
- The following image to the right shows how Alice creates a signature by signing the transaction contents that contain asset transfer information.



Transferring an Asset: An Example Cont.

- Once the asset transfer information has been signed by the user, other information, such as the user's public key for verification, the destination public address, and other information that is necessary for verification, is provided.
    - This information is broadcast to all the nodes so that it can be included in the blockchain.
- Alice will include information such as her public key and Bob's address in the transaction.
    - The information provided in the transaction should suffice for Bob to claim that the asset belongs to him once the transaction has been included in any of the blocks in the blockchain.



It should be noted that the public address of a node acts as the identifier for that node and is constructed from the public key by performing hashing and encoding. We'll talk about this more in later chapters.

Claiming the Asset

- When an asset is transferred and transmitted with the information required in the transaction, the blockchain network will ensure that it is included in the blockchain after validating the transaction and the block in which it was included.
    - When the transaction is included in the blockchain, everyone will be able to see this transaction, but only the owner to whom it was addressed will be able to claim the asset.
- The level of security provided by the asymmetric key cryptography used in the digital signature will make sure that **only** the node that owns the private key corresponding to the public address will be able to claim the asset that was transferred to that address.