

ECC nedir:

ECC, kısaltması "Elliptic Curve Cryptography" olan "Elips Eğrisi Şifreleme" anlamına gelir. ECC, şifreleme ve dijital imzalama gibi kriptografik işlemler için kullanılan bir algoritma ailesidir.

ECC, geleneksel olarak kullanılan diğer şifreleme algoritmaları (örneğin RSA) ile karşılaştırıldığında daha küçük anahtar boyutlarına ve daha hızlı işlem yapabilme yeteneğine sahiptir. Bu nedenle, özellikle kaynak kısıtlı ortamlarda (örneğin mobil cihazlar) ve güvenlik gerektiren uygulamalarda tercih edilen bir şifreleme yöntemidir.

Elips eğrisi şifreleme, bir elips eğrisi üzerindeki matematiksel işlemleri kullanarak şifreleme ve dijital imza oluşturma işlemlerini gerçekleştirir. Elips eğrisi, bir düzlemdeki noktaların oluşturduğu matematiksel bir eğridir. Bu eğrinin özel matematiksel özellikleri, şifreleme ve dijital imza algoritmalarında kullanılmak üzere tasarlanmıştır.

ECC'nin güvenliği, kullanılan elips eğrisinin matematiksel özelliklerine dayanır. Elips eğrisi şifrelemesi, büyük asal sayılar üzerindeki modüler aritmetiği içeren hesaplamalara dayanır. ECC, hesaplama yoğunluğu gerektiren işlemleri gerçekleştirirken düşük güç tüketimi ve hızlı işlem avantajları sunar.

Genel olarak, ECC, güvenli iletişimde ve kriptografik uygulamalarda kullanılan etkili bir şifreleme ve dijital imza yöntemidir.

Kriptografik algoritmalar ve protokoller, verilerin güvenliğini sağlamak için kullanılan matematiksel işlemlerdir. Bu algoritmalar, verilerin şifrlenmesi, şifrelerin çözülmesi ve güvenli veri iletişimi gibi işlemlerde kullanılır. Aşağıda, kriptografik algoritmaların ve protokollerin genel açıklamalarını ve bazı türleri:

- **Symmetric encryption:** Used to conceal the contents of blocks or streams of data of any size, including messages, files, encryption keys, and passwords.
- **Asymmetric encryption:** Used to conceal small blocks of data, such as encryption keys and hash function values, which are used in digital signatures.
- **Data integrity algorithms:** Used to protect blocks of data, such as messages, from alteration.
- **Authentication protocols:** These are schemes based on the use of cryptographic algorithms designed to authenticate the identity of entities.

- Şifreleme Algoritmaları: Bu algoritmalar, verileri şifrelemek için kullanılır. Şifreleme, verilerin anlaşılabilir hale getirilmesini sağlar ve sadece yetkili kullanıcıların şifreyi çözebilmesini mümkün kılar. Simetrik ve asimetrik şifreleme olmak üzere iki temel türü vardır.
- Simetrik Şifreleme: Aynı anahtarın hem şifreleme hem de şifre çözme işlemlerinde kullanıldığı bir şifreleme türüdür. AES (Advanced Encryption Standard) ve DES (Data Encryption Standard) gibi algoritmalar simetrik şifreleme örnekleridir.
- Asimetrik Şifreleme: Farklı anahtar çiftlerinin kullanıldığı bir şifreleme türüdür. Bir anahtar (genellikle "halka açık anahtar" olarak adlandırılır) verileri şifrelerken kullanılırken, diğer anahtar (genellikle "özel anahtar" olarak adlandırılır) şifreleri çözmek için kullanılır. RSA ve ElGamal gibi algoritmalar asimetrik şifreleme örnekleridir.

RSA encryption (with the public key) is **faster** than the corresponding operation with ElGamal, or half of Diffie-Hellman. On the other hand, RSA decryption (with a private key) is a bit slower than ElGamal decryption or the other half of Diffie-Hellman (especially the elliptic curve variants).

When to use RSA and when ElGamal asymmetric encryption

crypto.stackexchange.com/questions/1677/when-to-use-rsa-and-when-el...

- Karma Fonksiyonları: Karma fonksiyonları, verilerin benzersiz bir karmasını (hash) oluşturan algoritmalar. Bu algoritmalar, parola saklama, dijital imzalar ve veri bütünlüğü doğrulama gibi birçok uygulamada kullanılır. MD5, SHA-1 ve SHA-256 gibi karma fonksiyonları örnek verilebilir.
- Sayı Üreteçleri: Sayı üreteçleri, rastgele sayıların oluşturulması için kullanılan algoritmalar. Güvenli rastgele sayı üretimi, şifreleme ve kimlik doğrulama protokollerinde önemlidir. Pseudorandom sayı üreteçleri (PRNG) ve gerçek rastgele sayı üreteçleri (TRNG) olmak üzere iki tür sayı üretici bulunur.
- İmza Algoritmaları: İmza algoritmaları, bir mesajın doğruluğunu ve bütünlüğünü sağlamak için kullanılır. Mesajın gönderici tarafından imzalandığı ve imzanın alıcı tarafından doğrulandığı bir güvenlik mekanizmasıdır. RSA ve DSA (Digital Signature Algorithm) gibi algoritmalar imza oluşturma ve doğrulama için kullanılır.
- Anahtar Değişim Protokolleri: Anahtar değişim protokolleri, iletişimde kullanılan şifreleme anahtarlarının güvenli bir şekilde paylaşılmasını sağlar. Diffie-Hellman anahtar değişimi protokolü, güvenli anahtar değişimi sağlamak için yaygın olarak kullanılan bir örnektir.
- SSL/TLS Protokolü: Secure Socket Layer (SSL) ve Transport Layer Security (TLS), web tabanlı iletişimde kullanılan güvenli bir protokol çiftidir. Bu protokoller, sunucu ile istemci arasındaki verilerin şifrlenmesini ve kimlik doğrulamasını sağlar. Bu şekilde, güvenli bir şekilde veri alışverişi yapılabilir.

Bu liste, kriptografik algoritmaların ve protokollerin sadece birkaç örneğini içermektedir. Kriptografi alanı sürekli gelişmekte olup, yeni algoritmalar ve protokoller ortaya çıkmaktadır. Güvenlik ihtiyaçlarınıza ve uygulamanıza en uygun algoritmayı ve protokolü seçmek için güvenlik uzmanları ve kriptografi standartlarına başvurmak önemlidir.

private key nedir :

Özel anahtar (private key), kriptografik bir algoritma kullanılarak oluşturulan bir anahtar çiftinin (key pair) gizli olan kısmıdır. Kriptografi sistemlerinde, genellikle açık anahtarlı şifreleme (asimetrik şifreleme) algoritmaları kullanılır. Bu algoritmalar, iki anahtarın birbirine matematiksel olarak bağlı olduğu bir çift oluşturur: özel anahtar (private key) ve açık anahtar (public key).

Özel anahtar, bir kişi ya da kuruluşun kimliklerini doğrulamak, dijital imza oluşturmak veya şifreli verileri çözmek gibi işlemleri gerçekleştirmek için kullanılır. Özel anahtar, gizli ve güvende tutulması gereken bir bilgidir, çünkü bu anahtara sahip olan kişi, ilgili kriptografik işlemleri gerçekleştirebilir ve kişinin kimliğini taklit edebilir.

Örneğin, Bitcoin gibi kripto para birimlerinde, her kullanıcının bir çift özel ve açık anahtarı vardır. Açık anahtar, diğer kullanıcılar tarafından bilinir ve bu anahtar, kullanıcının kripto para göndermek için kullanılabilecek adresini temsil eder. Özel anahtar ise kullanıcının sahip olduğu varlıkları kontrol etmek için kullanılır. Özel anahtarın gizliliği önemlidir, çünkü başkaları özel anahtarı ele geçirirse, kullanıcının kripto para birimine erişimi ve kontrolü kaybedebilir.

Özetle, özel anahtar (private key), kriptografik sistemlerde kullanılan bir anahtar çiftinin gizli olan kısmıdır ve kimlik doğrulama, dijital imza oluşturma veya şifreli verileri çözme gibi işlemleri gerçekleştirmek için kullanılır.

public key nedir ve private key ile ne alakası vardır

Açık anahtar (public key), kriptografik bir algoritma kullanılarak oluşturulan bir anahtar çiftinin (key pair) herkese açık olan kısmıdır. Özel anahtar (private key) ile birlikte kullanılır ve ikisi arasında matematiksel bir ilişki vardır.

Asimetrik şifreleme (açık anahtarlı şifreleme) algoritmalarında, özel anahtar ve açık anahtar birbirine bağlıdır. Bir kişi ya da kuruluş, özel anahtarı kullanarak bir çift anahtar oluşturur. Özel anahtar, gizli tutulmalıdır ve yalnızca sahibi tarafından erişilebilir olmalıydı. Açık anahtar ise herkese açık bir şekilde paylaşılabilir.

Açık anahtar, diğer kişilerin verileri şifrelemek veya dijital olarak imzalamak için kullanabilecekleri bir anahtardır. Özel anahtar ile ilişkili açık anahtar, birbirini tamamlayan anahtar çiftleridir. Bir kişi bir mesajı şifrelemek istediğinde, alıcının açık anahtarını kullanarak şifreler. Bu şifrelenmiş veri, yalnızca ilgili özel anahtarın sahibi tarafından çözülebilir. Bu nedenle, açık anahtar herkese açık olarak paylaşılabilirken, özel anahtar gizli tutulmalıdır.

Özetle, açık anahtar (public key), herkese açık olan bir anahtardır ve diğer kişilerin verileri şifrelemek veya dijital olarak imzalamak için kullanabilecekleri bir anahtardır. Özel anahtar (private key) ile

birlikte kullanılır ve ikisi arasında matematiksel bir ilişki vardır: biri ile şifrelenen veri, sadece diğerine ait olan anahtar ile çözülebilir.

bitcoin ile ne alakası vardır:

Bitcoin, kripto para birimi olarak dijital bir varlık olduğu için kriptografi tekniklerini kullanır. Bitcoin'in işleyişinde, açık anahtarlı (asimetrik) şifreleme yöntemleri ve özel ve açık anahtar çiftleri büyük önem taşır. Bu nedenle, Bitcoin ile ilgili bir konuşmada özel anahtar ve açık anahtar kavramları sık sık kullanılır.

Bitcoin ağındaki her kullanıcının bir çift özel ve açık anahtarı vardır. Açık anahtar, diğer kullanıcılara veya ağa açık bir şekilde paylaşılabilir. Bitcoin işlemleri, alıcı tarafının açık anahtarını kullanarak şifrelenir ve ağ üzerinde iletilir. Bu sayede, yalnızca alıcıya ait olan özel anahtarla bu işlem çözülebilir ve alıcı, Bitcoin'i kontrol edebilir.

Özel anahtar, Bitcoin cüzdanının sahibi tarafından gizli tutulması gereken bir bilgidir. Özel anahtar, Bitcoin'i harcama yetkisine sahip olduğu için büyük önem taşır. Bir kişinin özel anahtarı ele geçirilirse, bu kişi Bitcoin'lerini kontrol etme yetkisini kaybedebilir.

Özetle, Bitcoin ile ilgili olarak özel anahtar ve açık anahtar kavramları, güvenli kimlik doğrulama, işlem şifreleme ve Bitcoin'in kontrolü gibi kriptografik işlemleri anlamak için önemlidir. Özel anahtar, Bitcoin sahipliğini temsil ederken, açık anahtar, işlemlerin şifrelenmesi ve doğrulanmasında kullanılır.

Private key public key dönüştürme :

Özel anahtarı genel anahtara dönüştürme işlemi, Bitcoin ve diğer dijital para birimlerinde kullanılan asimetrik şifreleme algoritmasına dayanan ECDSA (Elliptic Curve Digital Signature Algorithm) algoritması tarafından gerçekleştirilir.

Genel :

ECDSA'da, eliptik eğriler özel ve genel anahtarların oluşturulmasında kullanılır. Elips eğrisi, "eğri noktaları" adı verilen, eliptik eğri denklemini karşılayan (x, y) çiftlerini içeren benzersiz bir küme belirler.

Özel anahtar oluşturulurken, uzun bir rastgele sayı olan "gizli sayı" (private number) seçilir. Ardından, bu gizli sayının belirli bir noktadaki eliptik eğriyle "nokta çarpma" (point multiplication) olarak adlandırılan karmaşık bir matematiksel işlem gerçekleştirilir. Kullanılan nokta, algoritmanın belirlediği kurallara göre seçilir.

Nokta çarpma işlemi, eliptik eğri üzerindeki Özel Anahtar Adresi'ni üretir. Bu nihai adres, özel anahtarı temsil eder.

Özel anahtarı genel anahtara dönüştürmek için aynı nokta çarpma işlemi kullanılır. Ancak, gizli sayı yerine özel anahtar kullanılır. İşlem gerçekleştirildikten sonra, eliptik eğri üzerinde Genel Anahtar Adresi üretilir.

Elde edilen genel anahtar, herkese açık bir şekilde paylaşılabilir ve Bitcoin işlemlerinde şifreleme ve doğrulama için kullanılabilir. Uygun algoritma kullanılarak, başka bir kişi orijinal özel anahtarı genel anahtardan çıkaramaz.

Özel anahtarın genel anahtara dönüştürülmesi mümkün olmasına rağmen, genel anahtarın özel anahtara dönüştürülmesi mümkün değildir. Bu, ECDSA ve eliptik eğrilerin kullanıldığı matematiksel işlemlerin geri dönüşü zor olması nedeniyle mümkün olmaktadır. Bu, özel anahtarın gizliliğini ve genel anahtarın güvenliğini sağlar.

Detay :

Bir gizli anahtardan elitik eğri kullanarak bir genel anahtar üretmek için aşağıdaki adımları izleyebiliriz:

- Elitik eğri seçimi: Belirli bir sonlu alandaki elitik eğriyi seçin. Eğri parametreleri, elitik eğrinin karakteristiklerini tanımlar.
- Gizli anahtar oluşturma: Rastgele bir gizli anahtar oluşturun. Gizli anahtar, elitik eğri parametreleri tarafından belirlenen belirli bir aralık içindeki bir sayıdır. Gizli anahtar gizli tutulmalı ve kimseyle paylaşılmamalıdır.
- Skaler çarpma işlemi: Gizli anahtarı, elitik eğride önceden belirlenmiş bir baz noktayla çarpın. Baz nokta, önceden anlaşılan eğri üzerindeki sabit bir noktadır. Skaler çarpma işlemi, baz noktayı gizli anahtar değerine eşit sayıda kendisiyle toplayarak gerçekleştirilir.
- Genel anahtarı elde etme: Skaler çarpma işlemi sonucunda elitik eğride yeni bir nokta elde edilir ve bu nokta genel anahtar olur. Bu nokta, eğride bir koordinatı temsil eder ve bir x-koordinatı ve bir y-koordinatı içerir.

Elde edilen genel anahtar genellikle (x, y) koordinatlarını içeren bir çift olarak temsil edilir. Genel anahtar, anahtar değişimi veya dijital imzalar gibi çeşitli kriptografik işlemler için kullanılabilir.

Elitik eğri parametrelerinin ve skaler çarpma için kullanılan özel algoritmanın, etkileşim kabiliyeti ve güvenlik sağlamak için standartlaştırılması gerektiğini belirtmek önemlidir. ECC, RSA gibi diğer kriptografik algoritmalara kıyasla daha küçük anahtar boyutlarıyla güçlü bir güvenlik sunar.

A, B ve C'nin arasında mesaj göndermek için bir public key ve private key senaryosu şu şekilde olabilir:

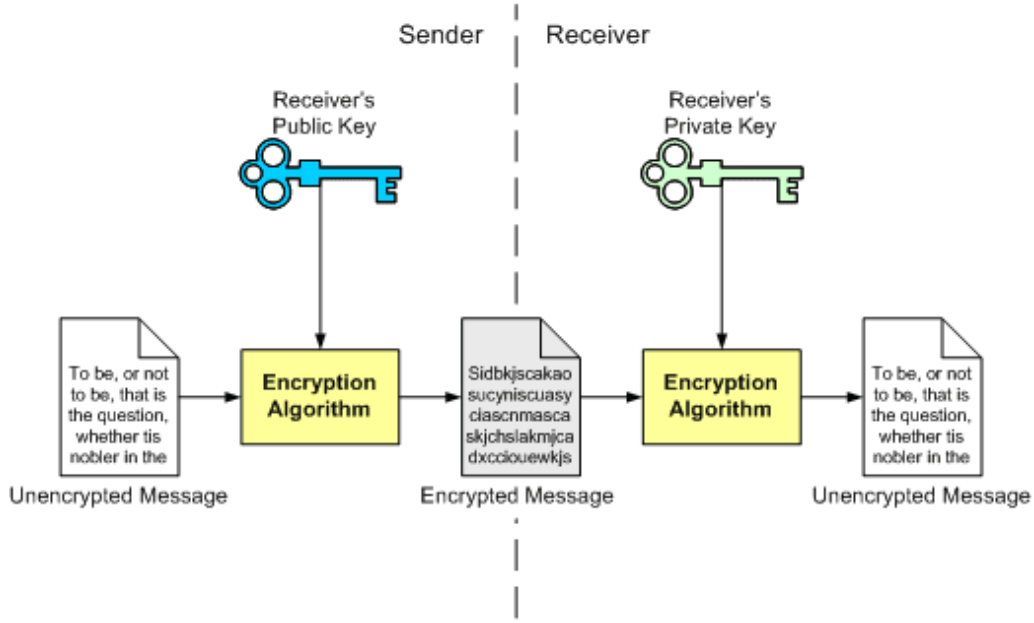
- Her bir kişi, kendine ait bir public key ve private key çifti oluşturur.
- A, A_PublicKey ve A_PrivateKey
- B, B_PublicKey ve B_PrivateKey
- C, C_PublicKey ve C_PrivateKey
- Mesaj göndermek isteyen A, mesajını şifrelemek için B'nin public key'ini kullanır.
- A, mesajını B_PublicKey ile şifreler ve şifreli mesajı B'ye gönderir.
- B, şifreli mesajı alır ve kendi private key'ini kullanarak mesajı çözer.
- B, şifreli mesajı B_PrivateKey ile çözer ve orijinal mesajı elde eder.

Bu senaryoda, mesaj A tarafından şifrelenirken, sadece B'nin sahip olduğu private key ile çözülebilir. Bu, mesajın sadece B tarafından okunabilmesini sağlar ve gizlilik sağlar. Diğer kişilerin, yani C'nin, A'nın mesajına erişimi olmaz.

Eğer A, C'ye de mesaj göndermek istiyorsa, A ayrıca C'nin public key'ini kullanarak mesajını C'ye şifreleyebilir. Aynı şekilde, C de kendi private key'ini kullanarak mesajı çözebilir ama C hiç bir şekilde A ve B nin public keyleri kullanarak A dan B ye gönderilen mesajın şifresini kıramaz.

Bu senaryo, genellikle asimetrik şifreleme olarak bilinen bir kriptografik yaklaşımı kullanır. Her bir kişinin kendi public key'ini herkesin erişebileceği bir şekilde paylaşması, diğer kişilerin mesajlarını şifrelemesine olanak tanır. Ancak, sadece ilgili kişinin sahip olduğu private key, şifreli mesajı çözmek için kullanılabilir. Bu şekilde, güvenli bir mesajlaşma sağlanabilir.

Şifreleme işleminin detayları aşağıda açıklanmıştır:



- A, mesajını B'ye göndermeden önce B'nin public key'ini kullanarak mesajını şifreler:
- A, mesajını şifrelemek için bir şifreleme algoritması (örneğin ECC) ve B'nin public key'ini kullanır.
- Şifreleme algoritması, mesajı parçalara ayırır ve her bir parçayı public key ile şifreler.
- Şifreleme işlemi sonucunda ortaya çıkan şifreli parçaları birleştirerek şifreli mesajı oluşturur.
- A, şifreli mesajı B'ye gönderir:
- Şifreli mesaj, A tarafından B'ye iletilir. Mesajın şifreli olduğu bilgisi B tarafından bilinir.
- B, şifreli mesajı çözmek için kendi private key'ini kullanır:
- B, şifreli mesajı private key ile çözmek için şifre çözme algoritmasını (örneğin RSA) kullanır.
- Şifre çözme algoritması, şifreli mesajı parçalara ayırır ve her bir parçayı private key ile çözer.
- Şifre çözme işlemi sonucunda ortaya çıkan parçaları birleştirerek orijinal mesajı elde eder.

Bu süreç, asimetrik şifreleme algoritması kullanılarak gerçekleştirilir. Şifreleme için kullanılan public key, sadece mesajı çözecek olan kişi tarafından kullanılabilir. Bu nedenle, sadece B'nin private key'iyle şifrelenen mesajı çözebilir. Diğer kişilerin mesaja erişimi olmaz.

Bu aşamalar, mesajın güvenli bir şekilde şifrlenmesi ve sadece alıcının mesajı çözebilmesini sağlayan genel bir işlem akışını temsil etmektedir. Elbette, gerçek uygulamalar daha karmaşık olabilir ve farklı kriptografik algoritmalar veya protokoller kullanılabilir.

Node js kullanarak örnek kod:

```
const crypto = require('crypto');

// Anahtar çiftinin oluşturulması
const { privateKey, publicKey } = crypto.generateKeyPairSync('ec', {
  namedCurve: 'secp256k1',
});

// Örnek bir mesaj
const message = 'Merhaba, bu bir örnek mesajdır.';

// Dijital imza oluşturma
const sign = crypto.createSign('SHA256');
sign.update(message);
const signature = sign.sign(privateKey, 'hex');

console.log('Dijital İmza:', signature);

// Dijital imza doğrulama
const verify = crypto.createVerify('SHA256');
verify.update(message);
const isValid = verify.verify(publicKey, signature, 'hex');

console.log('Doğrulama Sonucu:', isValid);
```

Kodun detaylı açıklaması:

```
const crypto = require('crypto');

// Anahtar çiftinin oluşturulması
const { privateKey, publicKey } =
```



```
crypto.generateKeyPairSync('ec', {
  namedCurve: 'secp256k1',
});
```

Burada, **crypto** modülü kullanılarak ECDSA anahtar çifti oluşturulmaktadır. **generateKeyPairSync** fonksiyonu ile asenkron olmayan bir şekilde anahtar çifti oluşturulur. İlk parametre olarak **'ec'** kullanarak ECDSA (Elliptic Curve Digital Signature Algorithm) algoritmasını belirtiyoruz. İkinci parametre olarak ise **namedCurve** olarak **'secp256k1'** belirterek Bitcoin'in kullandığı yaygın bir eliptik eğriyi (elliptic curve) temsil eden bir parametre sağlıyoruz. **generateKeyPairSync** fonksiyonu, **privateKey** ve **publicKey** olmak üzere bir nesne döndürür.

```
const message = 'Merhaba, bu bir örnek mesajdır.';
```

message değişkenine, imzalanacak olan örnek bir mesaj atanır.

```
const sign = crypto.createSign('SHA256');
sign.update(message);
const signature = sign.sign(privateKey, 'hex');
```

Burada, **createSign** fonksiyonu ile bir imza oluşturma nesnesi oluşturulur. İmza oluşturulacak algoritmayı **'SHA256'** olarak belirtiyoruz. Ardından, **update** fonksiyonu ile imzalanacak mesajı güncelliyoruz. Son olarak, **sign** fonksiyonu ile özel anahtarı (**privateKey**) ve imza formatını (**'hex'** olarak belirtilen onaltılık format) kullanarak imzayı oluşturuyoruz. Elde edilen imza, **signature** değişkenine atanır.

```
console.log('Dijital İmza:', signature);
```

Burada, oluşturulan imza konsola yazdırılır.

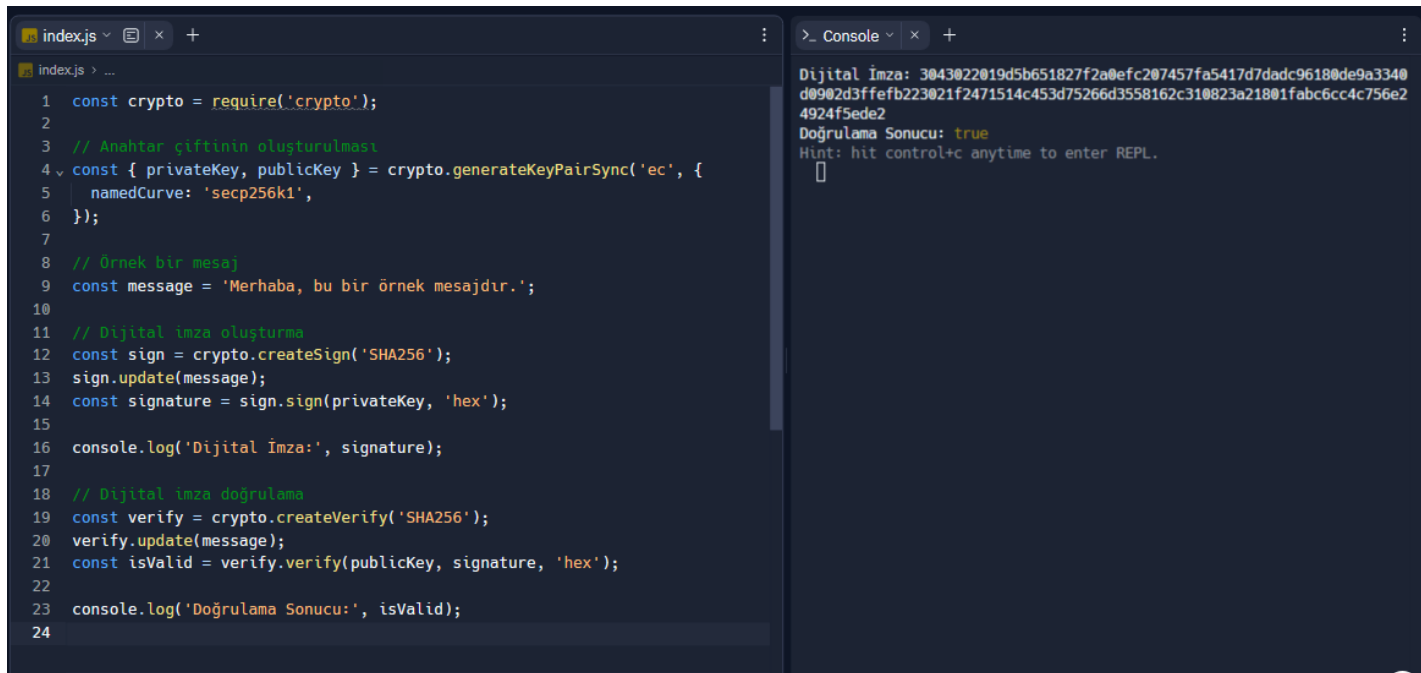
```
const verify = crypto.createVerify('SHA256');
verify.update(message);
const isValid = verify.verify(publicKey, signature,
'hex');
```

Bu bölümde, **createVerify** fonksiyonu ile bir doğrulama nesnesi oluşturulur. İmzanın doğrulanması için kullanılacak algoritmayı **'SHA256'** olarak belirtiyoruz. Ardından, **update** fonksiyonu ile doğrulanacak mesajı güncelliyoruz. Son olarak, **verify** fonksiyonu ile genel anahtarı (**publicKey**), imzayı (**signature**) ve imza formatını (**'hex'** olarak belirtilen onaltılık format) kullanarak doğrulama işlemini gerçekleştiriyoruz. Doğrulama sonucu, **isValid** değişkenine atanır.

```
console.log('Doğrulama Sonucu:', isValid);
```

Son olarak, doğrulama sonucu konsola yazdırılır.

Çıktı:



```
index.js > ...
1 const crypto = require('crypto');
2
3 // Anahtar çiftinin oluşturulması
4 const { privateKey, publicKey } = crypto.generateKeyPairSync('ec', {
5   namedCurve: 'secp256k1',
6 });
7
8 // Örnek bir mesaj
9 const message = 'Merhaba, bu bir örnek mesajdır.';
10
11 // Dijital imza oluşturma
12 const sign = crypto.createSign('SHA256');
13 sign.update(message);
14 const signature = sign.sign(privateKey, 'hex');
15
16 console.log('Dijital İmza:', signature);
17
18 // Dijital imza doğrulama
19 const verify = crypto.createVerify('SHA256');
20 verify.update(message);
21 const isValid = verify.verify(publicKey, signature, 'hex');
22
23 console.log('Doğrulama Sonucu:', isValid);
24
```

```
>_ Console >_ +
Dijital İmza: 3043022019d5b651827f2a0efc207457fa5417d7dadc96180de9a3340
d0902d3ffefb223021f2471514c453d75266d3558162c310823a21801fab6cc4c756e2
4924f5ede2
Doğrulama Sonucu: true
Hint: hit control+c anytime to enter REPL.

```

Anahtarları yazdırma

The image shows a code editor with a file named `index.js` and a console window. The code in `index.js` is as follows:

```
20 const isValid = verify.verify(publicKey, signature, 'hex');
21
22 console.log('Private Key:', privateKey.export({ format: 'pem',
23 type: 'pkcs8' }));
24 console.log('Public Key:', publicKey.export({ format: 'pem',
25 type: 'spki' }));
```

The console output shows the generated private and public keys in PEM format:

```
Private Key: -----BEGIN PRIVATE KEY-----
MIGeAgEAMBAGByqGSM49AgEGBSuBBAAKBG0wawIBAQQgBAdw4/Nc3cE...
n3uV
s4InecUJ32CSMPHQ0ATHeF2hRANCAAU2Umcm7vjL5Su0QEKPdUP8jEU5rPy
902J
v6crZvzAbck9cPRK9djC3NesaYIeR/kJcmHt1nd8dJYIQt5TS08z
-----END PRIVATE KEY-----

Public Key: -----BEGIN PUBLIC KEY-----
MFYwEAYHKoZIzj0CAQYFK4EEAAoDQgAEFNlJnJu745eUrtEBCj3VD/IxF0az
8vdN
lb+nK2b8wG3JPXD0SvXYwtzXrGmCHkf5CXJh7dZ3fHSWCELeU0jvMw==
-----END PUBLIC KEY-----

Hint: hit control+c anytime to enter REPL.
```

Bu şekilde, ECDSA ile dijital imza oluşturma ve doğrulama işlemlerini gerçekleştirebiliriz. ECDSA algoritması, kriptografik güvenlik gerektiren uygulamalarda kullanılan bir yöntemdir ve uygun anahtar boyutu ve diğer güvenlik önlemlerine dikkat edilmesi önemlidir.

Motaz

ahmed

abdulateef

Kaynaklar:

- [\(120\) Information Security and Cryptography Arabic - YouTube](#)
- [\(120\) Asymmetric Encryption - Simply explained - YouTube](#)
- [\(120\) How Public and Private Key Work In Your Crypto Wallets - YouTube](#)

Yazımsal ve mantıksal hatalar (open ai-chat gpt v-3.5) tarafından düzeltildi.