What is the difference between encrypting and signing in asymmetric encryption?

Asked 11 years, 3 months ago Active 3 months ago Viewed 145k times

284

159

What is the difference between encrypting some data vs signing some data (using RSA)?

Does it simply reverse the role of the public-private keys?

For example, I want to use my private key to generate messages so only I can possibly be the sender. I want my public key to be used to read the messages and I do not care who reads them. I want to be able to encrypt certain information and use it as a product-key for my software. I only care that I am the only one who can generate these. I would like to include my public key in my software to decrypt/read the signature of the key. I do not care who can read the data in the key, I only care that I am the only verifiable one who can generate them.

Is signing useful in this scenario?

encryption rsa signing license-key

edited Jan 17 '09 at 22:15

asked Jan 17 '09 at 21:07
mmcdole
82.3k ● 60 ● 174 ● 221

11 Answers

Active Oldest Votes

429

When encrypting, you use **their public key** to write a message and they use **their private key** to read it.

When signing, you use **your private key** to write message's signature, and they use **your public key** to check if it's really yours.

> I want to use my private key to generate messages so only I can possibly be the sender.
>
> I want my public key to be used to read the messages and I do not care who reads them

This is *signing*, it is done with your private key.

If you only need to know it to yourself, you don't need to mess with keys to do this. You may just generate random data and keep it in a database.

But if you want people to know that the keys are really yours, you need to generate random data, keep in it a database AND sign it with your key.

> I would like to include my public key in my software to decrypt/read the signature of the key.

You'll probably need to purchase a certificate for your public key from a commercial provider like Verisign or Thawte, so that people may check that no one had forged your software and replaced your public key with theirs.

edited Aug 17 '19 at 5:36
Joshua
360 ● 1 ● 8 ● 17

answered Jan 17 '09 at 21:20
Quassnoi
361k ● 79 ● 555 ● 577

7 Technically speaking, when you say the private key is used to write the message's signature, you're saying that the hash of the message is being encrypted with my private key? – Andy Ibanez Nov 2 '15 at 21:39

4 @AndyIbanez: what is encrypted (the *digest*) may also include timestamp and some random salt, but yes, that's the gist of it. – Quassnoi Nov 2 '15 at 21:53

7 @Quassnoi In fact when we say 'sign with the private key', it means not 'encrypt' but instead means 'decrypt'. Sign the message roughly speaking is the same as decrypt with the private key and in the receiver encrypt with public key, this way the hash will became the same and could be compared. – Johnny Willer Nov 6 '15 at 19:52

5 @JohnnyWiller: as @slim mentioned below, the mathematical core is the same for both encrypt and decrypt functions. They are not separate functions, they are the same function `f(key, message)`, such that `f(private, f(public, message)) === f(public, f(private, message)) === message` – Quassnoi Nov 6 '15 at 21:57

2 @Honey no it's not. Generating the key pair is free. The cost is involved in making the others believe the signature is really yours. To do that, you have your public key itself, which you have earlier generated for free, signed by an authority which might or might not charge you money for that. – Quassnoi Jul 26 '19 at 10:45

141

In RSA crypto, when you generate a key pair, it's completely arbitrary which one you choose to be the public key, and which is the private key. If you encrypt with one, you can decrypt with the other - it works in both directions.

So, it's fairly simple to see how you can encrypt a message with the *receiver's public* key, so that

A signature is proof that the signer has the private key that matches some public key. To do this, it would be enough to encrypt the message with that sender's *private key*, and include the encrypted version alongside the plaintext version. To verify the sender, decrypt the encrypted version, and check that it is the same as the plaintext.

Of course, this means that your message is not secret. Anyone can decrypt it, because the public key is well known. But when they do so, they have proved that the creator of the ciphertext has the corresponding private key.

However, this means doubling the size of your transmission - plaintext and ciphertext together (assuming you want people who aren't interested in verifying the signature, to read the message). So instead, typically a signature is created by creating a *hash* of the plaintext. It's important that fake hashes can't be created, so cryptographic hash algorithms such as SHA-2 are used.

So:

- To generate a signature, make a hash from the plaintext, encrypt it with your private key, include it alongside the plaintext.
- To verify a signature, make a hash from the plaintext, decrypt the signature with the sender's public key, check that both hashes are the same.

edited Mar 19 '19 at 13:36

community wiki
4 revs, 2 users 76%
slim

8 @headcode it *only* applies to asymmetric keys. Symmetric keys don't come in pairs. – slim Jul 4 '15 at 18:06

3 In fact, it only applies to RSA keys. With ECDSA keys, for example, you can trivially generate the public key from the private key, and the private key is a scalar while the public key is a coordinate. – David Schwartz Jun 13 '16 at 16:57

4 How can you decrypt messages with PUBLIC keys? Aren't messages only decrypted with private keys? – daremkd Jul 25 '16 at 10:52

3 5 sources contradicting this answer 1, 2, 3, 4, 5 – wha7ever Aug 10 '17 at 15:03

6 It's not arbitrary which one you call public and private. You can generate the public key from the private key, but you cannot generate the private key from the public key. Isn't that a big difference? – Greg Schmit Apr 3 '18 at 17:10

22

There are two distinct but closely related problems in establishing a secure communication

1. Encrypt data so that only authorized persons can decrypt and read it.
2. Verify the identity/authentication of sender.

Both of these problems can be elegantly solved using public key cryptography.

I. Encryption and decryption of data

Alice wants to send a message to Bob which no one should be able to read

- Bob receives the message and decrypts it using his private Key.

> Note that if A wants to send a message to B, A needs to use the Public key of B (which is publicly available to anyone) and neither public nor private key of A comes into picture here.

So if you want to send a message to me you should know and use my public key which I provide to you and only I will be able to decrypt the message since I am the only one who has access to the corresponding private key.

II. **Verify the identity of sender (Authentication)**

Alice wants to send a message to Bob again. The problem of encrypting the data is solved using the above method.

But what if I am sitting between Alice and Bob, introducing myself as 'Alice' to Bob and sending my own message to Bob instead of forwarding the one sent by Alice. Even though I can not decrypt and read the original message sent by Alice(that requires access to Bob's private key) I am hijacking the entire conversation between them.

Is there a way Bob can confirm that the messages he is receiving are actually sent by Alice?

- Alice signs the message with her private key and sends it over. (In practice, what is signed is a hash of the message, e.g. SHA-256 or SHA-512.)
- Bob receives it and verifies it using Alice's public key. Since Alice's public key successfully verified the message, Bob can conclude that the message has been signed by Alice.

edited Mar 19 '19 at 1:49
Gilles 'SO- stop being evil'
85.1k ● 23 ● 177 ● 215

answered Jan 12 '18 at 21:31
Siva Prakash
3,137 ● 26 ● 22

2 I think this is a very clear answer – Henry Yang Oct 3 '18 at 2:29

1 So when you sign the message, do you sign the actual message itself or the encrypted message? – FrostyStraw Nov 9 '18 at 5:04

20

Yeah think of signing data as giving it your own wax stamp that nobody else has. It is done to achieve **integrity and non-repudiation**. Encryption is so no-one else can see the data. This is done to achieve **confidentiality**. See wikipedia http://en.wikipedia.org/wiki/Information_security#Key_concepts

A signature is a hash of your message signed using your private key.

edited Aug 11 '17 at 17:34
wha7ever
885 ● 1 ● 11 ● 25

answered Jan 17 '09 at 21:19
Johnno Nolan
27.1k ● 17 ● 104 ● 157

16

Encrypting uses the receiver's public key to encrypt the data; decoding is done with their private key.

So, the use of keys is not reversed (otherwise your private key wouldn't be private anymore!).

edited Jan 17 '09 at 22:06
slim
33.5k ● 10 ● 77 ● 106

answered Jan 17 '09 at 21:11
Doug Currie
37.3k ● 83 ● 110

In normal Asymmetric encryption, encryption is done with the recipients public key, not your private key. – mmcdole Jan 17 '09 at 21:16

Did you miss that this question was specifically about RSA where the use of the keys *is* reversed and where it does not compromise the private key. – David Schwartz Jun 13 '16 at 16:56

8

You are describing exactly how and why signing is used in public key cryptography. Note that it's very dangerous to sign (or encrypt) aritrary messages supplied by others - this allows attacks on the algorithms that could compromise your keys.

answered Jan 17 '09 at 21:20
Michael Borgwardt
315k ● 71 ● 449 ● 683

1 Don't web browsers encrypt arbitrary data when using SSL? – Ian Warburton Oct 24 '19 at 22:48

2 @IanWarburton: But they don't use asymmetric encryption for that. The actual data transfers use symmetric encryption with a randomly generated session key. – Michael Borgwardt Oct 25 '19 at 6:55

1 @IanWarburton: no, because neither would be *chosen* by an attacker. The danger lies in encrypting or signing something directly supplied by an attacker, because that can very deliberately crafted to reveal information about your private key, or even create signatures that appear valid for something you did not intend to sign. A detailed breakdown of how the latter case works for RSA is here: crypto.stackexchange.com/questions/35644/… – Michael Borgwardt Oct 25 '19 at 19:28

2 @IanWarburton: that's why RSA requires padding, so you never encrypt only a chose message: en.wikipedia.org/wiki/… - but it's really the operations that involve the private key (like signing) that are especially vulnerable against chosen input. – Michael Borgwardt Oct 26 '19 at 5:26

1 @IanWarburton I'm not a cryptography expert, but from what I understand, that should not cause any problems. – Michael Borgwardt Oct 28 '19 at 22:05

8

Signing indicates you really are the source or vouch for of the object signed. Everyone can read the object, though.

Encrypting means only those with the corresponding private key can read it, but without signing there is no guarantee you are behind the encrypted object.

answered Jan 17 '09 at 23:08
rjamestaylor

4

In your scenario, you do not encrypt in the meaning of asymmetric encryption; I'd rather call it "encode".

So you encode your data into some binary representation, then you sign with your private key. If you cannot verify the signature via your public key, you know that the signed data is not generated with your private key. ("verification" meaning that the unsigned data is not meaningful)

answered Jan 17 '09 at 22:13
devio
34.6k ● 6 ● 71 ● 134

4

> What is the difference between encrypting some data vs signing some data (using RSA)?

Encryption preserves confidentiality of the message ("some data"), while signing provides non-repudiation: i.e. only the entity that signed it could have signed it. There are functional differences as well; read on.

> Does it simply reverse the role of the public-private keys?

Absolutely not. The use of the same private keys for signing and *decryption* (or, likewise, the same public keys for verification and *encryption*) is frowned upon, as you should not mix purposes. This is not so much a mathematical issue (RSA should still be secure), but a problem with *key management*, where e.g. the signing key should have a shorter live and contain more protection before it is used.

For the same message, you should use the senders private key for signing and the receivers trusted public key for encryption. Commonly sign-then-encrypt is used otherwise an adversary could replace the signature with his own. Likewise you should use the private key of the receiver for decryption and the *trusted* public key of the sender for verification.

Furthermore, you should understand that signature generation doesn't use "encryption with the private key". Although all RSA operations are based upon modular exponentiation, the padding scheme is entirely different for signature generation. Furthermore, the public key has entirely different properties than the RSA private key in all practical uses of RSA.

> For example, I want to use my private key to generate messages so only I can possibly be the sender.

That's non-repudiation property, which can be achieved by signing.

> I want my public key to be used to read the messages and I do not care who reads them.

The public key should be considered known by all. If you want everybody to read the messages, then you simply do not encrypt them.

where the appendix is the message. It's a bit weird name as the message is considered more important than the signature over it, but yeah. Only few signatures offer (partial) message recovery; they are not used much anymore and are generally considered deprecated.

Note that signature protocols such as CMS may deploy a *container format* that includes both the message and the signature. In that case you'd need first get the - still unencrypted - message out of the container, much like unzipping a file from a plain .zip archive. So the message may be hidden from view and cannot be directly used in that case.

> I want to be able to encrypt certain information and use it as a product-key for my software. I only care that I am the only one who can generate these.

Encryption is used to achieve confidentiality. In the past RSA signature generation was often thought of as "encryption with the private key". However, the operations are quite different as explained above, and the later standards desperately try and separate encryption and signature generation.

> I would like to include my public key in my software to decrypt/read the signature of the key. I do not care who can read the data in the key, I only care that I am the only verifiable one who can generate them.

Yes, this is called establishing *trust* in the public key. However, protecting your program code is very different from protecting messages. You can perform *code signing* but then you'd need something to check the signature *outside of your code*. There are operating systems that offer this.

There is Microsoft Authenticode for instance. Application stores like the iStore and Android app store may or may not use code signing, but they offer some reassurance that your application isn't cloned or at least not cloned within the store. Cryptography is not always the solution after all.

Keeping your code from being cloned / altered *at all* is much harder, and you'd be solidly in DRM territory if you go that way.

> Is signing useful in this scenario?

Yes, absolutely. It can certainly help making sure that the messages were only signed by you, if there is trust in the public key. If it can be helpful for authenticating your *application code / integrated public key* depends entirely on the environment that you expect to run the code in.

answered Mar 16 '19 at 14:55
Maarten Bodewes
73.2k ● 11 ● 104 ● 202

"In the past RSA signature generation was often thought of as "encryption with the private key". However, the operations are quite different as explained above, and the later standards desperately try and separate encryption and signature generation." - this is not clear to me. Either KeyPair can be the pk or sk and what one encrypts the other can decrypt. If we agree on that, then how can we argue that signing and encrypting

Jan 25 at 0:40

2

Functionally, you use public/private key encryption to make certain only the receiver can read your message. The message is *encrypted* using the public key of the receiver and *decrypted* using the private key of the receiver.

Signing you can use to let the receiver know you created the message and it has not changed during transfer. Message signing is done using your own private key. The receiver can use your public key to check the message has not been tampered.

As for the algorithm used: that involves a one-way function see for example wikipedia. One of the first of such algorithms use large prime-numbers but more one-way functions have been invented since.

Search for 'Bob', 'Alice' and 'Mallory' to find introduction articles on the internet.

edited Jan 16 at 9:28

answered Jan 17 '09 at 21:16
Gerbrand
1,018 ● 7 ● 19

Can you comment on my use case? I want to use a private key to encrypt and a public key to allow anyone and everyone to decrypt. – mmcdole Jan 17 '09 at 21:17

1 It's not true that all asymmetric encryption is based on prime numbers, it's just the most well-known example (RSA); there are other methods such as elliptic curve cryptography. – Michael Borgwardt Jan 17 '09 at 21:18

" The message is encrypted then encrypted using the public key of the receiver." That sentence just doesn't make any sense, or at the very least it needs further explanation. "encrypted then encrypted"??? – Maarten Bodewes Dec 16 '19 at 1:20

1

Answering this question in the content that the questioners intent was to use the solution for software licensing, the requirements are:

1. No 3rd party can produce a license key from decompiling the app
2. The content of the software key does not need to be secure
3. Software key is not human readable

A Digital Signature will solve this issue as the raw data that makes the key can be signed with a private key which makes it not human readable but could be decoded if reverse engineered. But the private key is safe which means no one will be able to make licenses for your software (which is the point).

Remember you can not prevent a skilled person from removing the software locks on your product. So if they have to hack each version that is released. But you really don't want them to be able to generate new keys for your product that can be shared for all versions.

and of cause NaCl project to C examples

answered Mar 10 '16 at 11:38

oden

2,716 ● 1 ● 26 ● 27