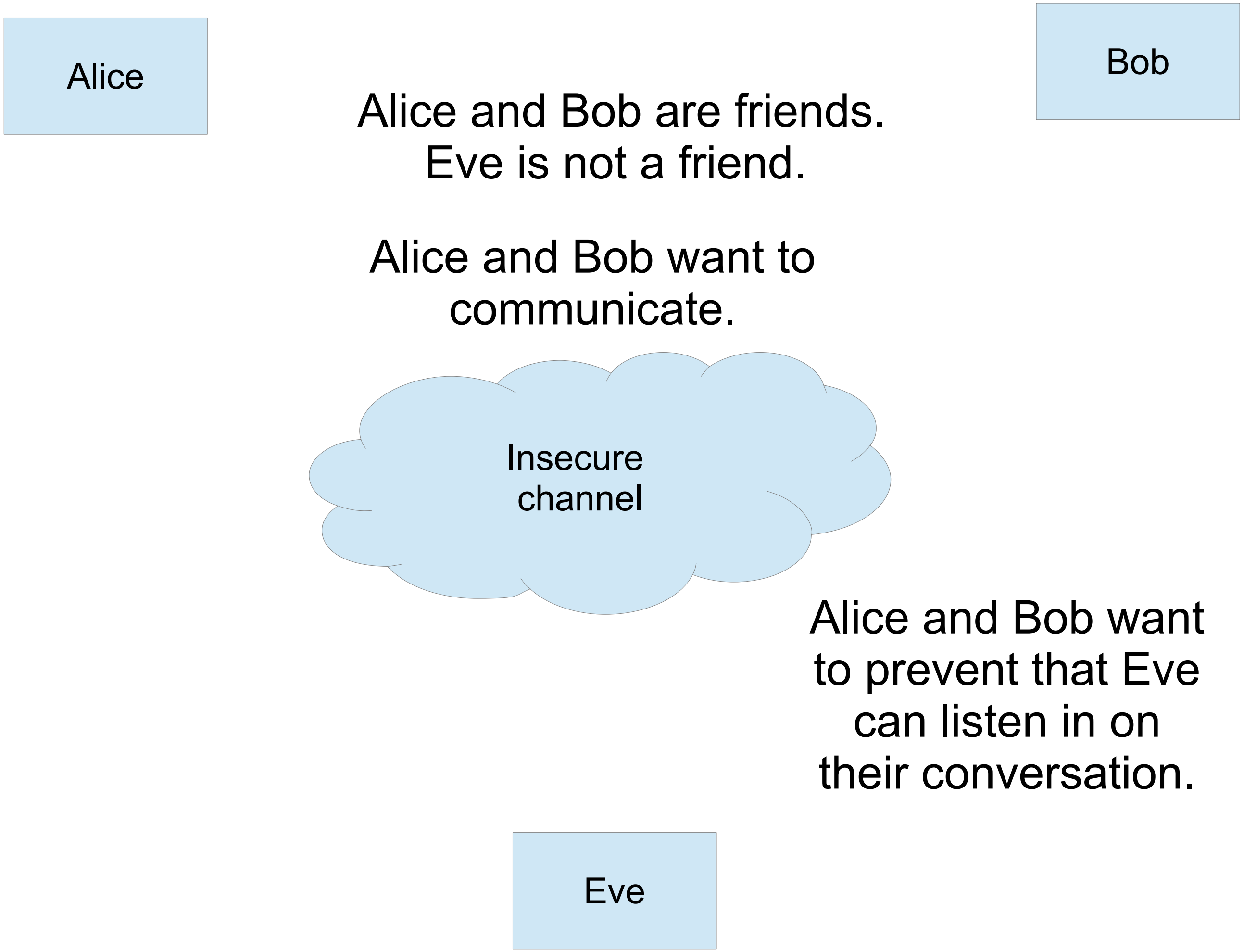
RSA
for fun

This video tries to portray an example of Alice and Bob and how they manage to communicate with asymmetric and symmetric encryption without Eve messing up their lives.

Secret-key



Alice and Bob are friends.
Eve is not a friend.

Alice and Bob want to communicate.

Alice and Bob want to prevent that Eve can listen in on their conversation.

Opening

So Alice wants to hide information from Eve. But she does want Bob to be able to read what she wants to let him know.

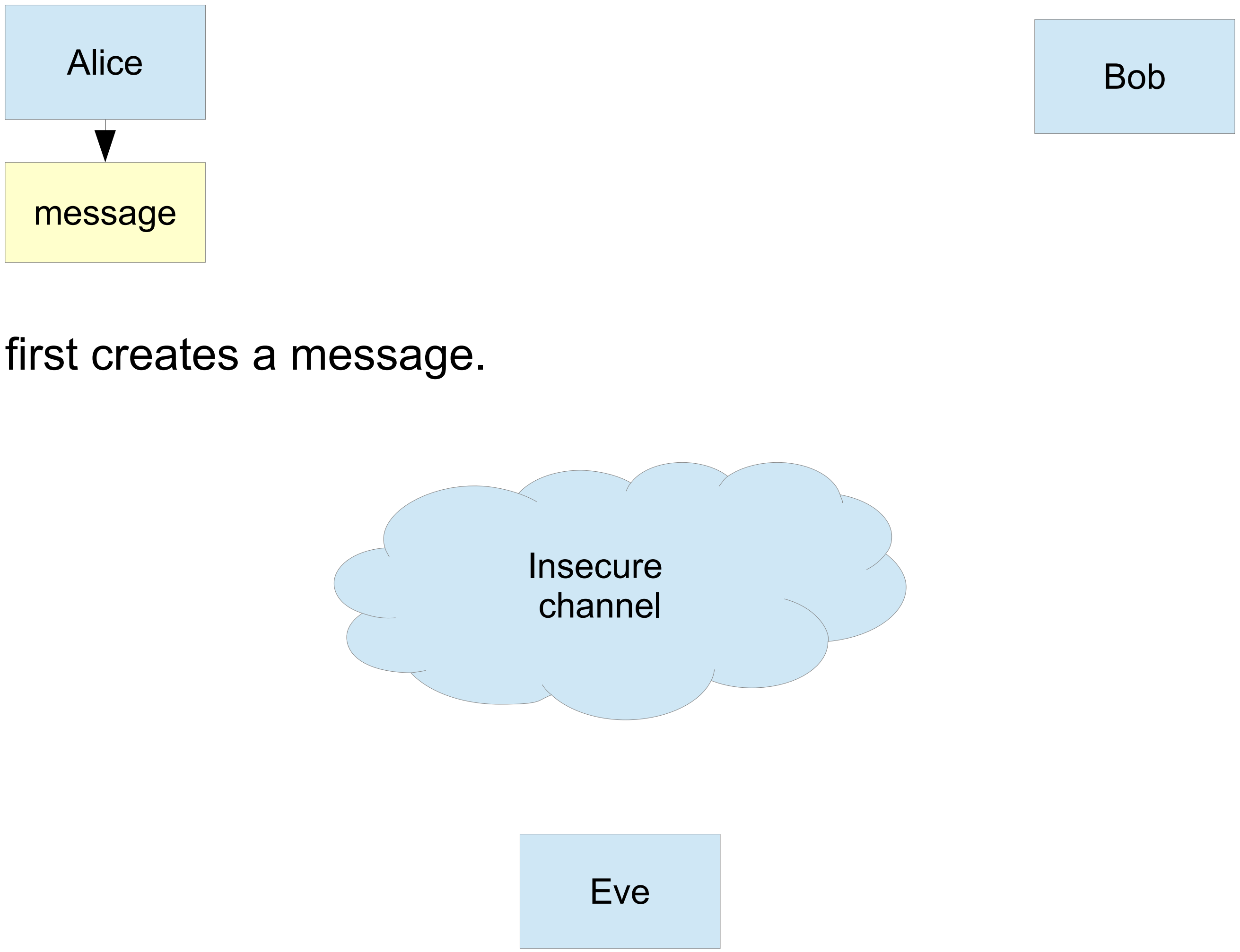
Commonly, messages are encrypted with a symmetric-key (secret-key), e.g. AES. *(Advanced Encryption Standard)*.

Alice will encrypt her message with the secret-key, and Bob will decrypt the message with the same secret-key.

So Alice and Bob need the **same** secret-key.

Like this:

secret-key



Alice

message

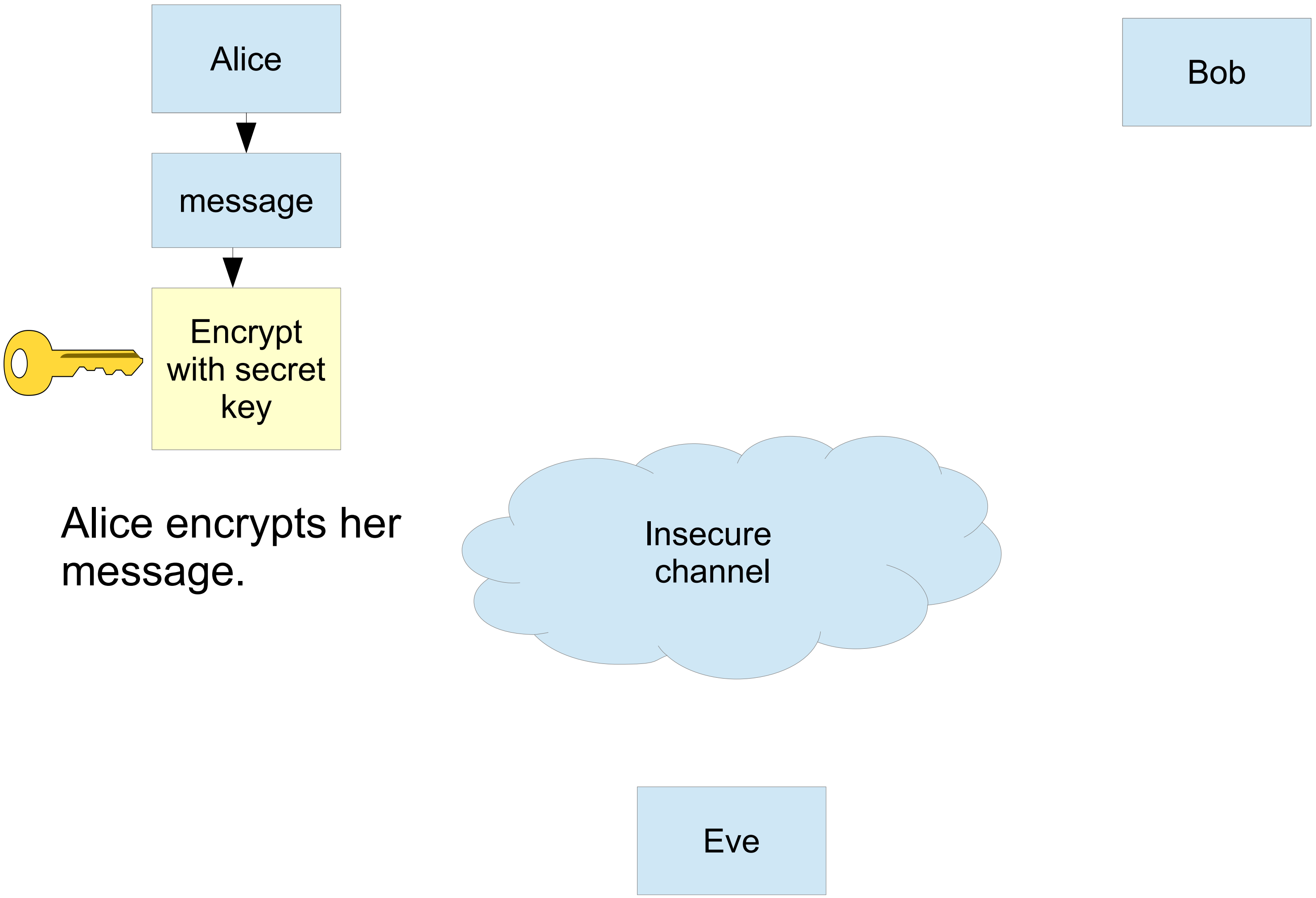
Bob

Alice first creates a message.

Insecure channel

Eve

secret-key



Alice

message

Encrypt with secret key

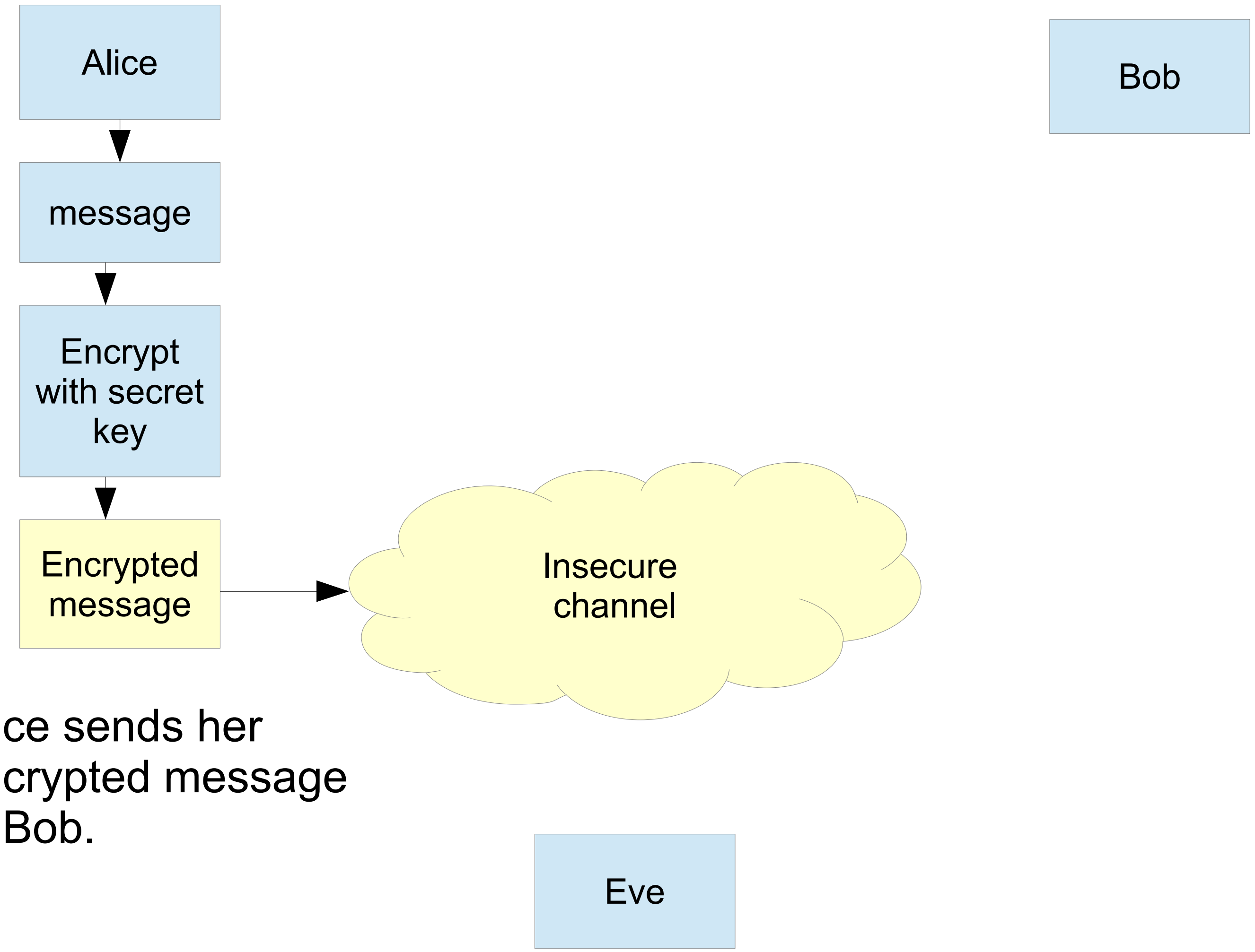
Insecure channel

Bob

Eve

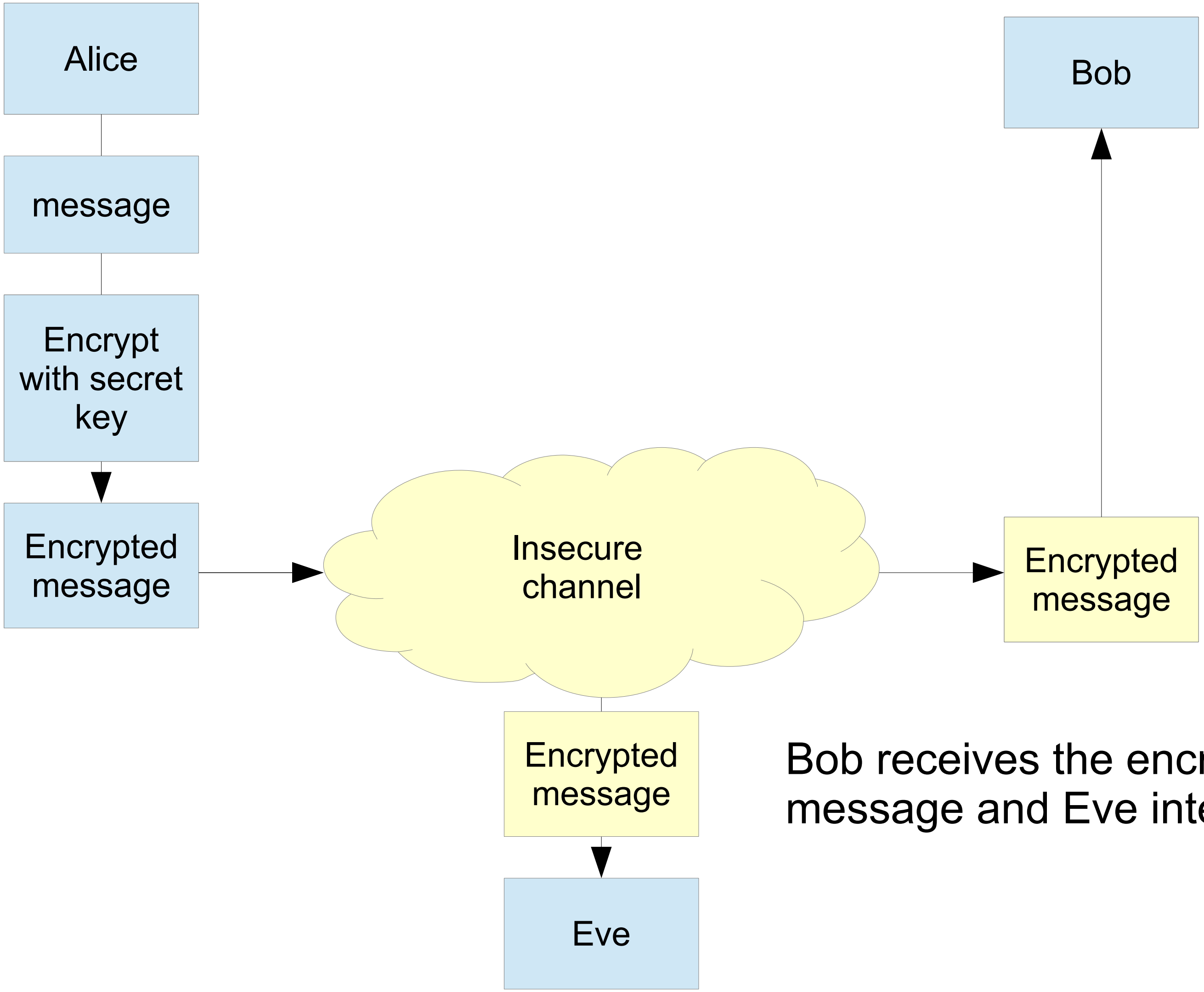
Alice encrypts her message.

secret-key



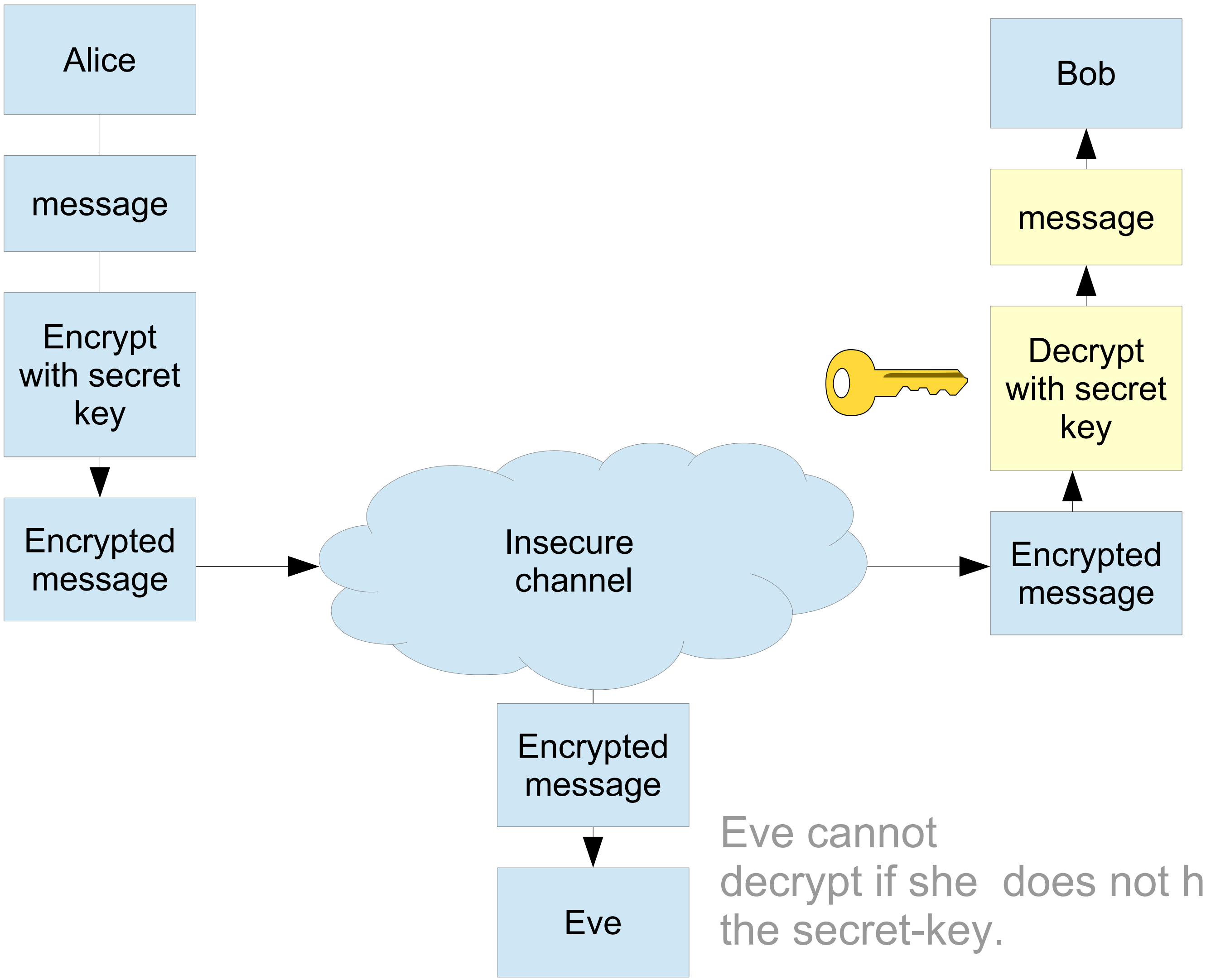
Alice sends her encrypted message to Bob.

secret-key



Bob receives the encrypted message and Eve intercepts it.

secret-key



Alice

message

Encrypt with secret key

Encrypted message

Insecure channel

Encrypted message

Eve

Bob

message

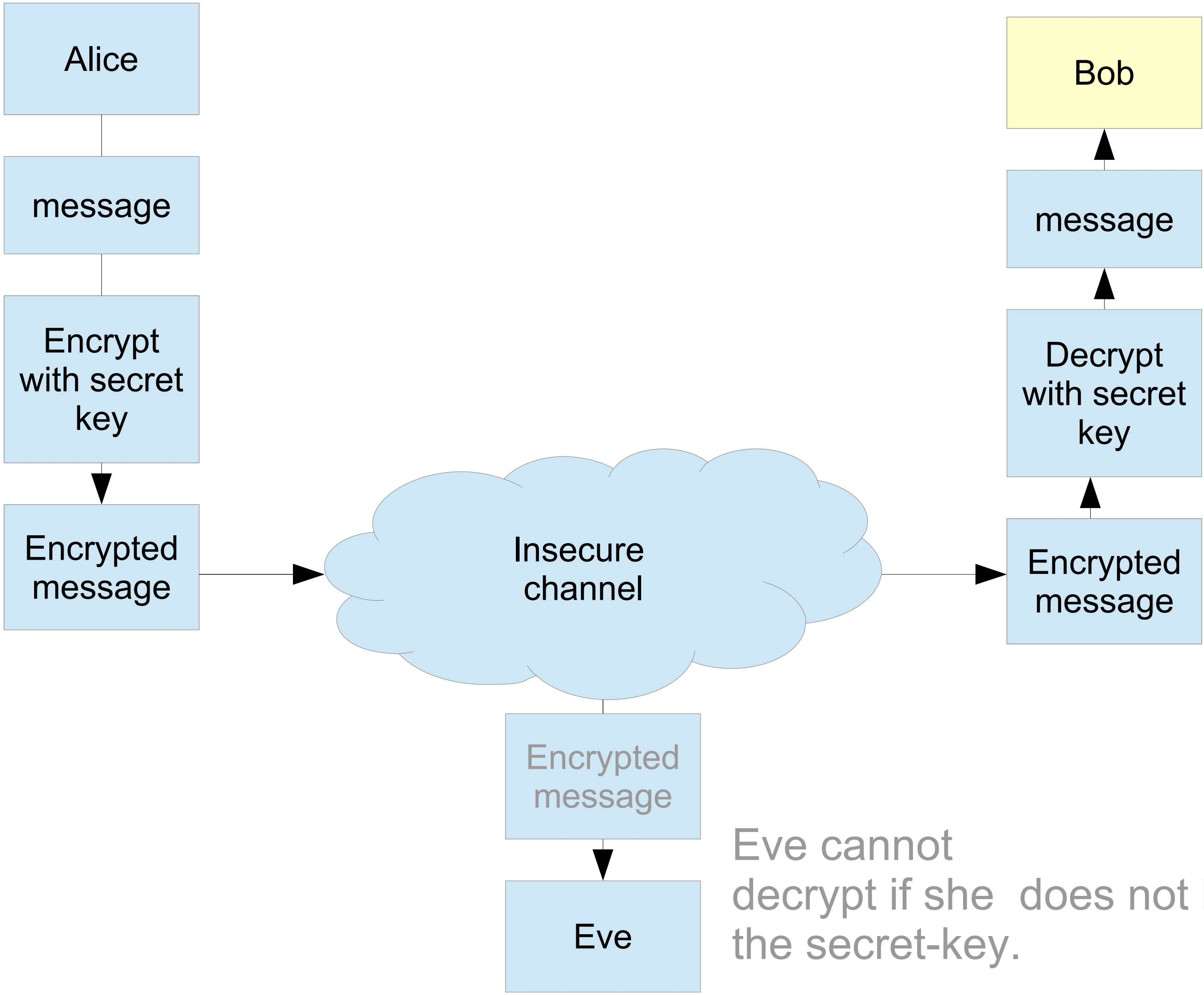
Decrypt with secret key

Encrypted message

Bob decrypts the message with the secret key.

Eve cannot
decrypt if she does not have
the secret-key.

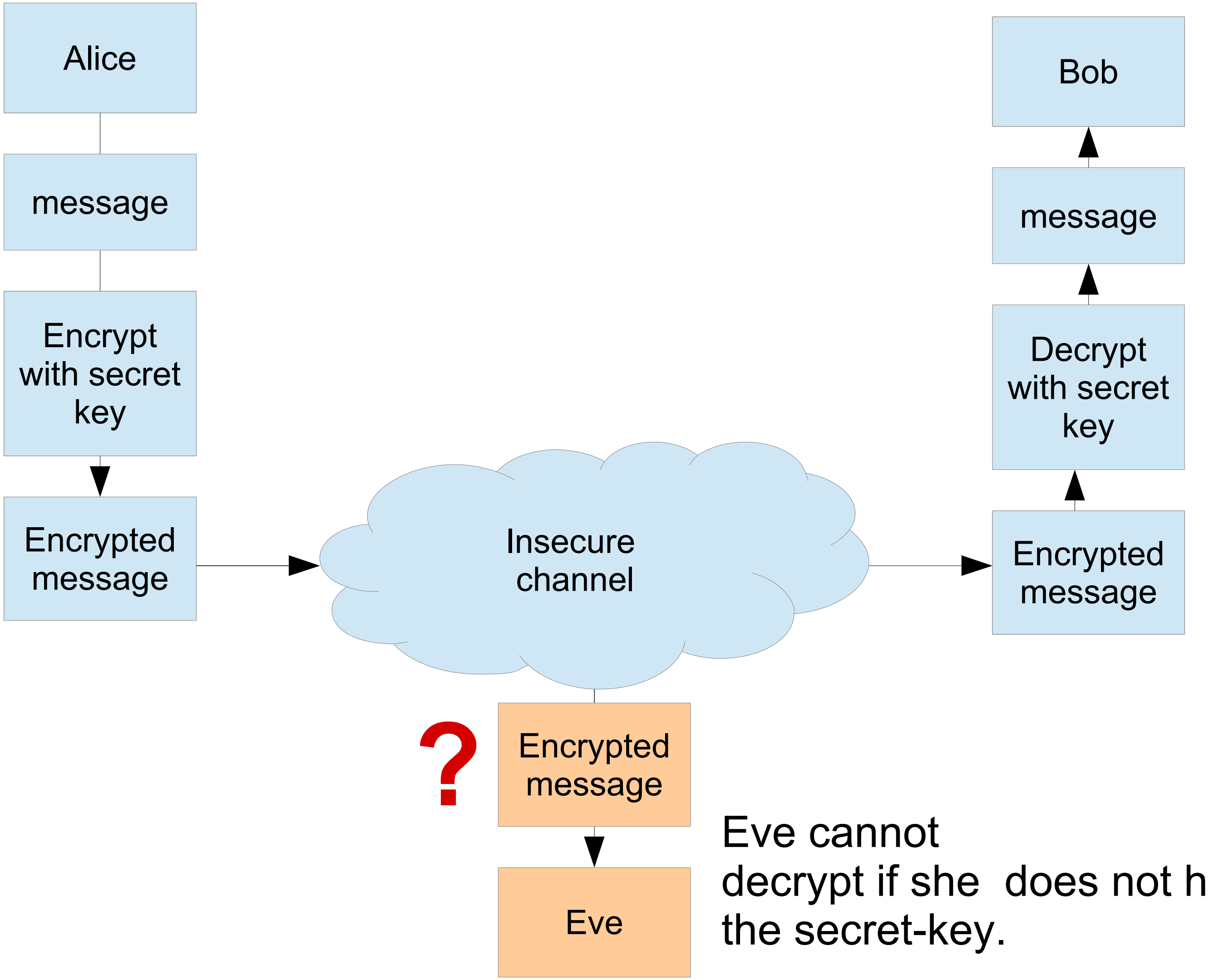
secret-key



Bob reads the message.

Eve cannot decrypt if she does not have the secret-key.

secret-key



Eve cannot
decrypt if she does not have
the secret-key.

The problem

The problem Alice faces is how to get
the secret-key
to Bob over the insecure channel.

Eve is always listening in.

Like this:

The problem

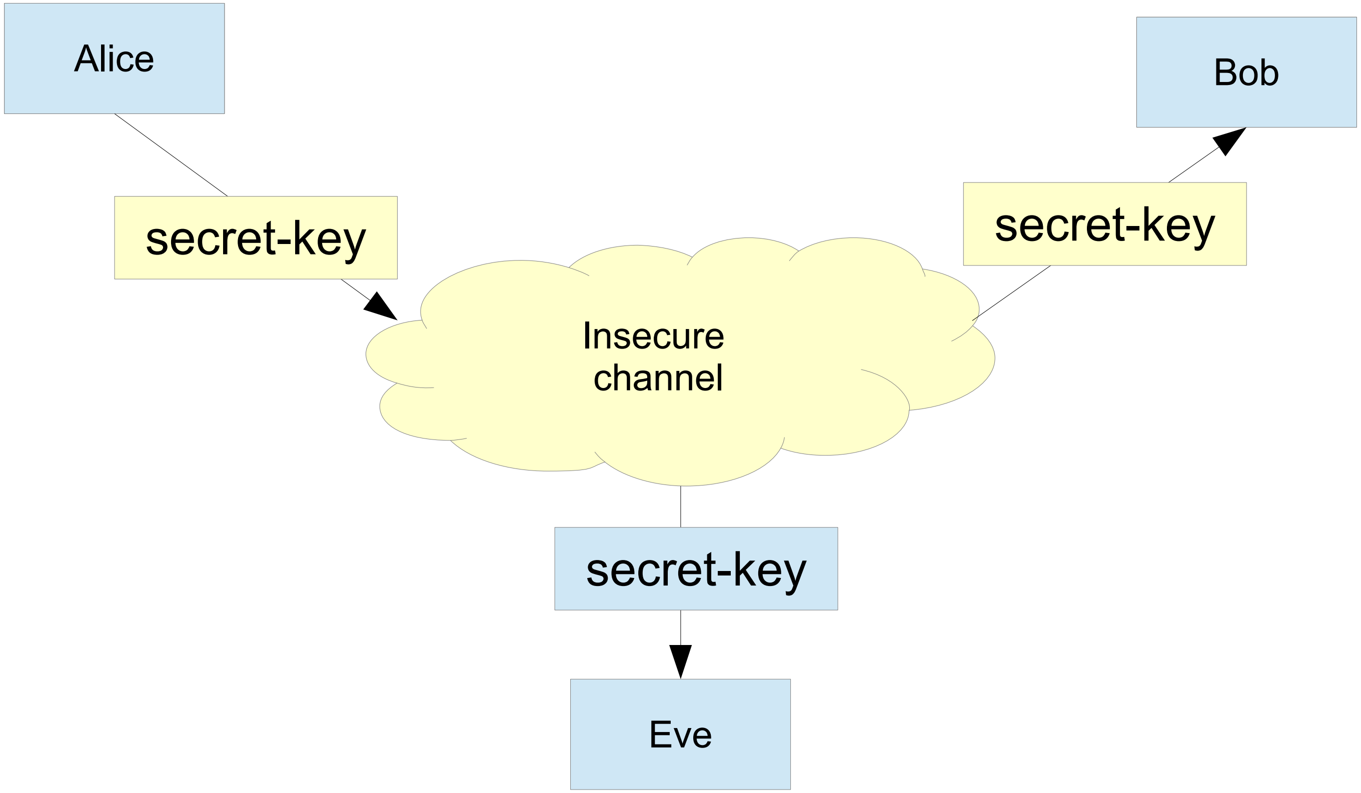
The problem Alice faces is how to get
the secret-key
to Bob over the insecure channel.

Eve is always listening in.

Like this:

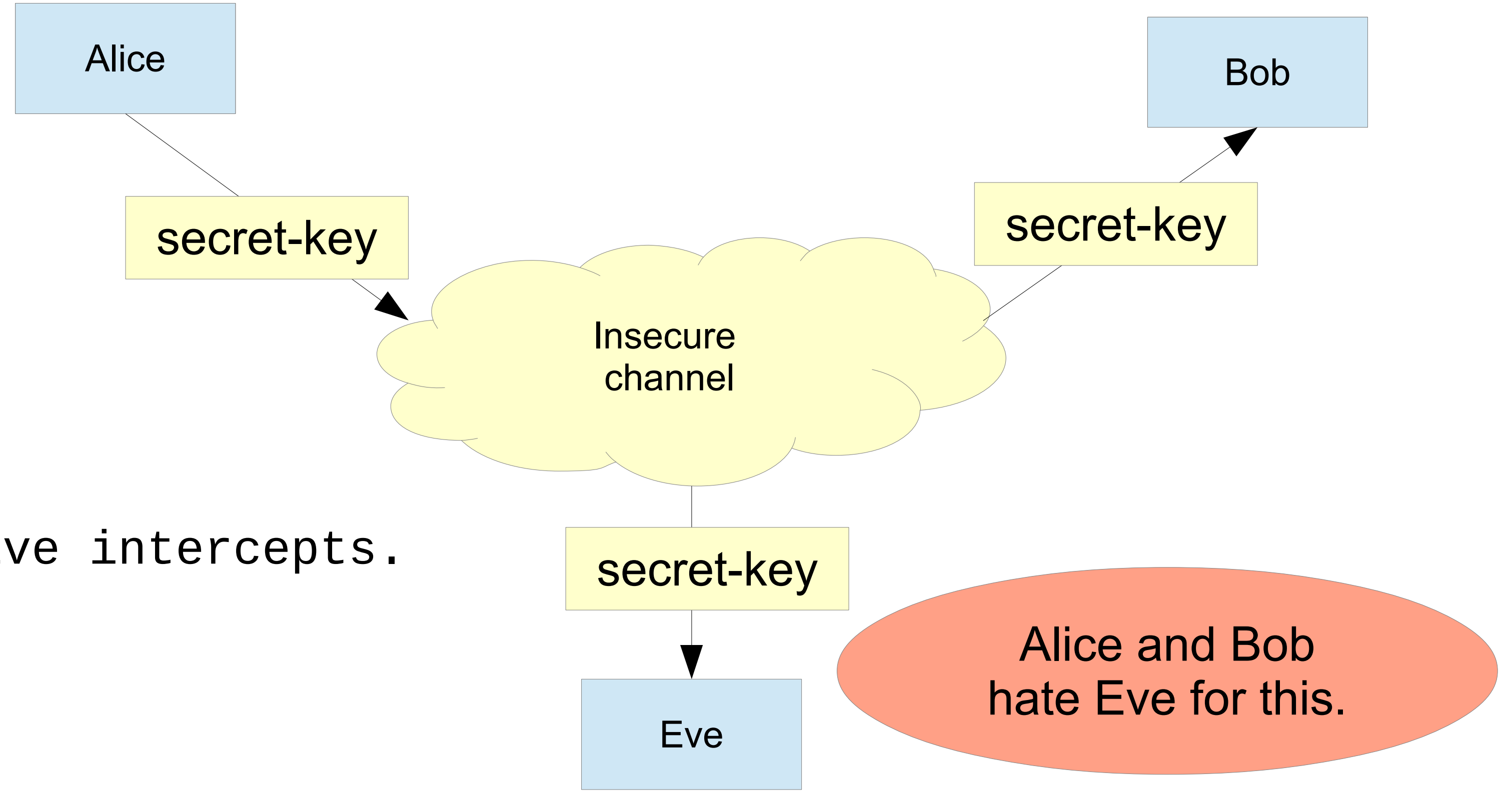
Eve misbehaves (as usual)

```
Alice sends the secret-key she want to use
for encryption, to Bob.
```



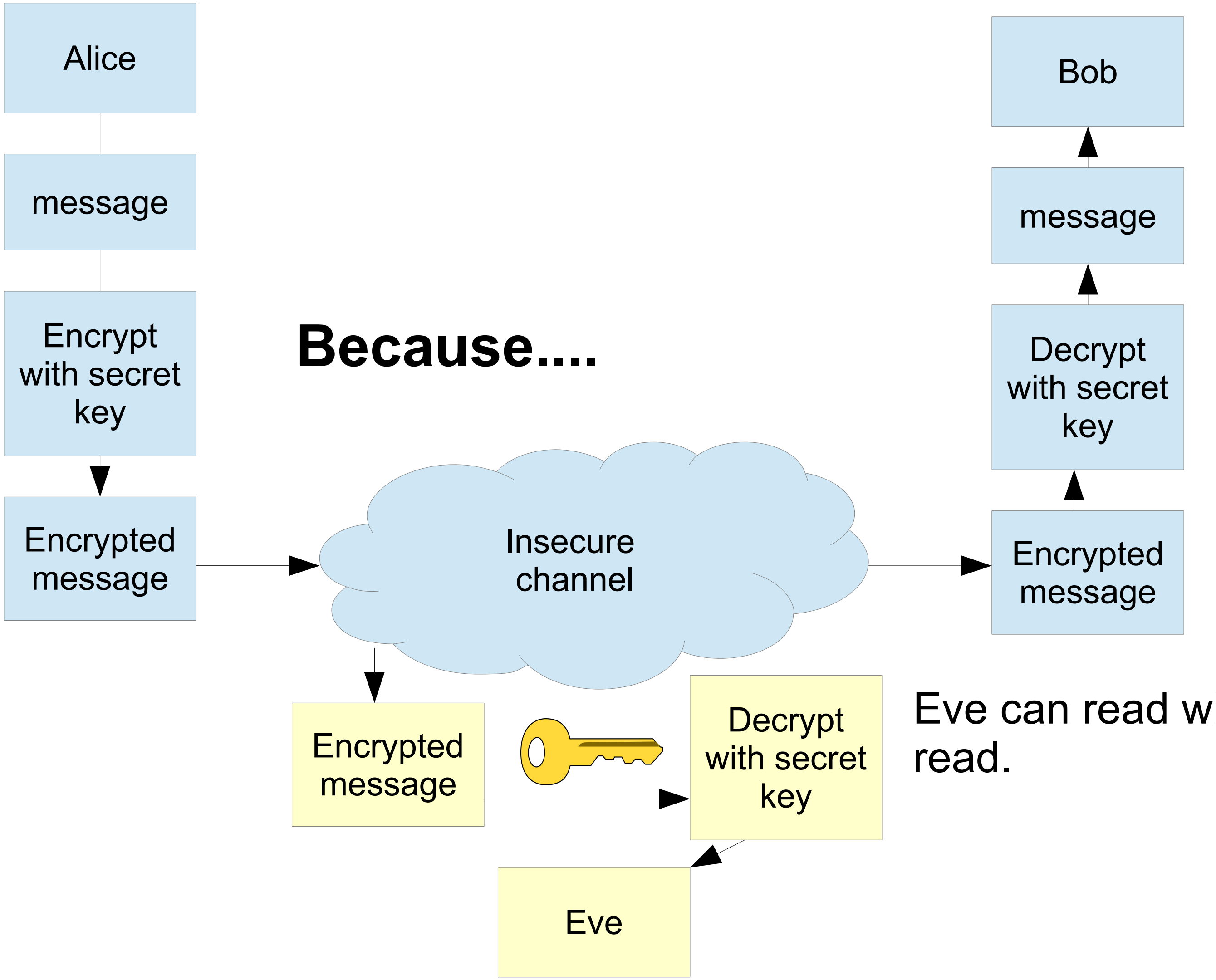
Eve misbehaves (as usual)

Alice sends the secret-key she want to use for encryption, to Bob.



Eve intercepts.

secret-key



Because....

Eve can read what Bob can read.

Asymmetric encryption

In comes asymmetric encryption. Very often RSA* is used to create a key-pair: a Public-key and a Private-key. How does this work?

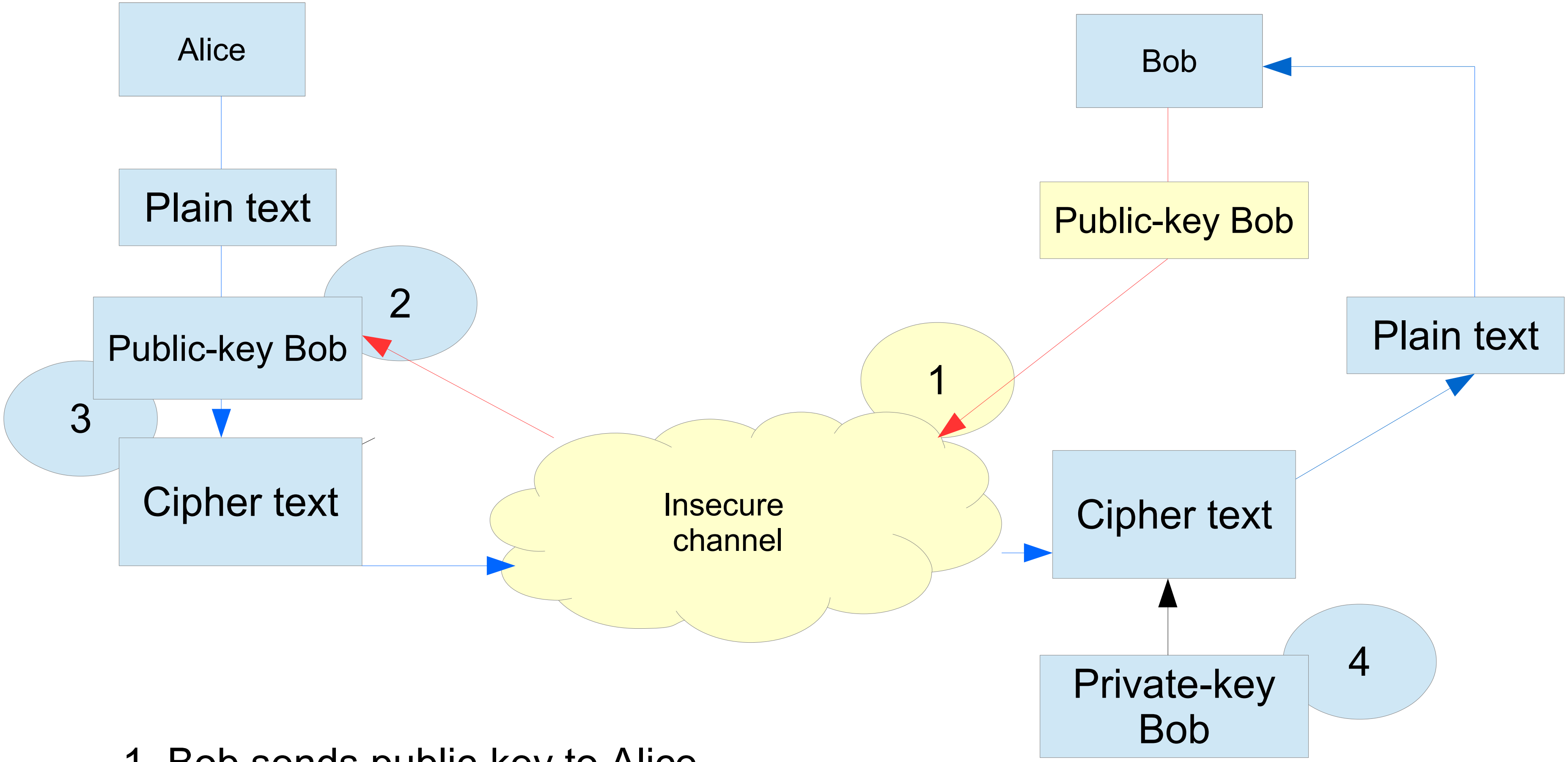
*(There are other Public-key algorithms; think of Diffie Hellman)

This is how that works

Bob creates a **key-pair.** He then sends his Public-key to Alice
and maybe others who also want to send him
encrypted messages. Anybody (Alice in this case)
will encrypt a message with Bob's Public-key, before
sending the message to Bob.
On arrival, Bob will decrypt the message with
his Private-key.
The message is often referred to as '**plain text**', the encrypted message is often referred to as '**cipher text**'.

Something like this:

Key-pair



1. Bob sends public-key to Alice
2. Alice receives public-key
3. Alice encrypts message with Bob's public key and sends
4. Bob receives message and decrypts with his private key

Summary for now (1).

Because the Public-key and Private-key are mathematically Related (yes they are), the only person that is able to decrypt the message that Alice sends, is the person with Bob's Private-key. It is obvious that Bob is the owner of his Private-key and he will **NOT** share this key with others.

Summary for now (2).

Let's have a look at a possible scenario:

1. Alice generates a secret-key with **AES**.
2. Bob generates an **RSA key-pair**.
3. Bob sends his **Public-key** to Alice.
4. Alice uses Bob's **Public-key** to encrypt the **(AES)secret-key**.
5. Alice sends the encrypted **message(cipher-text)** to Bob.
6. Bob decrypts the cipher-text(secret-key) with his **Private-key**.
7. Now Bob and Alice have the same **(AES)secret-key**.
8. Alice encrypts with the **secret-key** and sends her message.
9. Bob receives and decrypts the data with the **same secret-key**.

This is called 'hybrid encryption' (symmetric and asymmetric)

The fun...

Bob generates a Key-pair.

1. Bob chooses two (very large) prime numbers -> **p** and **q**.

(In this example, small number are used to make this understandable)

2. Bob multiplies the two prime numbers **p * q** to generate **n**.

- **p=11 q=17 n=p*q**
- **11 * 17 = 187**

This product (**n=187**) is going to be part of Bob's Public-key **and** Private-key.

So **n** is not secret. Although it is also going to be part of the Private-key.

Euler's theorem

3. Bob creates a *totient*. This totient (Euler's totient function or **phi**) lists all positive integers up to a given integer **n** that are relatively prime to **n**.

A number relatively prime to another number is a number that does not share a common factor with that other number.

For example: 13 is a prime number. The numbers from 1 to 12 share no common divisor with 13, so **phi** of 13 is 12. To give us **ph**i of the two prime numbers (**p** and **q**), we take the product of **phi** of **p** **(p-1)** and **phi** of **q** **(q-1)**. The symbol commonly used for phi is

is **(p-1) * (q-1).**

is **10 * 16 = 160.**

e

4.Bob chooses the second part of his public key
(remember: **n** was the first part).
The second part of his public key is another prime number.
We call it '**e**'.

This prime number should be less than phi and should not share a common factor with phi. In other words: e is *relatively prime* to phi. We try prime 13.

With small numbers we can easily determine that 13 does not share a common divisor with 160(phi).

160/13=12, 12*13=156.
The remainder is 4.
13 can only be divided by 1 and by itself, and the remainder of 160/13 is uneaqual to 0.
So it is safe to say that 160 and 13 share no common divisor other than 1.

Euclidean algorithm (1)

With larger numbers, however, this is harder to determine. To actually make sure that e is relatively prime to phi, we use the Euclidean algorithm.

Goes like this:

Euclidean algorithm (2)

In our example, **phi** is 160 and **e** is 13. Euclidean's algorithm says this:

160/13=12
remainder is 4
(we shift the divisor (13) and the remainder (4) to the left)

13/4=3
remainder is 1.

The greatest common divisor of 160 and 13 is 1. So, 160 and 13 are relatively prime.

In short: if a division gives a remainder of zero (0), then the two numbers are **not** relatively prime.

Euclidean algorithm (3)

Another example: phi=160 and e=5

160/5=32

remainder is 0

The greatest common divisor of 160 and 5 is **5**.

160 and 5 are **not** relatively prime.

So we better stick with 13.

Euclidean algorithm (3)

Another example: phi=160 and e=5

160/5=32

remainder is 0

The greatest common divisor of 160 and 5 is **5**.

160 and 5 are **not** relatively prime.

So we better stick with 13.

Pause.

Let's pause for a minute and see what we've got:

```
p=11 q=17 n=187
phi=160   e=13
```

Bob's Public-key is e and n.

So, Bob's Public-key is 13,187.

What is public and what is not?

p, q and phi are private.
e and n are public.

So: ***p q*** *and* ***phi*** *are highly classified.*
Bob can share ***e*** *and* ***n*** *with the world, also with Eve.*

Encrypting with public-key

If Alice wants to encrypt her message she will use the following formula:

Taken that message is M and encrypted message is C.

C=M to the power of e mod n.

C=M^e(mod n).

If the message is 8, then the C is 8^13(mod 187)

8^13=**54975581388**8

549755813888(mod 187)=**94**

C=**94**

Bob still has to generate his private key; The key that is needed to decrypt Alice's message. The Private-key is the **inverse of e**.

Tricky bit

So, the Private-key is: (d,n).

Here comes the tricky bit: what is d?

Because the Public-key and the Private-key must be mathematically related, we have to use the Public-key in the computation. We were free to choose the Public-key (remember: the Public-key is freely chosen, condition is that the prime (e) and phi are relatively prime).

In order to compute d, we have to solve the following problem:

e * d   1 (mod phi)

13 * ?   1 (mod 160)

What is ?, or, what is the second part of Bob's private Key?

Tricky bit

So, the Private-key is: (d,n).

Here comes the tricky bit: what is d?

Because the Public-key and the Private-key must be mathematically related, we have to use the Public-key in the computation. We were free to choose the Public-key (remember: the Public-key is freely chosen, condition is that the prime (e) and phi are relatively prime).

In order to compute d, we have to solve the following problem:

e * d   1 (mod phi)

13 * ?   1 (mod 160)

What is ?, or, what is the second part of Bob's private Key?

Extended Euclidean

We used the Euclidean algorithm to determine that **e** and **phi** were relatively prime. We will use the extended Euclidean algorithm to compute **d**.

There are multiple ways of doing this. We will start with the easiest, then we will try a (somewhat) more challenging way. Both should give us the same result for **d**.

Again: the first method hardly needs any thinking.

easy

We place phi in both columns.

Then we place **e** and **1** in the left and right column.

Simply follow the math and don't think.

160	160
13	1

We divide:
160/13=12
We multiply:
12*13 = 156 and 12*1 = 12
We subtract:
160-156 = **4** and 160-12 = **148**
We place the results in the
two columns.

easy

We place phi in both columns.

Then we place **e** and **1** in the left and right column.

Simply follow the math and don't think.

160	160
13	1

We divide:
160/13=12
We multiply:
12*13 = 156 and 12*1 = 12
We subtract:
160-156 = **4** and 160-12 = **148**
We place the results in the
two columns.

easy

We place phi in both columns.

Then we place **e** and **1** in the left and right column.

Simply follow the math and don't think.

160	160
13	1
4	**148**

We divide:
160/13=12
We multiply:
12*13 = 156 and 12*1 = 12
We subtract:
160-156 = 4 and 160-12 = 148
We place the results in the two columns.

We repeat the procedure
but we use the bottom two lines:

160	160
13	1
4	148

We divide:
13/4=3
We multiply:
3*4=12 and 3*148 = 444
We subtract:
13-12=**1** and 1-444 = **-443**

Alert: we cannot use a
negative number so we use
phi to get apositive number:
We mod:
-443 (mod 160) = **37**

We repeat the procedure
but we use the bottom two lines:

160	160
13	1
4	148

We divide:
13/4=3
We multiply:
3*4=12 and 3*148 = 444
We subtract:
13-12=**1** and 1-444 = **-443**

Alert: we cannot use a
negative number so we use
phi to get apositive number:
We mod:
-443 (mod 160) = **37**

We repeat the procedure
but we use the bottom two lines:

160	160
13	1
4	148

We divide:
13/4=3
We multiply:
3*4=12 and 3*148 = 444
We subtract:
13-12=**1** and 1-444 = **-443**

Alert: we cannot use a negative number so we use phi to get apositive number:
We mod:
-443 (mod 160) = **37**

We repeat the procedure
but we use the bottom two lines:

160	160
13	1
4	148

We divide:
13/4=3
We multiply:
3*4=12 and 3*148 = 444
We subtract:
13-12=1 and 1-444 = -443

Alert: we cannot use a
negative number so we use
phi to get apositive number:
We mod:
-443 (mod 160) = 37

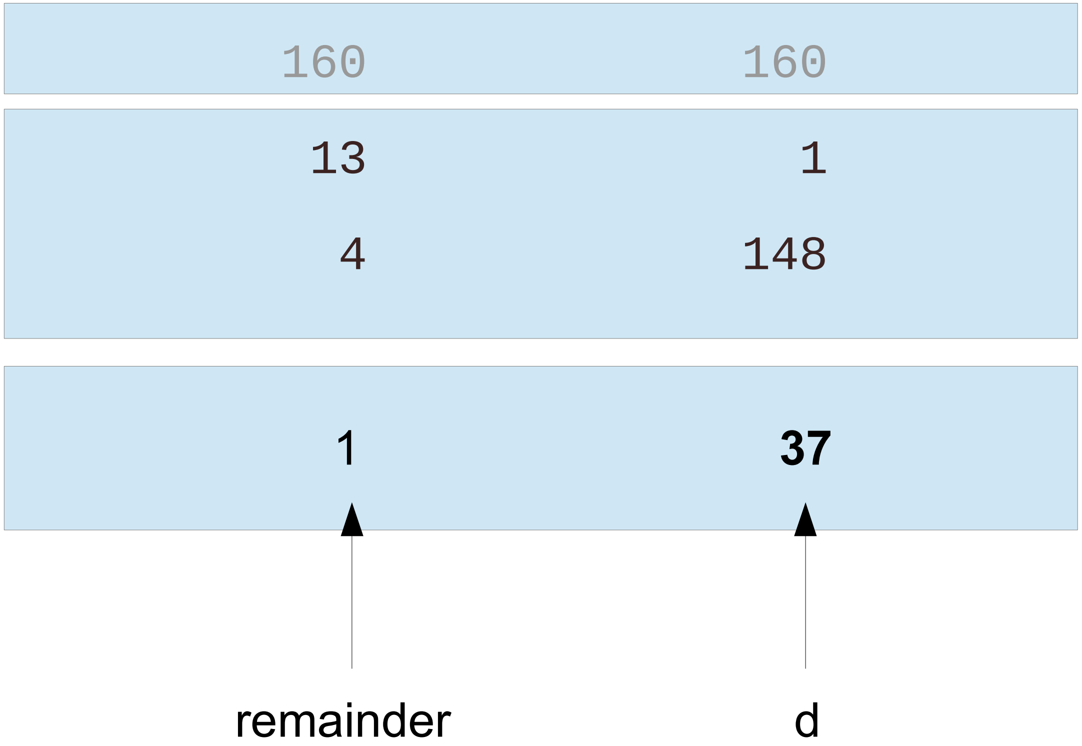
We repeat the procedure
but we use the bottom two lines:

160	160
13	1
4	148

We divide:
13/4=3
We multiply:
3*4=12 and 3*148 = 444
We subtract:
13-12=1 and 1-444 = -443

Alert: we cannot use a negative number so we use phi to get a positive number.
We mod:
-443 (mod 160) = 37

We repeat the procedure
but we use the bottom two lines:



160	160
13	1
4	148
1	**37**

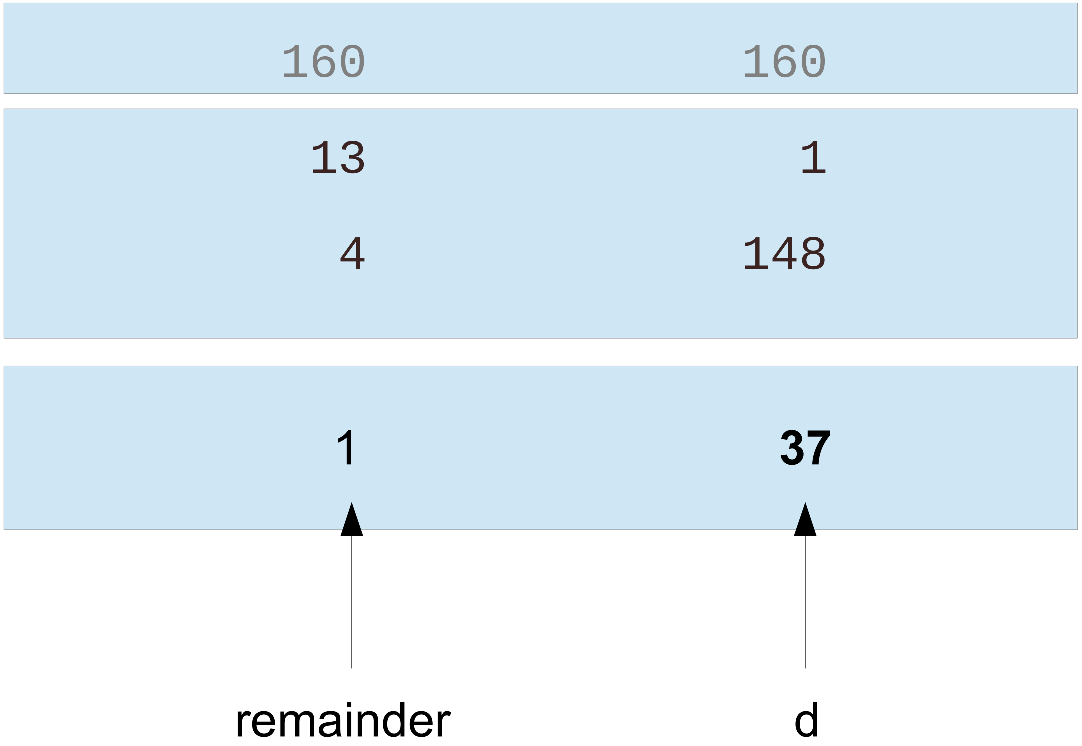
13 * 37   1 (mod 160)

We place 1 and 37 in column 1 and column 2.

Stop: we have found the second part of the private key.

The second part of Bob's private key is **37**. So Bob's private key is (d,n), (**37**,**187**)

We repeat the procedure
but we use the bottom two lines:



160	160
13	1
4	148
1	**37**

13 * 37   1 (mod 160)

We place 1 and 37 in column 1 and column 2.

Stop: we have found the second part of the private key.

The second part of Bob's private key is **37**. So Bob's private key is (d,n), (**37,187**)

Another method

Before we go into a 'real live' encryption example we will try a second method for the same problem. This requires a little more thinking, but not much.

We first use the Euclidean algorithm find the greatest common divisor,and then we use the Extended Euclidean Algorithm with 'like terms' to find d.

Like terms means that you do not use the results, but the equasions as such in the equasion that gives you the remainder of **mod 160**. In the end that will give you d.

Euclidean again.

First we find the gcd, like we did before.

160/13=12

160=12(13)+4

13/4=3

13=3(4)+1

The remainder is 1 so we can stop. 1 is the greatest common divisor.

Extended Euclidean

Then we start with the last equasion and reverse the process. You insert the equasion that gave you the result. So in fact, you replace the result with the equasion itself.

1=13-3(4) can be rewritten as 1=13-3(160-12(13))

Finally, we transform this equasion and we are done. Mind you. We've got -3 times 160 and 1 times 13 and -3 times -12 times 13. That will give us -3 times 160 and +37*13, because negative times negative is positive.

1=-3(160)+**37**(13) We have found d: **37**

e * d = 1 (mod phi)

13 * 37   1 (mod 160)

So Bob's Private-key is (d, n) (37, 187)

Exchange the symmetric (secret-key) by using asymmetric (public-key) encryption.

Finally, the real live example.

As we have seen, Alice encrypted her message(secret-key) with Bob's Public-key: M^e(mod n)

If the message is 8, then C is 8^13(mod 187)

8^13=549755813888

549755813888(mod 187)=94

C=**94**

Then she sent the message to Bob. Bob now decrypts as follows:

M=C to the power of d mod n

M=C^d(mod n)

M=**94**^**37**(mod **187**)

M=**8**

Bob just received Alice's message(secret-key). Now they can communicate safely with a shared(secret-key).