FHV
Vorarlberg University
of Applied Sciences

Application Integration and Security

Philipp Scambor
Valmir Bekiri

Learning outcomes and Methodology

- Learning outcomes
  - Basics of Cyber Security
  - Basics of Cryptography

- Methodology
  - Lecture
  - Exercises – 17.04

Agenda

- Security Goals
- Examples of bad security
- Cryptology/Cryptography
- SSL/TLS

Literature

- Paulus, Sachar (2011): Basiswissen Sichere Software: Aus- und Weiterbildung zum ISSECO Certified Professional for Secure Software Engineering.
  - https://vlb-katalog.vorarlberg.at/F?local_base=fhb01&func=find-c&ccl_term=SYS=000063110

- Basin D., Schaller P., Schläpfer M.(2011): Applied Information Security
  - https://vlb-katalog.vorarlberg.at/F?local_base=fhb01&func=find-c&ccl_term=SYS=000065729

Security Goals

- To secure a system/software you have to know the goals that you want to achieve
- Different goals requirement different actions
- Goals:
  - **C**onfidentiality
  - **I**ntegrity
  - Authenticity
  - Non-Repudiation
  - **A**vailability
  - Accountability
  - Access Control

Security Goals

- **C**onfidentiality
  - Basically privacy
  - Only authorized people should be able to read the given data
- **I**ntegrity
  - Data should be not manipulated
  - Data should be trustworthy
- **A**vailability
  - The system/software/data should be available all the time
  - Data should not be lost

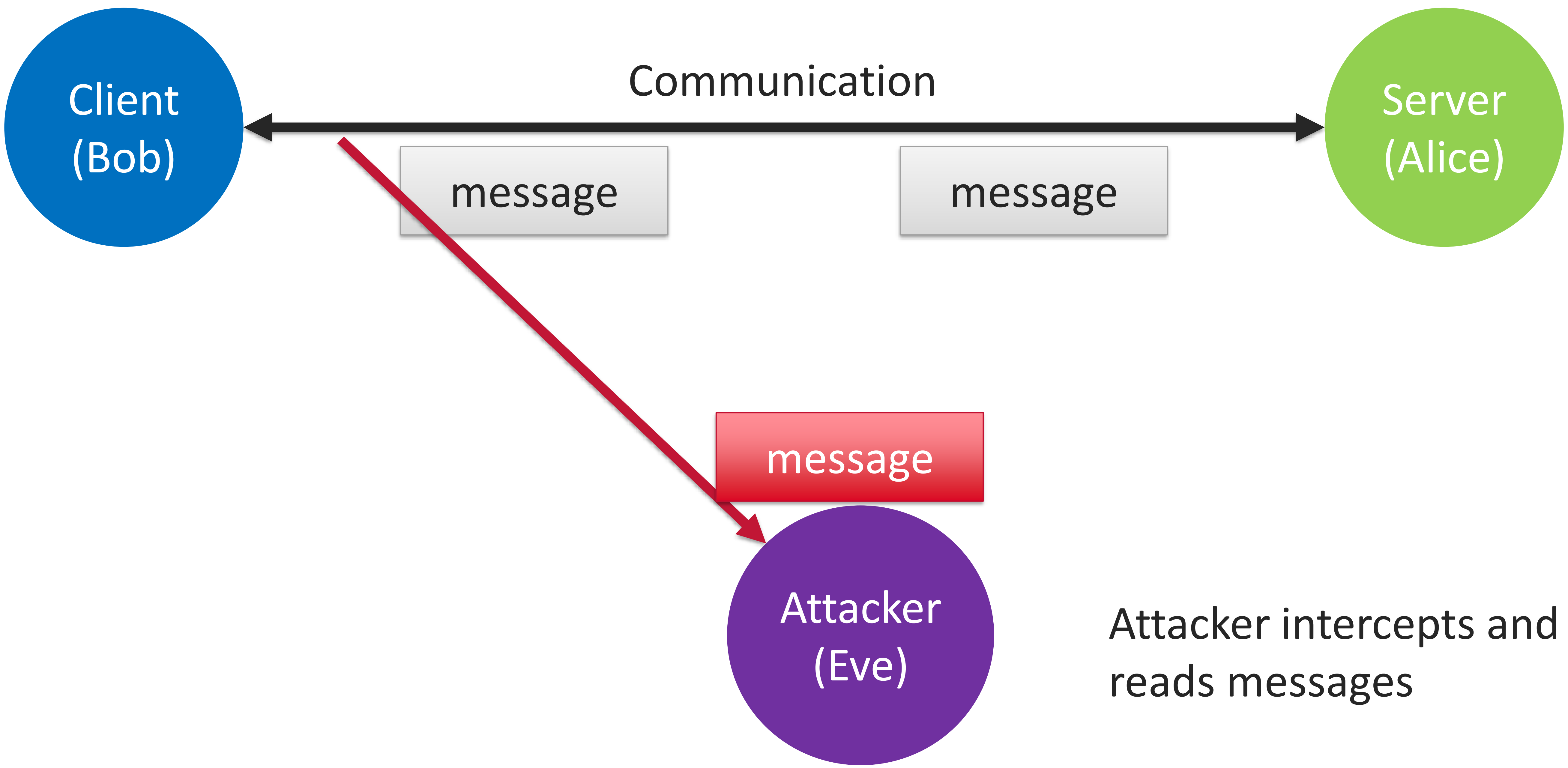
- Those are the three main goals of Cyber Security – often called CIA Triad

Security Goals

- Authenticity
  - The data or the sender of the data should be authentic
- Non-Repudiation
  - Taken action are provable and not deniable
- Accountability
  - Everything should be tracked and logged
- Access Control (Authorization)
  - Actions (on data) can only be done by authorized subjects

Security Goals

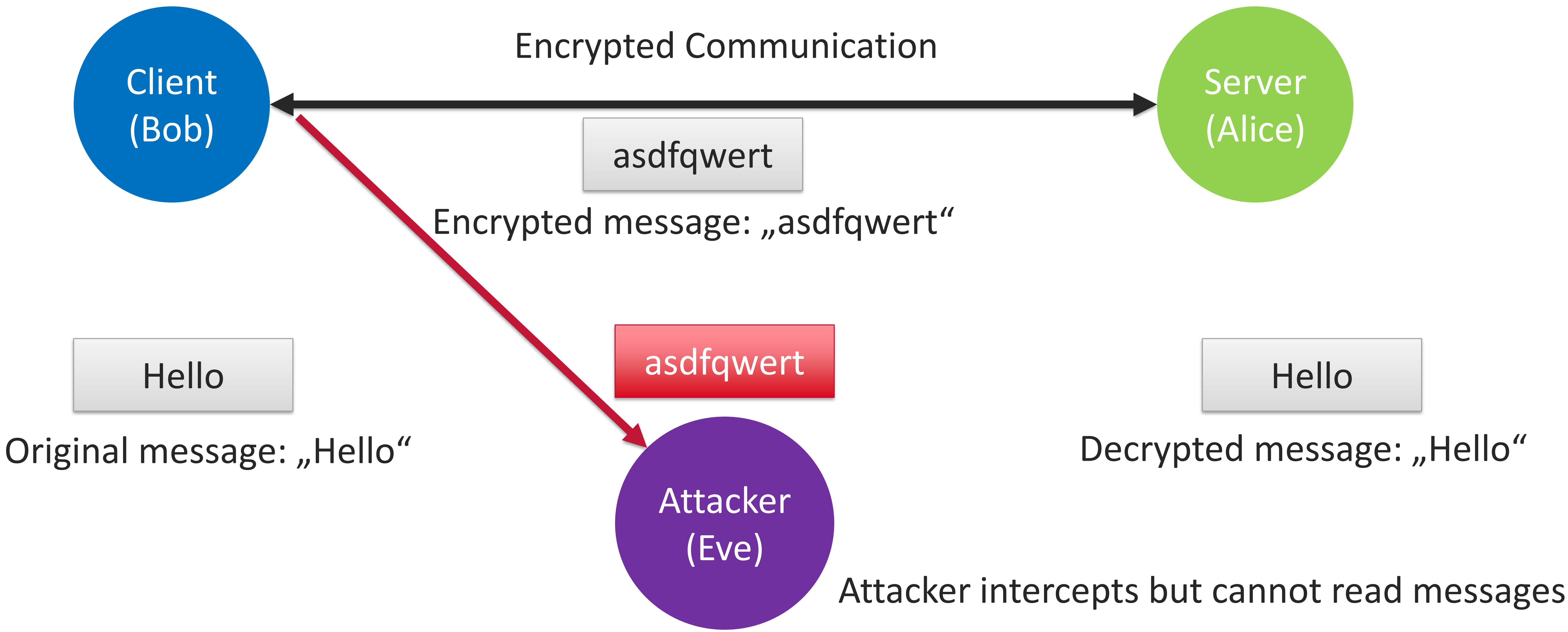
- Attack on **Confidentiality**
  - Eavesdropping (Sniffing)



Attacker intercepts and reads messages

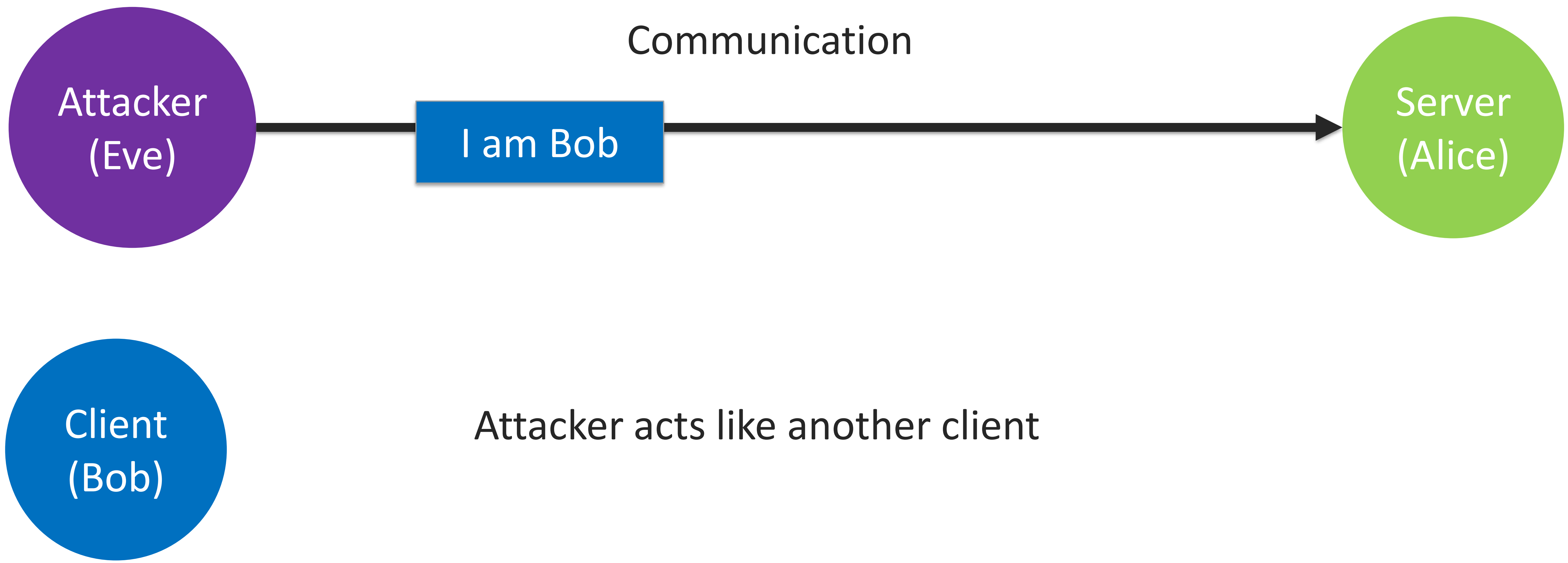
Security Goals

- Typical security measurement against Sniffing
  - Encrypting



Security Goals

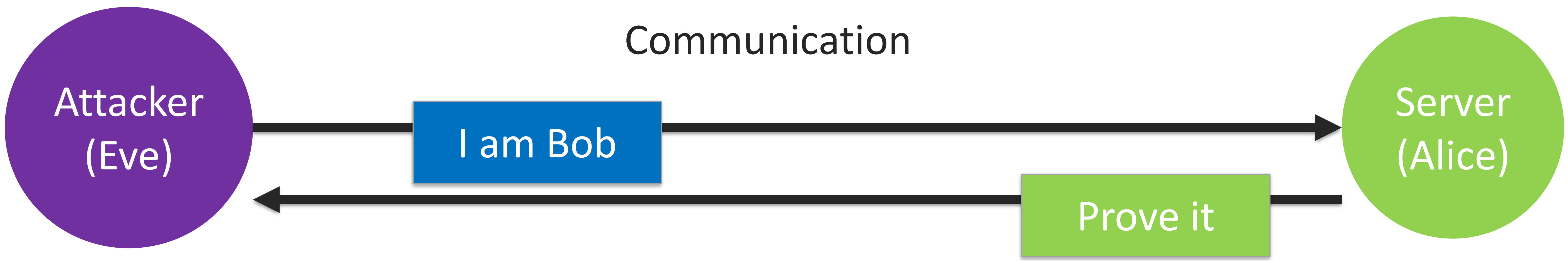
- Attack on **Authenticity**
  - Spoofing, Masquerading

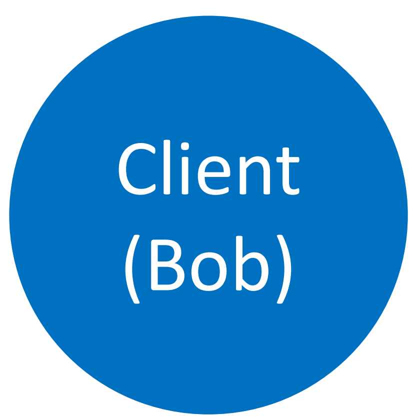


Attacker acts like another client

Security Goals

- Typical measurement against spoofing
  - Shared knowledge (secret) e.g. password





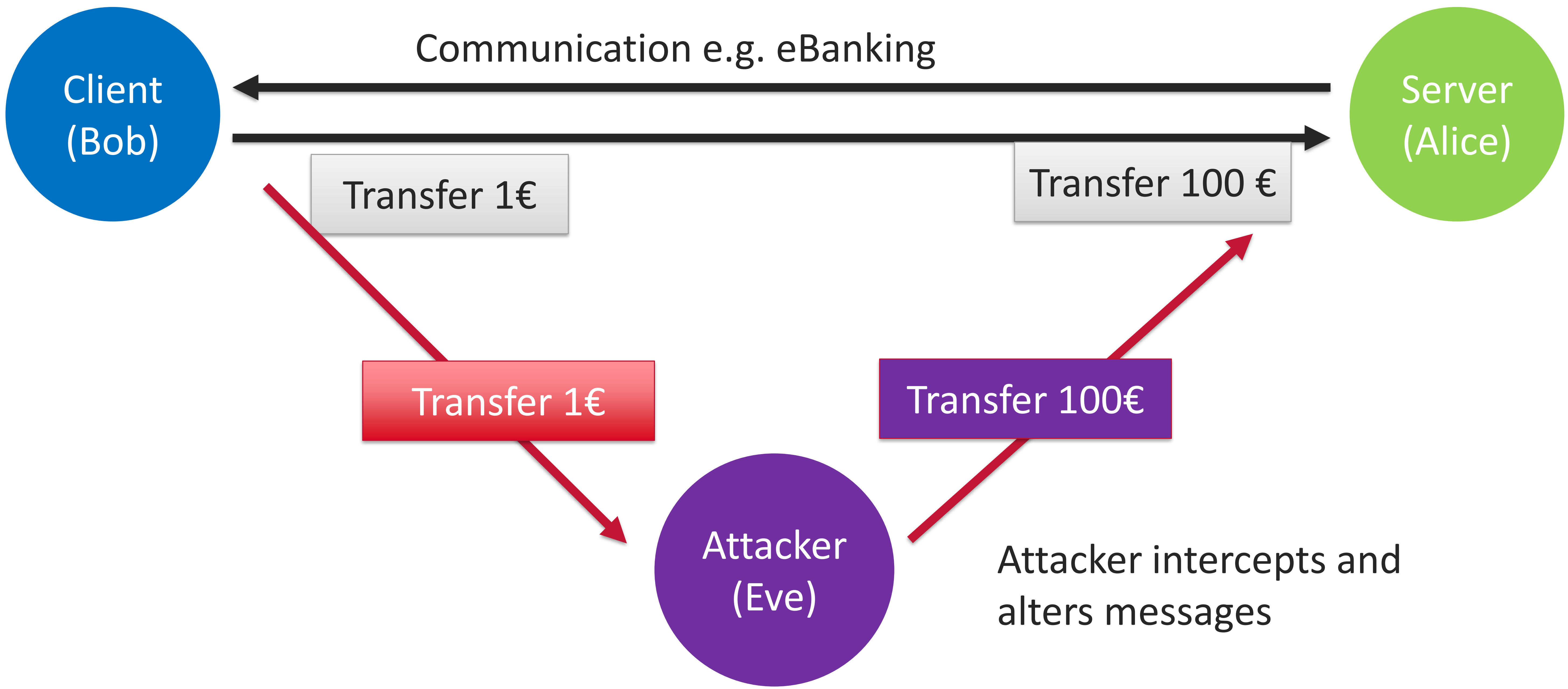
Server wants to verify if the client is really the person that it claims to be!

Solutions:

- Password
- Fingerprint
- Smartcard/Key
- …

Security Goals

- Attack on **Integrity**
  - Manipulating data



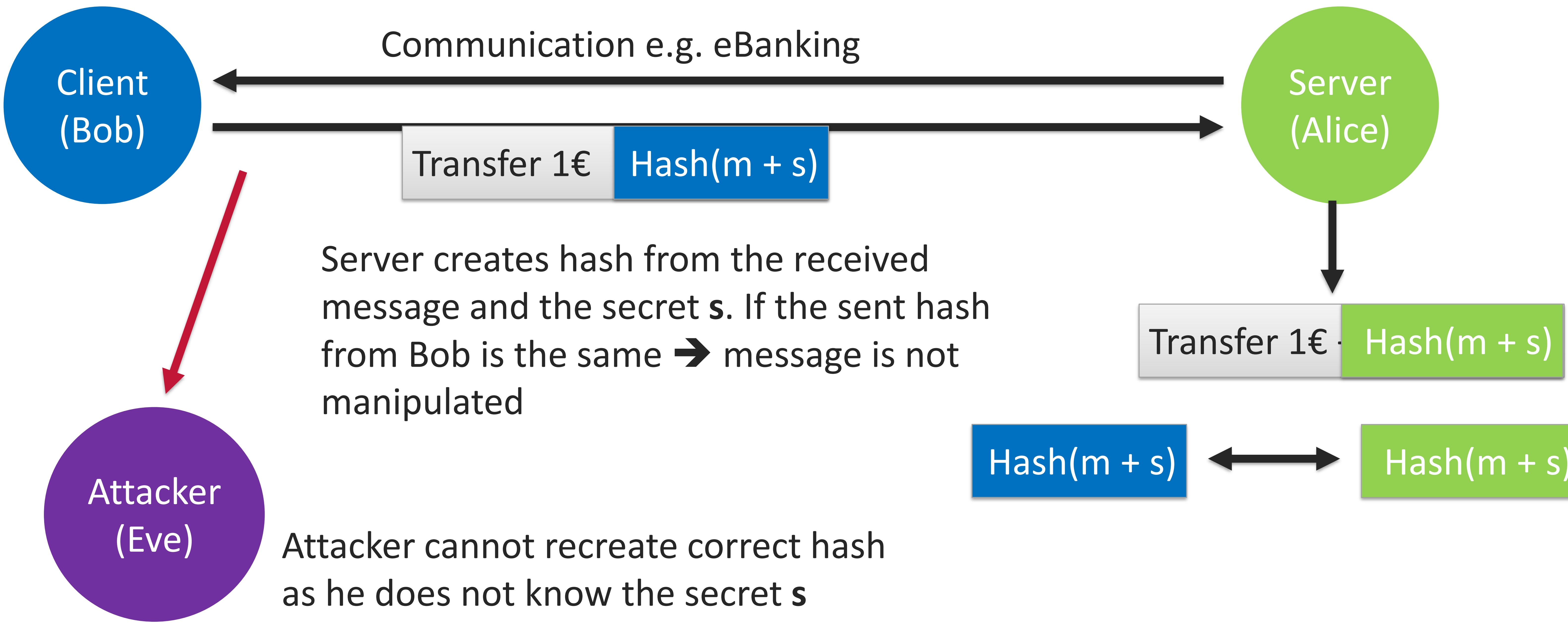
Attacker intercepts and alters messages

Security Goals

Transfer 1€ = m

Password = s

- Typical measurement against data manipulation
  - Checksum/Hashing

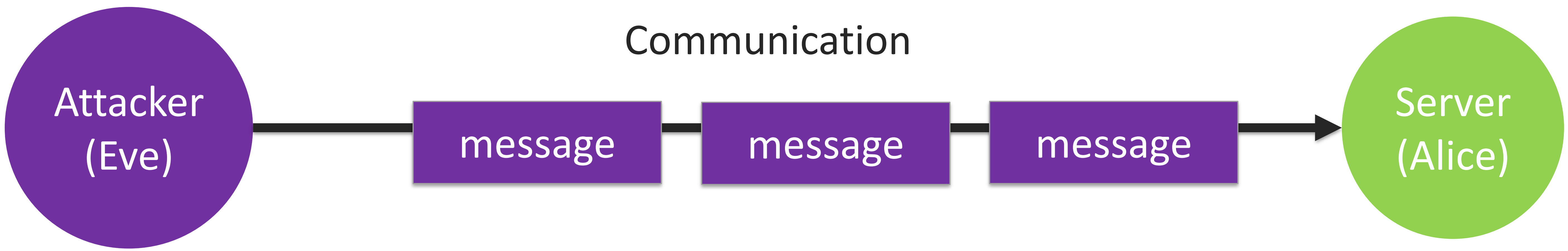


Server creates hash from the received message and the secret **s**. If the sent hash from Bob is the same ➔ message is not manipulated

Attacker cannot recreate correct hash as he does not know the secret **s**

Security Goals

- Attack on **Availability**
  - (Distributed-)Denial of Service Attack – (D)DoS



Attacker floods server with messages

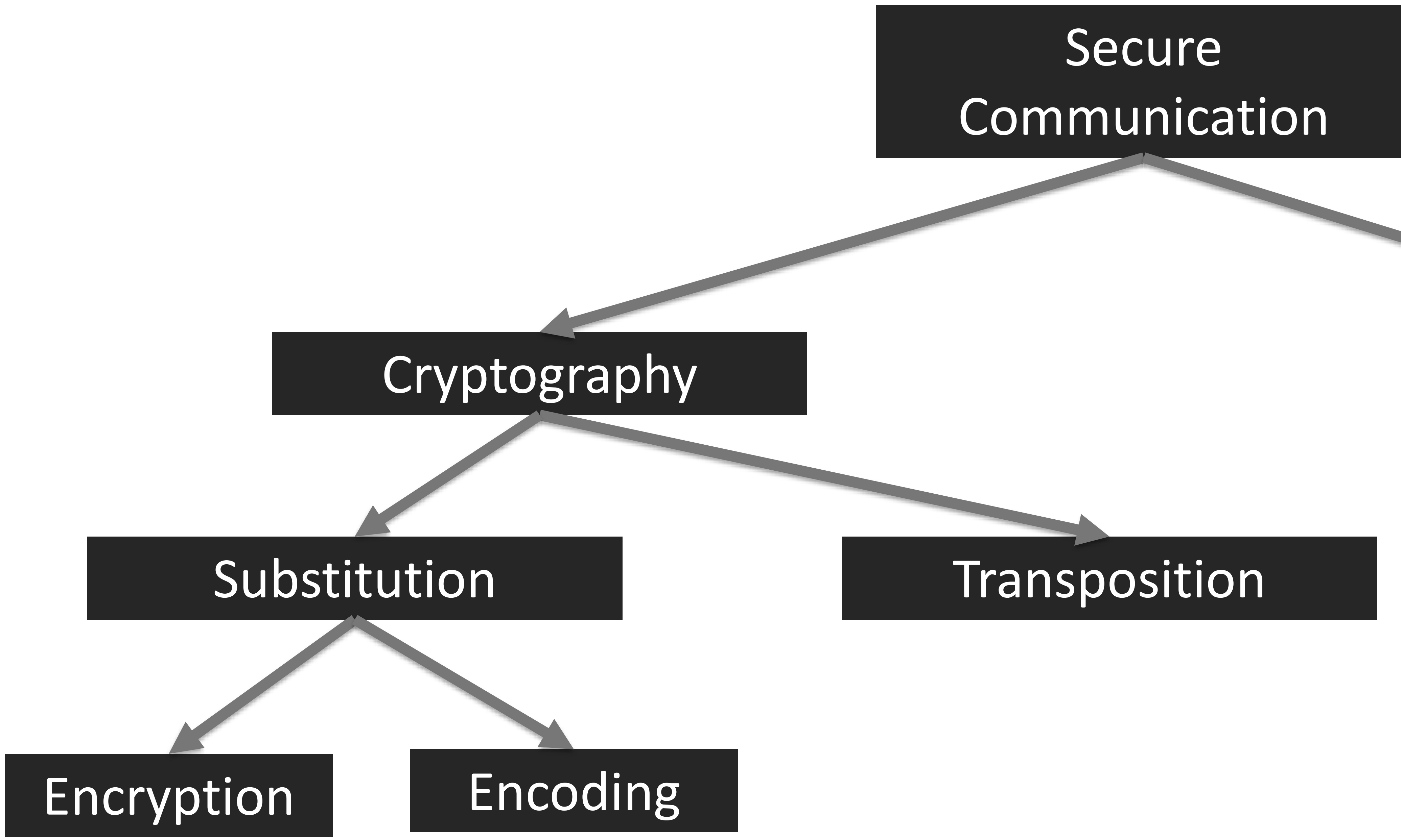
Examples of bad security

- Twitter 2017
  - Due to bad programming (officially it was a bug) passwords of users have been logged in internal log-files in plain text
- Sony PSN hack 2011
  - Due to a DDoS attack Sony had to temporarily shut down the PS-Network for its users
- Sony Pictures hack 2014
  - Hackers stole massive amount of data (e.g. E-Mails) of the company
  - Passwords have been apparently secured loosely (e.g. in txt files)
- Ashley Madison hack 2015
  - Hackers stole user data from the platform

Cryptography

- Terms:
  - P → Plaintext
  - C → Ciphertext
  - K → Key
  - Enc → Encrypt
  - Dec → Decrypt

Cryptography – Types of cipher



[https://www.probabilisticworld.com/caesar-column-ciphers-ancient-cryptography/]

Cryptography – Classic encryption

- Caesar cipher :
  - Simple monoalphabetic shift cipher
    - Every character in the alphabet will be shifted by X characters

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d

    - Key ➔ 4
    - P = hello
    - Enc(P,4) = **lipps**
    - How to crack this cipher? ➔ brute force
      - Try all shifts (length of alphabet)

Cryptography – Classic encryption

- Vigenère cipher :
  - Simple polyalphabetic shift cipher
    - Every character in the alphabet will shifted **by different X** characters
    - Basically multiple Caesar shifts
    - Key ➔ [4,2]
    - P = hello
    - Enc(P,[4,2]) = **lgpns**
    - The structure of the word is not recognisable anymore!
    - How to crack? ➔ Using frequency distribution
    - Every language uses specific constellation of characters
    - By building histo-, di- and trigram we can detect the length of the key
    - Brute force similar to Caesar Encryption

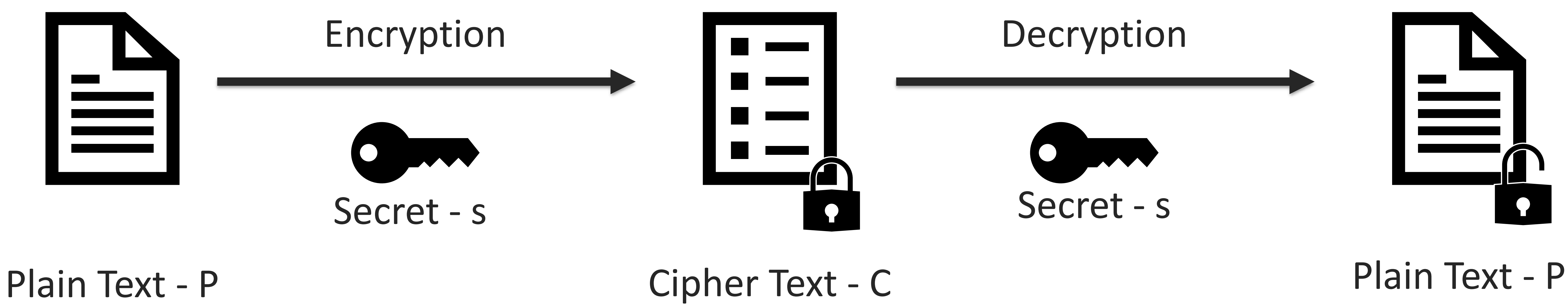
h	e	l	l	o
4	2	4	2	4

Cryptography – Kerckhoffs's principle

- Created in the 19th century
- Security through obscurity?
  - By hiding/developing a (proprietary) encryption algorithm you are not achieving any security!
- A secure system should not be dependend on the obscurity of the algorithm! It is very likely that the algorithm can be reverse engineered.
- The security comes with the obscurity/security of the key/secret used in an algorithm.
- A public algorithm can be checked and evaluated by experts. Vulnerabilities can be exposed and fixed quickly.

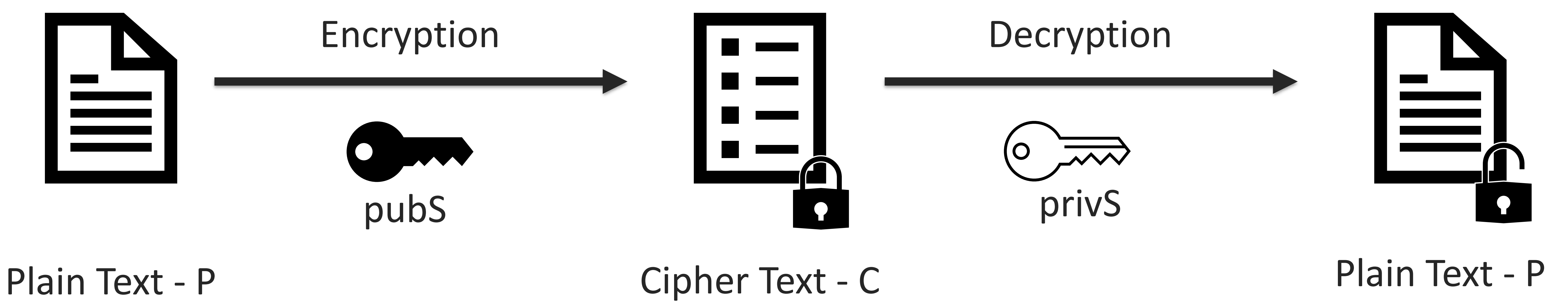
Cryptography – Symmetric encryption

- We have only one key/secret/password
- This secret s is able to encrypt plain text and decrypt cipher text



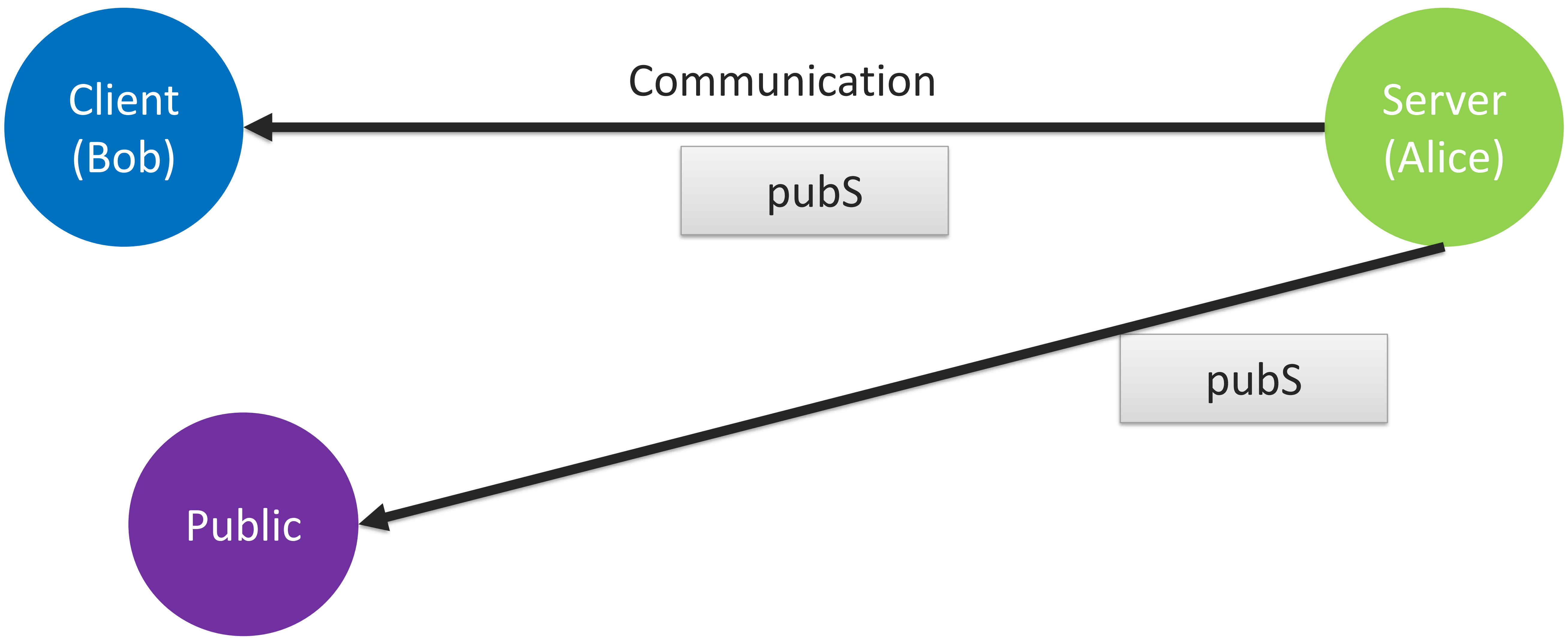
Cryptography – Asymmetric encryption

- We have two keys/secrets/passwords
- Public Secret – everyone can know this – **pubS**
- Private Secret – only the secret keeper should know this - **privS**
- Both secrets can encrypt plain text and decrypt cipher text
- But the keys only work together – if you encrypt with pubS you can only decrypt with privS – vice versa



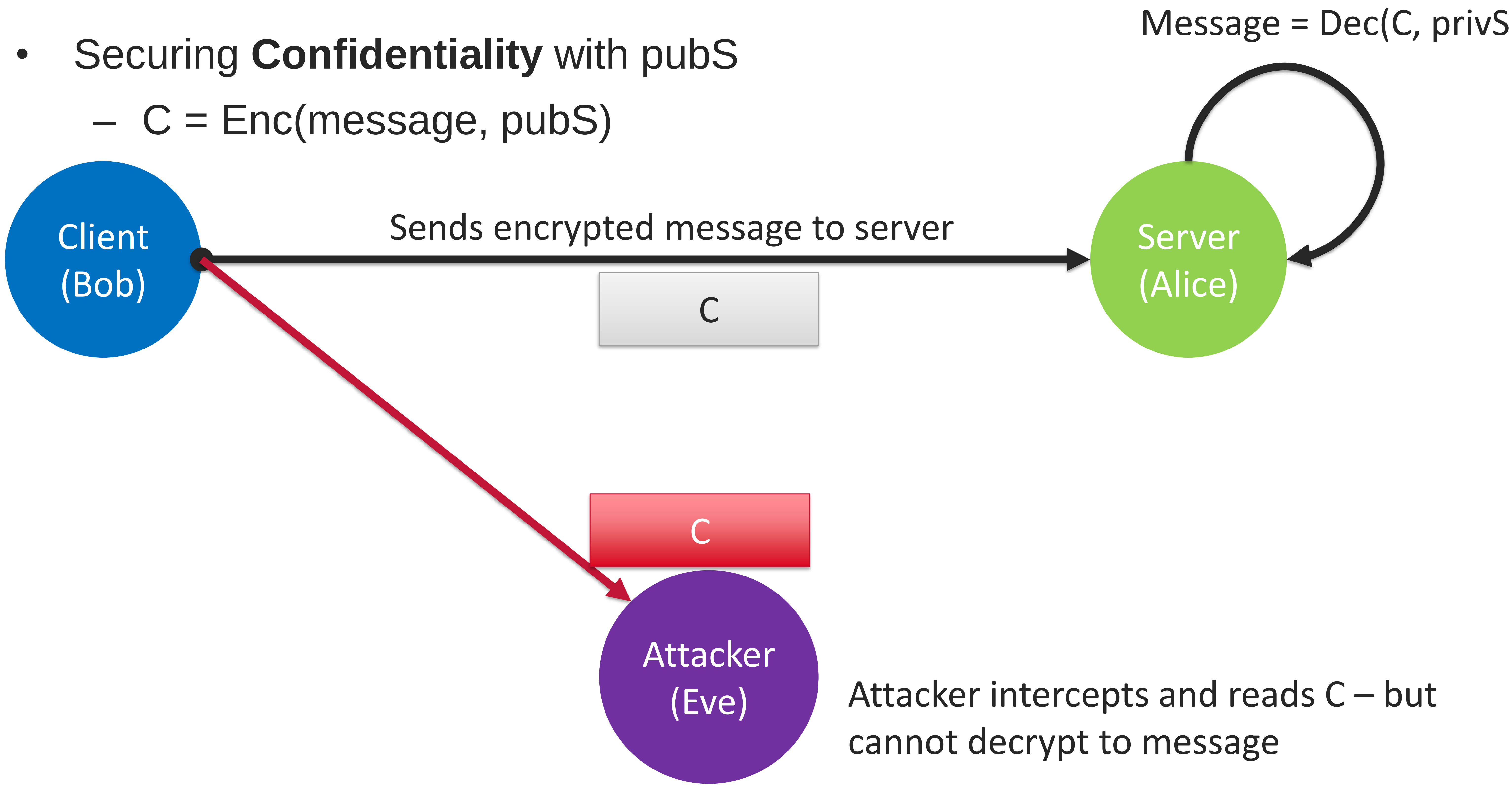
Cryptography – Asymmetric encryption

- Server publishes his public key (pubS)
  - Everyone can see and get this secret



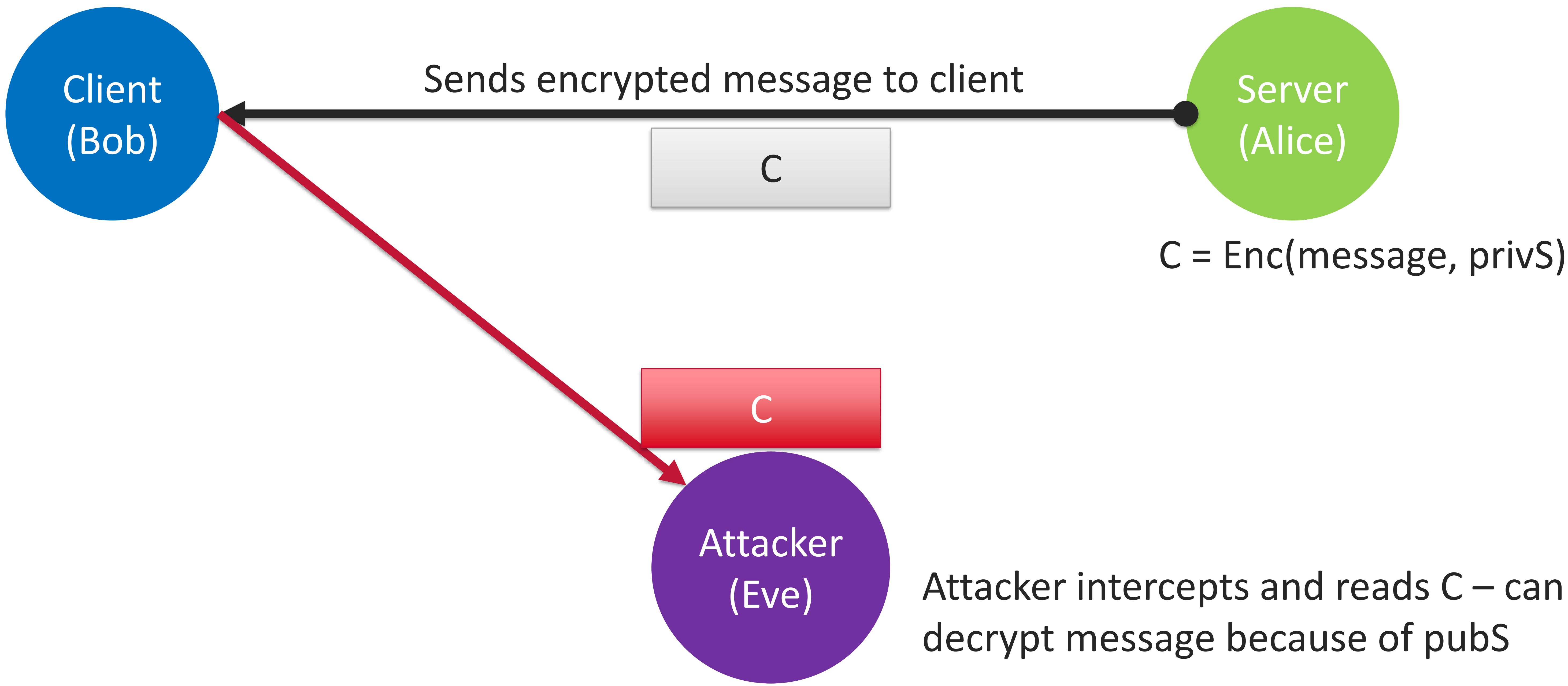
Asymmetric encryption - Confidentiality

- Securing **Confidentiality** with pubS
  - C = Enc(message, pubS)



Asymmetric encryption – Authenticity and Integrity

- But if the server send a message to a client everyone can decrypt it?



Asymmetric encryption – Authenticity and Integrity

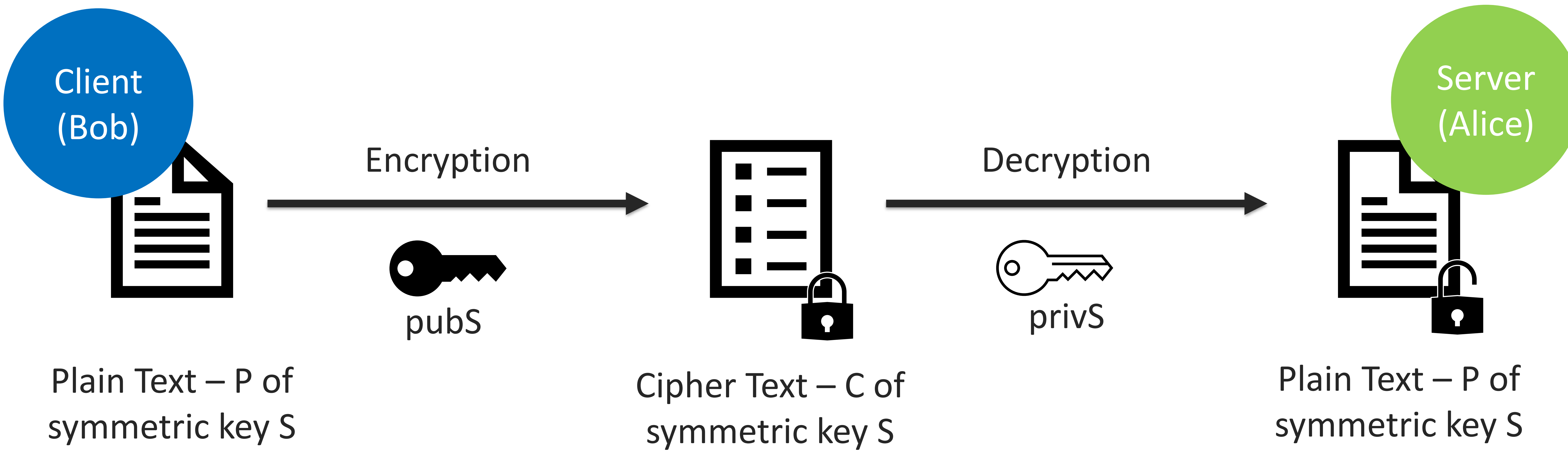
- But if the server send a message to a client everyone can decrypt it?
  - Yes
- Why is it still useful?
  - By being able to decrypt the message with the pubS it is clear that the origin of the message has to be the server!
  - Only the server should have the privS with which the message was encrypted in the first place.
  - We have (theoretically) achieved Authenticity ➔ we know who has sent the message
  - We have (theoretically) achieved also **I**ntegrity ➔ the message was not altered

Comparison Symmetric - Asymmetric

- Symmetric
  - Fast encryption/decryption
  - Secret needs to be exchanged!
- Asymmetric
  - Slow encryption/decryption
  - No Keys need to be securely exchanged!

Hybrid Solution

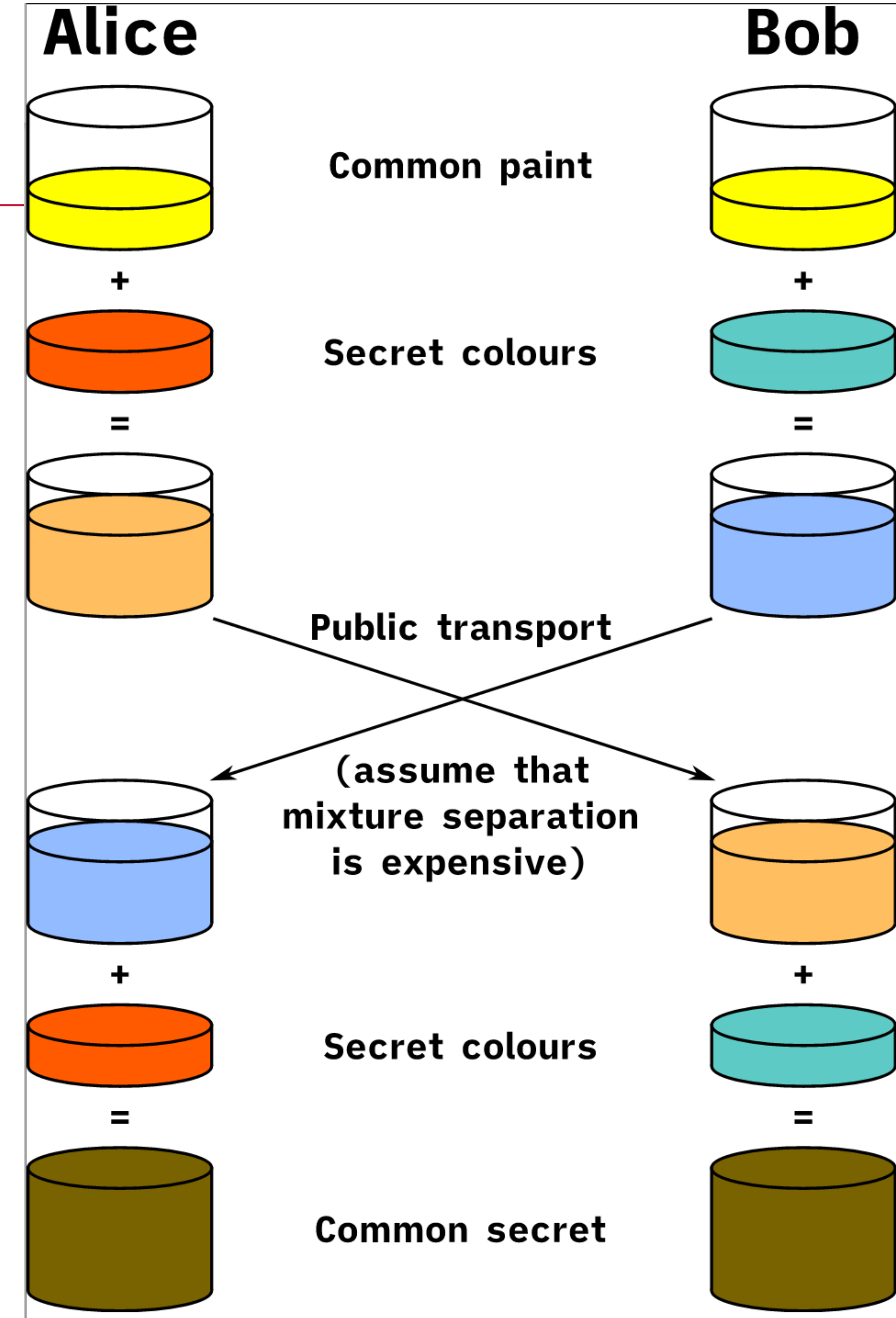
- By using both encryption methods we can combine the advantages of fast encryption/description and the necessity to exchange keys/secrets is also overcome!



Now both the client and the server have a symmetric key S with which they can encrypt and decrypt symmetrically

Hybrid Solution – Key Exchange DH

- The "common paint" can be shared publicly.
- The "Secret colours" are basically the privS.
- The base idea is that it is easy to mix colours but difficult to find the base colours!
- By adding the secrets individually they can end up with a common secret without sharing their privS.
- In reality discrete logarithm is used to perform those one way functions.
- It is very difficult to reverse calculate modulo operation on prime numbers.



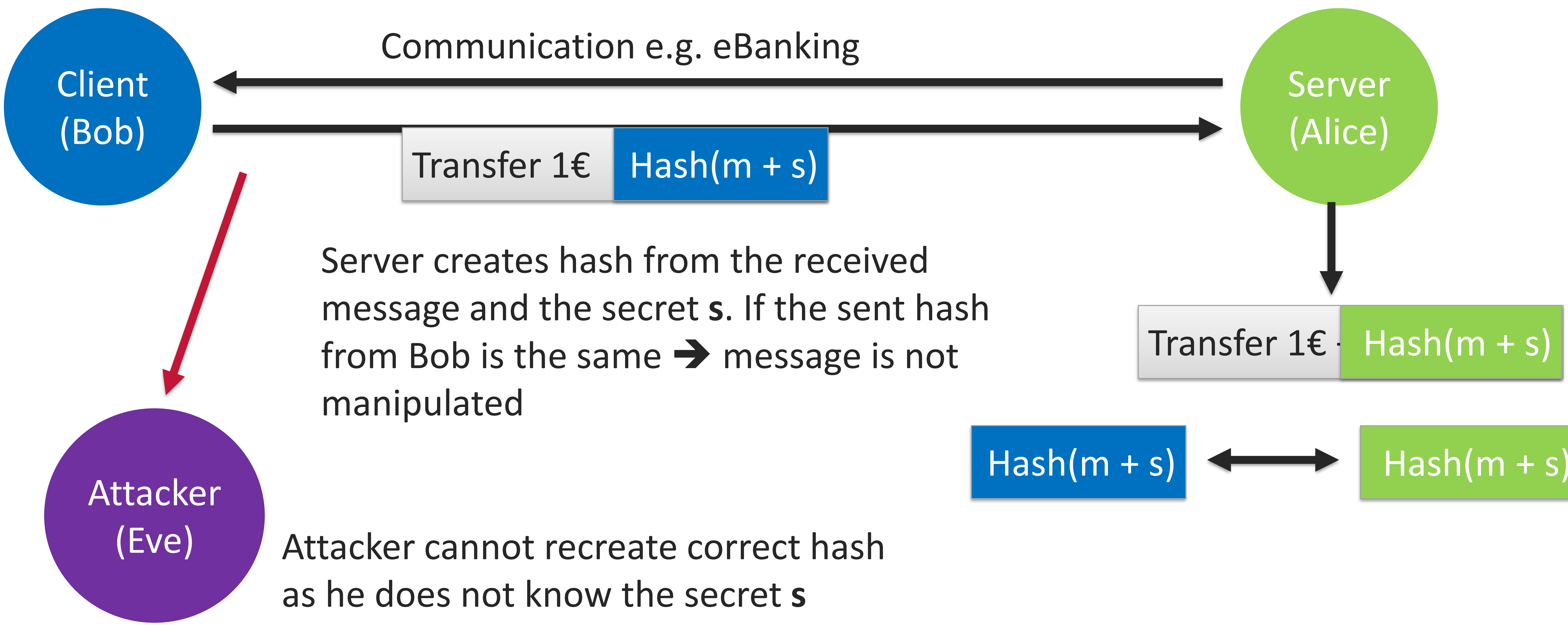
https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange

Cryptography – Hashing

- Hashing are mathematical functions that
  - Can accept “any” size of input
  - Create a result (hash) of a fixed size
  - Can be calculated easily in only one way ➔ Hash(message) = **h**
  - Not easy to find a message behind a **h**
  - High collision resistance
    - Hash(message1) != Hash(message2)
    - Birthday Attack
- Hashing does not use any keys/secrets
- SHA-3 is a current hashing algorithm that is deemed to be secure at the moment

Cryptography – Hashing Example usage

- Hashing (e.g. MAC or HMAC) are used to achieve Authenticity & Integrity
  - s is a shared secret/key



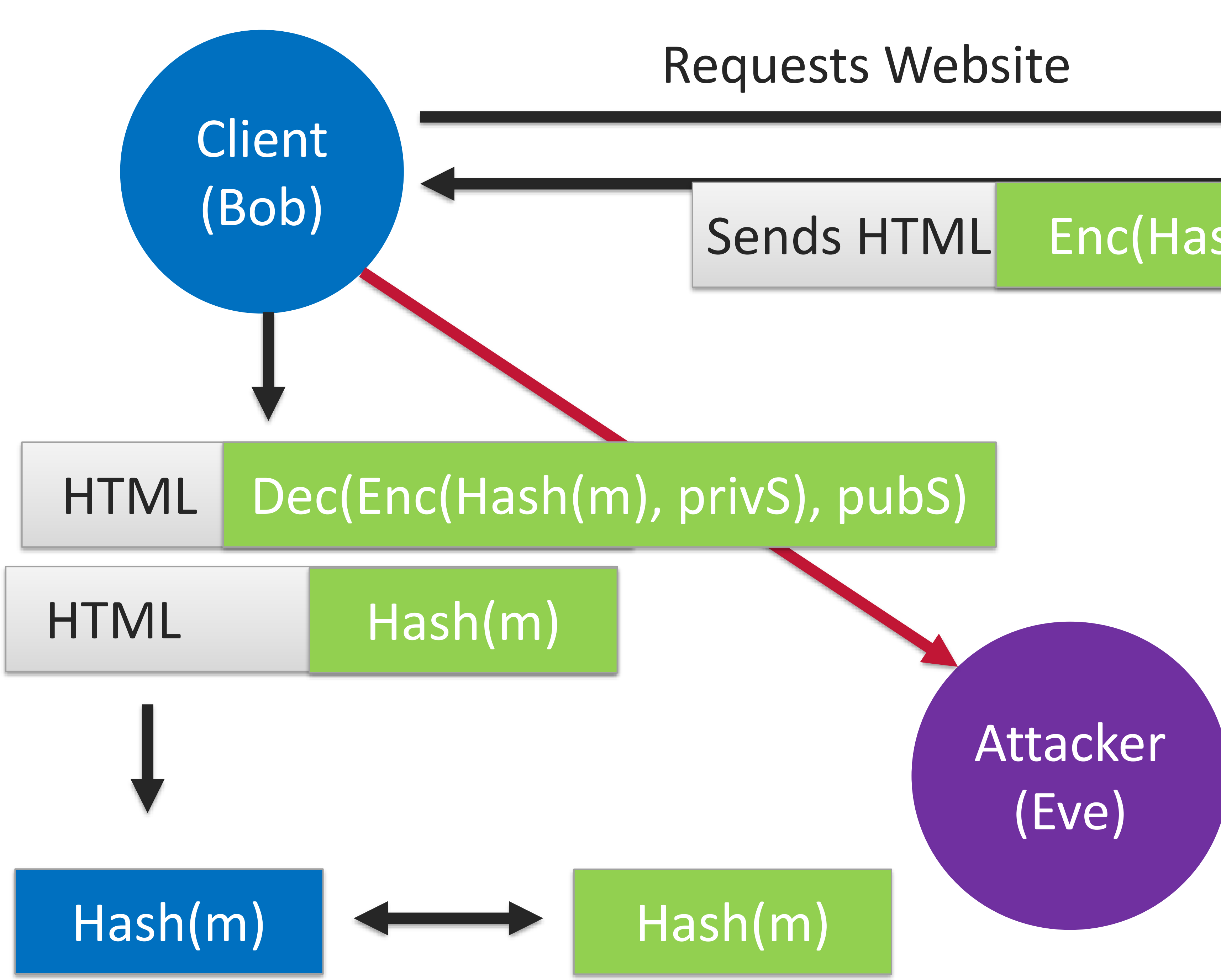
Server creates hash from the received message and the secret **s**. If the sent hash from Bob is the same ➔ message is not manipulated

Attacker cannot recreate correct hash as he does not know the secret **s**

Cryptography – Digital Signature

- We learned about the hybrid method of combining asymmetric and symmetric cryptographic but if we need only the security goal Authenticity & **I**ntegrity and no **C**onfidentiality the hybrid method is to much overhead
- Normal asymmetric encryption would also be to much overhead!
- Therefore we make use of an efficient hashing to achieve A & I
- Additionally the digital signature achieves also the security goal "Non-Repudiation"

Cryptography – Digital Signature/Certificate Example usage



Client decrypts the encrypted signature and recreates the hash from the received HTML. If the encrypted hash and the recreated hash are identical then that means that the message (HTML) is authentic and was not manipulated

Attacker cannot recreate correct encrypted addition to the message as he does not know the secret **privS** so he cannot fake the website

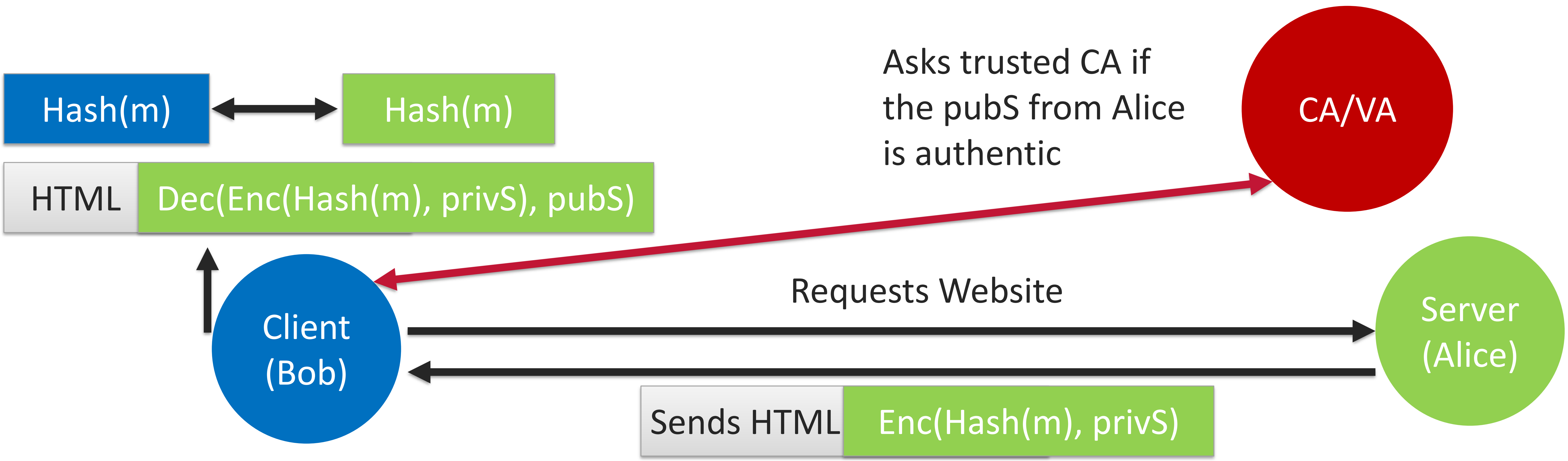
Cryptography – Certificates

- Basic problem of asymmetric encryption:
  - Is the pubS authentic?
- To make sure that the public key is authentic we need a trusted third party – those are also called **Certification Authority (CA)**
- The CA signs a message with the private key of itself (CA-privS)
  - It adds the pubS, expiration date and the subject to the message
- This message + the signed message = digital certificate
- This happens when the certificate is issued!

- There is a PKI ➔ public key infrastructure
- They are authorized to create/sign, validate and invalidate certificates

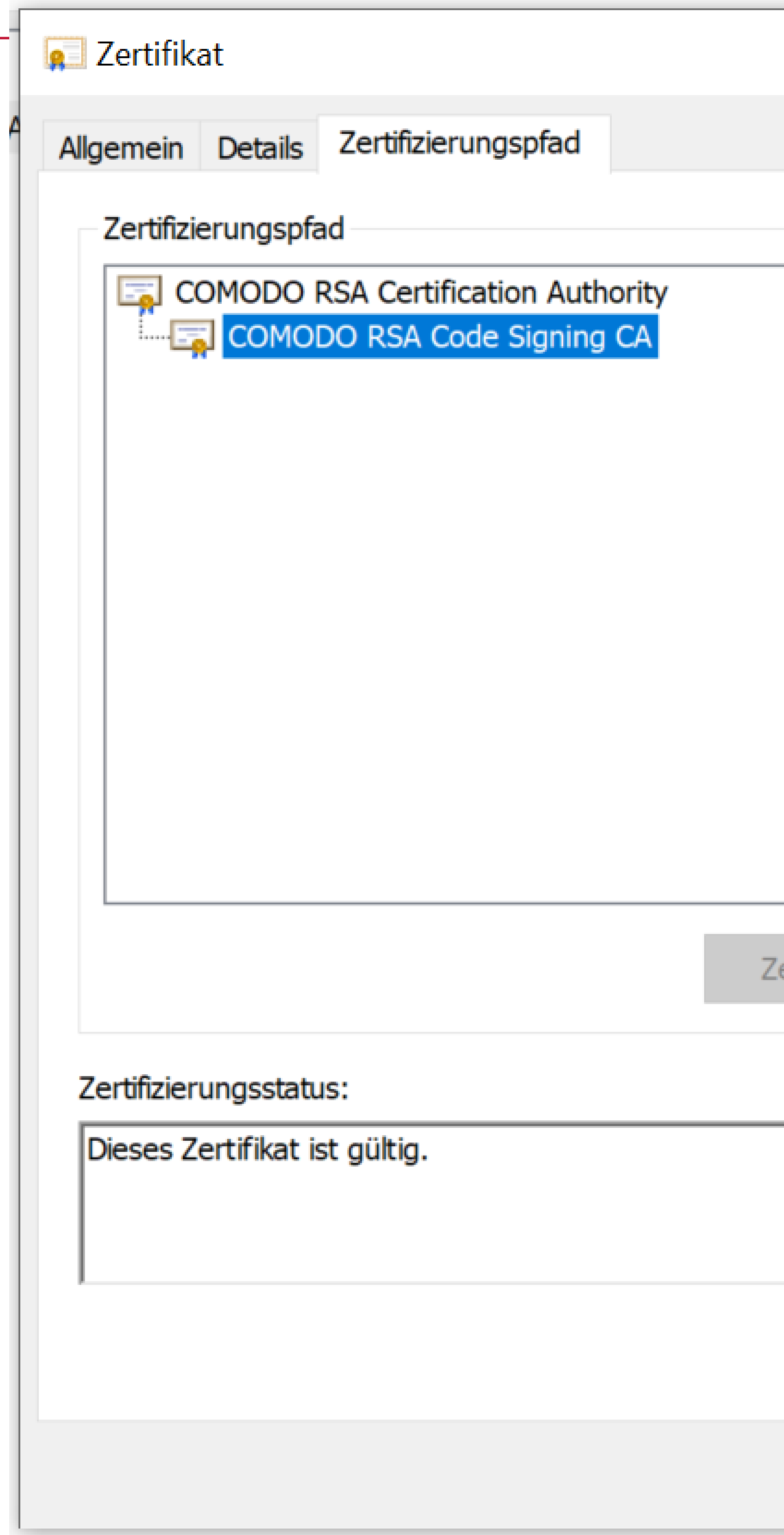
Cryptography – Certificates - PKI

- Typical operating systems/browser have pre defined sets of trusted CA
- This CA contains other parts like the validation authority (VA) which is responsible for checking issued certificates
- The client asks the CA immediately after receiving the pubS from a server



Cryptography – Certificates - PKI

- The installed certificates in our operating systems/browsers are mostly root certificates which makes the issuer a root authority!
- Those root authorities can issue certificates for other sub CA ➔ Intermediate CA – a Chain of Trust is build
- Those sub CA can also issue certificates for other sub CA
- You do not have to have every sub CA installed on your machine
- If a certificates needs to be checked for validation the sub CA will be checked by the root CA who issued it in the first place
- If it is valid the certificate that was issued is also valid



Cryptography – SSL/TLS

- SSL (Secure Socket Layer) is unfortunately still used as a name although this technology is not used anymore! TLS (Transport Layer Security) currently in the version 1.2 & 1.3 are the standard “security” protocol on the internet. Sit on the 5 layer between TCP&HTTP
- Based on two main sub-protocols
  - Handshake protocol
  - Record protocol
- The security goals are mainly
  - Authentication
  - Confidentiality
  - Integrity
  - (Non-Repudiation)

Cryptography – SSL/TLS

- Handshake protocol
  - Before a browser can get HTML from a server they need to go through a whole process of checking the certificate and exchanging keys!
  - In TLS 1.2 all of the different encryption (symm+asymm) and hashing methods are being negotiated.
  - Combination of hybrid encryption and digital signature is used.
- Record protocol
  - The exchange of data happens within this protocol
  - Integrity + Authenticity through Digital Signature/(H)MAC
  - Confidentiality through symmetric encryption
- Whole HTTPS communication is **secured – still don’t use GET for Logins…**

Cryptography – Breaking Security

- Brute Force
  - Try every combination of passwords til we find the correct password
  - Solution? Try to block suspicious requests e.g. > 10 requests withing 30s
- Rainbow Tables
  - Attacker has a pre-created list of typical terms/words/passwords in a hashed format
  - If he gets access to the hashed passwords he might compare those with the rainbow table
  - Solution? Use salt (& pepper) when hashing passwords
- Access to Data?
  - Attacker might also have full access to the encryption system which means he can reverse engineer the password – Kerckhoffs's priciple

Cryptography – Conclusion

https://www.internet-sicherheit.de/crypto-poster

How Safe Is Your Password?

Time it would take a computer to crack a password with the following parameters

Number of characters	Lowercase letters only	At least one uppercase letter	At least one uppercase letter +number	At least one uppercase letter +number+symbol
1	Instantly	Instantly	-	-
2	Instantly	Instantly	Instantly	-
3	Instantly	Instantly	Instantly	Instantly
4	Instantly	Instantly	Instantly	Instantly
5	Instantly	Instantly	Instantly	Instantly
6	Instantly	Instantly	Instantly	Instantly
7	Instantly	Instantly	1 min	6 min
8	Instantly	22 min	1 hrs	8 hrs
9	2 min	19 hrs	3 days	3 wks
10	1 hrs	1 mths	7 mths	5 yrs
11	1 day	5 yrs	41 yrs	400 yrs
12	3 wks	300 yrs	2,000 yrs	34,000 yrs

Source: Security.org

statista