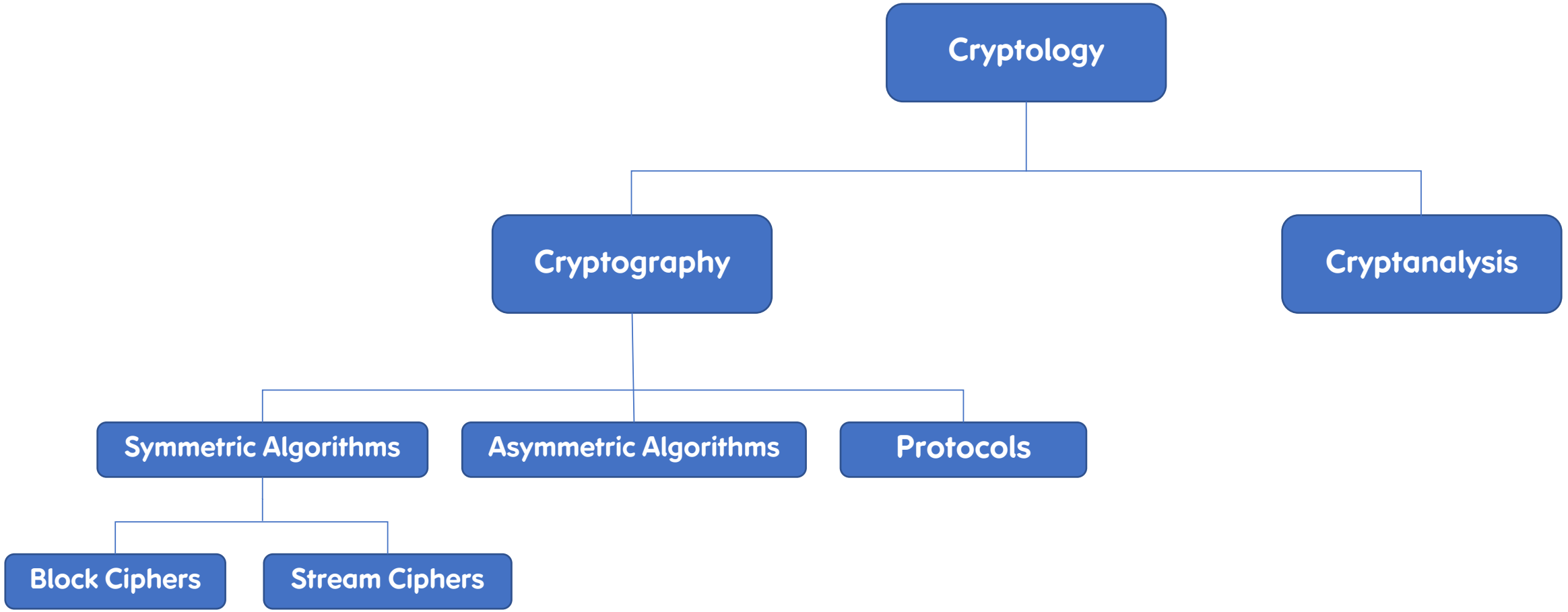
Modern Cryptography

Viktorija Almazova

Cloud Security Architect

Where do we have crypto?



Classification of the Field of Cryptology

Some basic facts..

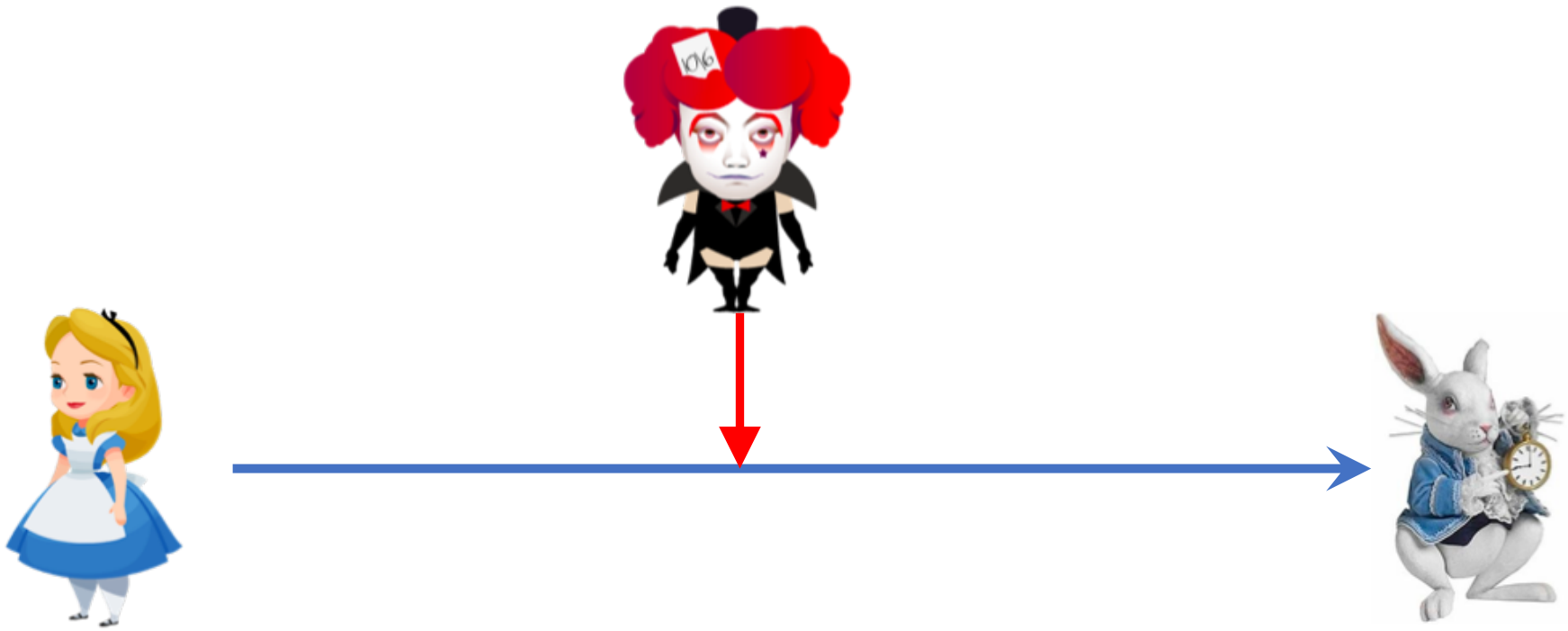
- **Ancient Crypto:** Early signs of encryption in Egypt in ca. 2000 B.C. Letter-based encryption schemes (e.g., Caesar cipher) popular ever since.

- **Symmetric algorithms:** All encryption schemes from ancient times until 1976 were symmetric ones.

- **Asymmetric algorithms:** In 1976 public-key (or asymmetric) cryptography was openly proposed by Diffie, Hellman and Merkle.

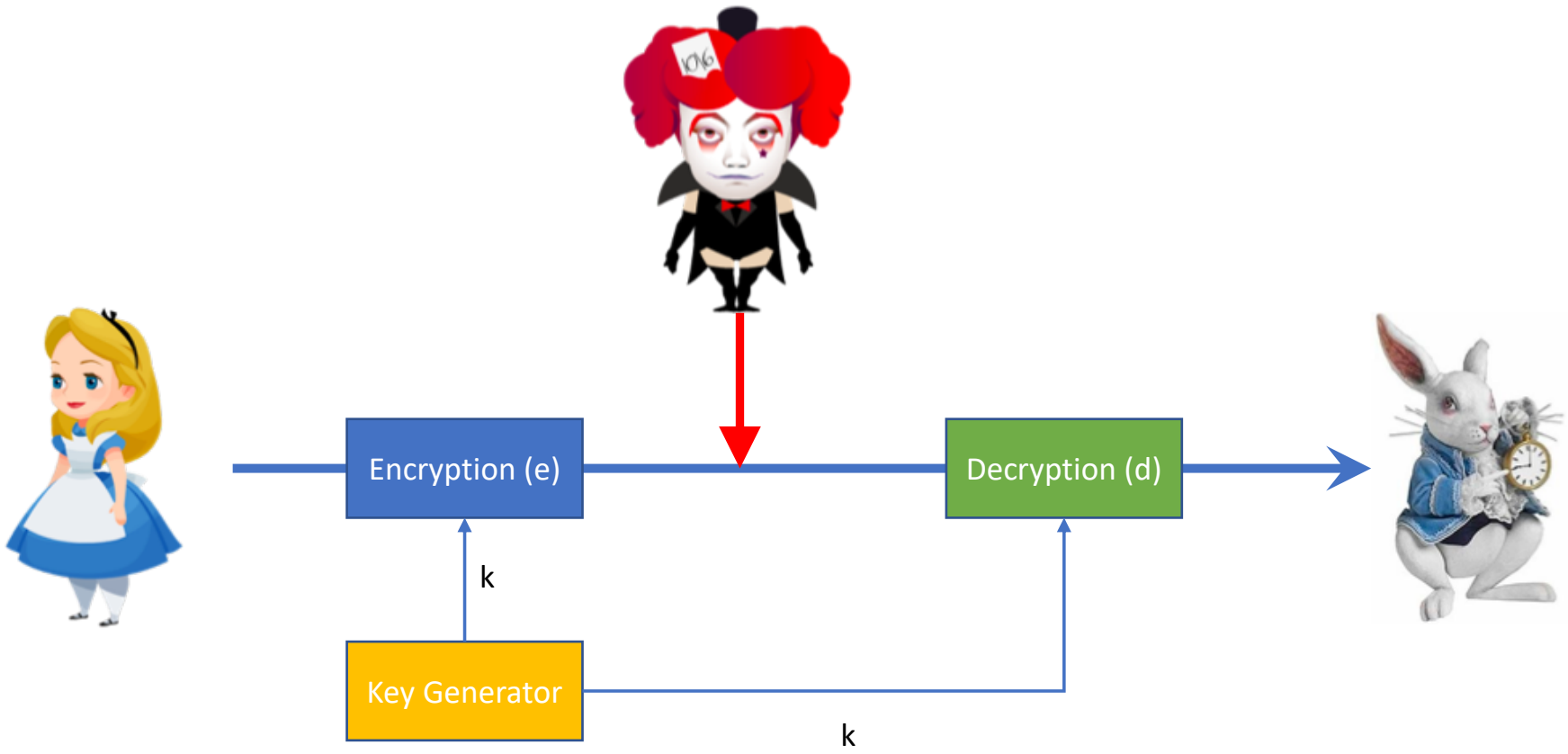
- **Hybrid Schemes:** The majority of today‘s protocols are hybrid schemes, i.e., the use both
  - symmetric ciphers (e.g., for encryption and message authentication) and
  - asymmetric ciphers (e.g., for key exchange and digital signature).

Symmetric Cryptography



Symmetric Cryptography

Alternative names: **private-key**, **single-key** or **secret-key** cryptography



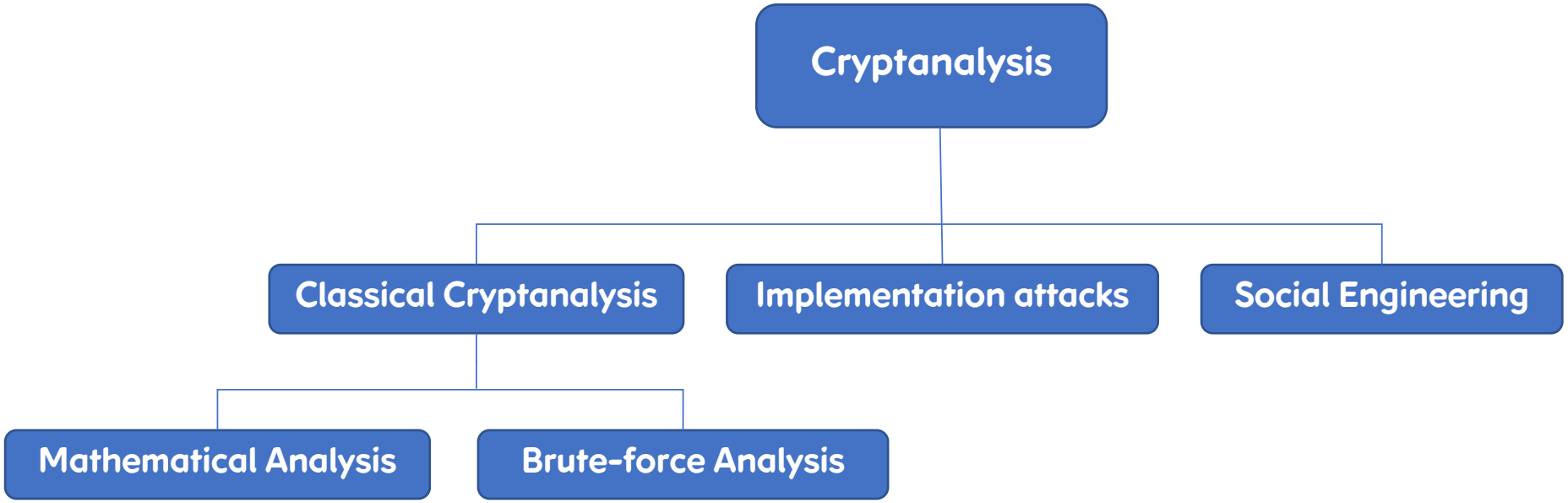
Cryptanalysis..

- There is no mathematical proof of security for any practical cipher
- The only way to have assurance that a cipher is secure is to try to break it (and fail)!

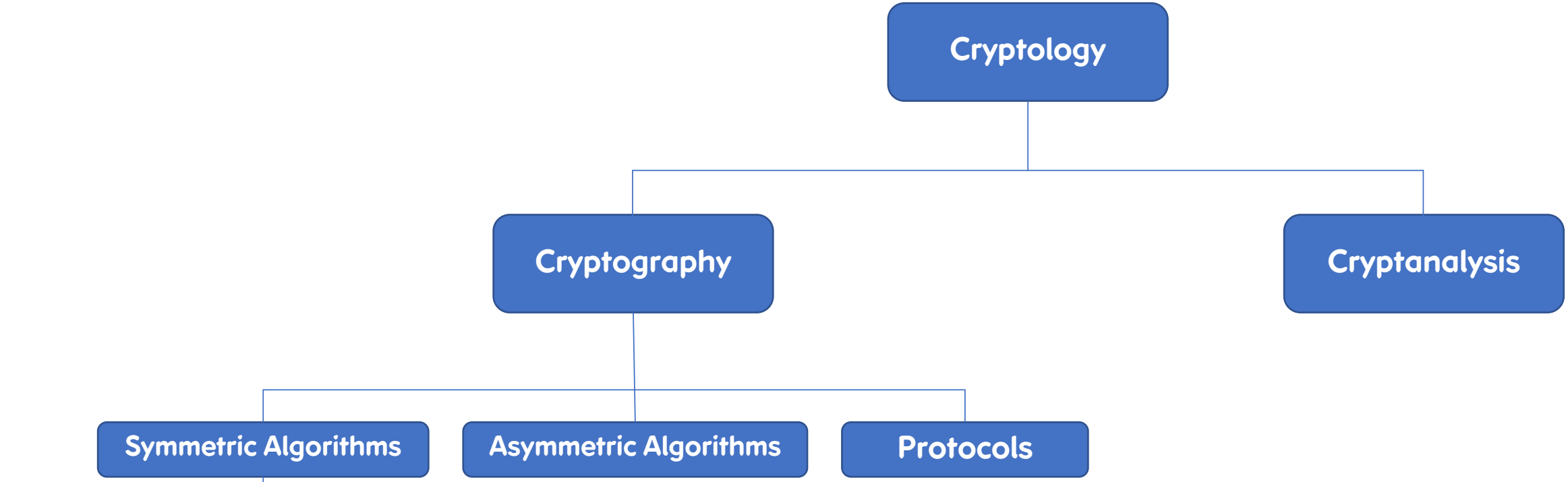
Kerckhoff‘s Principle (1883) is paramount in modern cryptography:

> A cryptosystem should be secure even if the attacker knows all details about the system, with the exception of the secret key.

In order to achieve Kerckhoff‘s Principle in practice: Only use widely known ciphers that have been cryptanalyzed for several years by good cryptographers!



Classification of the Field of Cryptanalysis


Cryptology
Cryptography
Cryptanalysis
Symmetric Algorithms
Asymmetric Algorithms
Protocols
Block Ciphers
Stream Ciphers
DES/3DES
Blowfish
Twofish
AES
RC4
A5/2
Trivium
ChaCha20
RSA
ECDSA
Digital signatures
RSA, ElGamal
Hash Functions
MD5, SHA-1,
Message Authentication Codes (MACs)
HMAC
Key Establishment
Diffie-Hellman key exchange

SSL/TLS..

What is SSL?

- Secured Sockets Layer (SSL) is an older version of Transport Layer Security (TLS).
- SSL and TLS are often used to refer to the same thing, but you have to know that SSL is since 2014 considered as non-secure.
- All commonly used browsers support now both protocols.

What is TLS?

- A key point is to remember that TLS is a secure communication protocol point-to-point.

Typical use of TLS encryption

- Web
- Email
- SFTP

Certificates

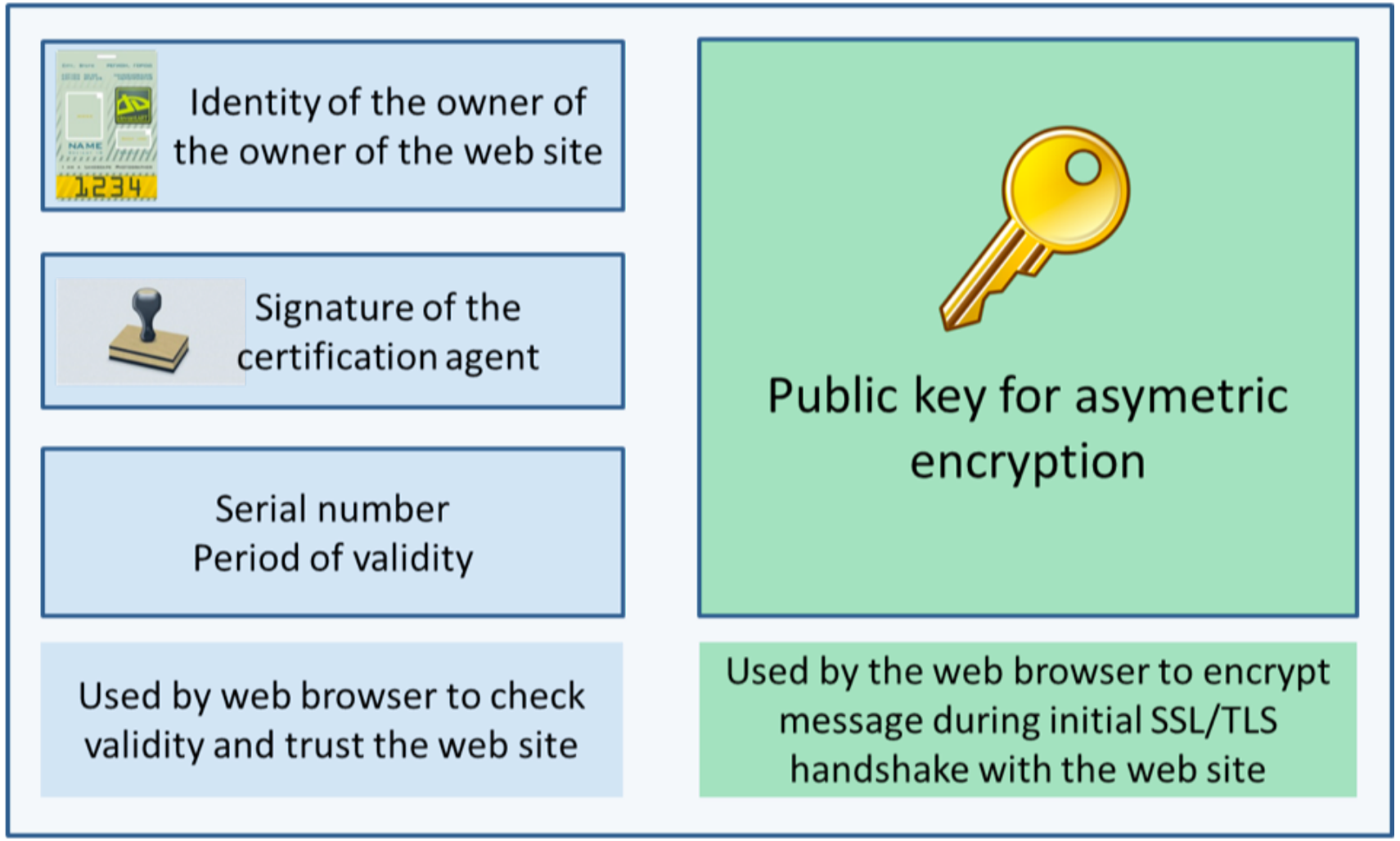
Using a SSL / TLS link between a web browser and a web server, the web browser requires that the server identifies itself using an SSL / TLS certificate.

The browser uses this certificate to verify if it is valid and to trust the web site. The web browser will trust the web server if the certificate of the web site is valid:

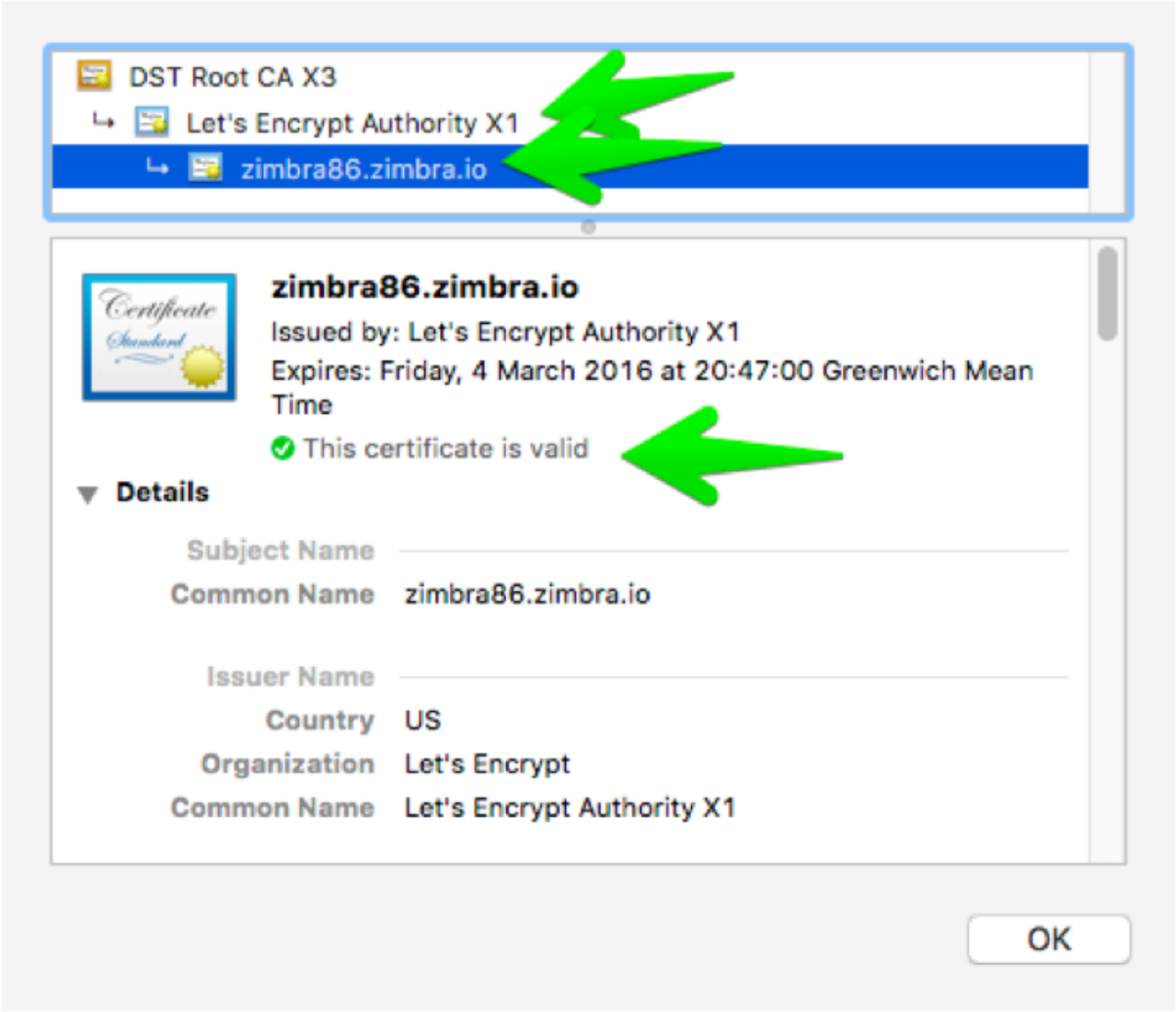
- The certificate is issued for the website to which the browser is connecting to
- The certificate is not expired
- The certificate is not revoked
- The certificate is signed by a trusted third-party acting as certification agent (CA)

If the certificate is valid, the web browser is ready to start an encrypted connection with the web server; else the web browser warns the visitor that the website is not trusted.

Certificate



Chain of Trust



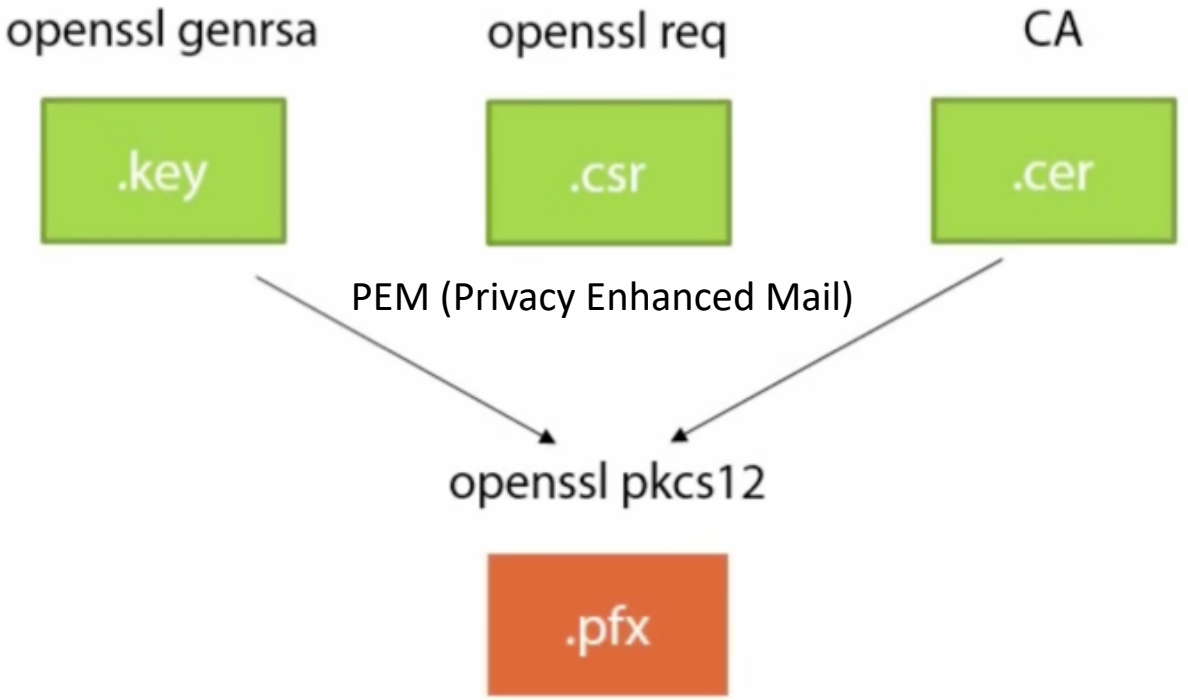
https://letsencrypt.org/certificates/

Public Key Cryptography Standards

- PKCS #7
  - Cryptographic Message Syntax
  - Certificates
- PKCS #10
  - Certification Request
- PKCS #12
  - Personal Information Exchange Syntax
  - Private Keys
- Common file formats

Public Key Cryptography Standards

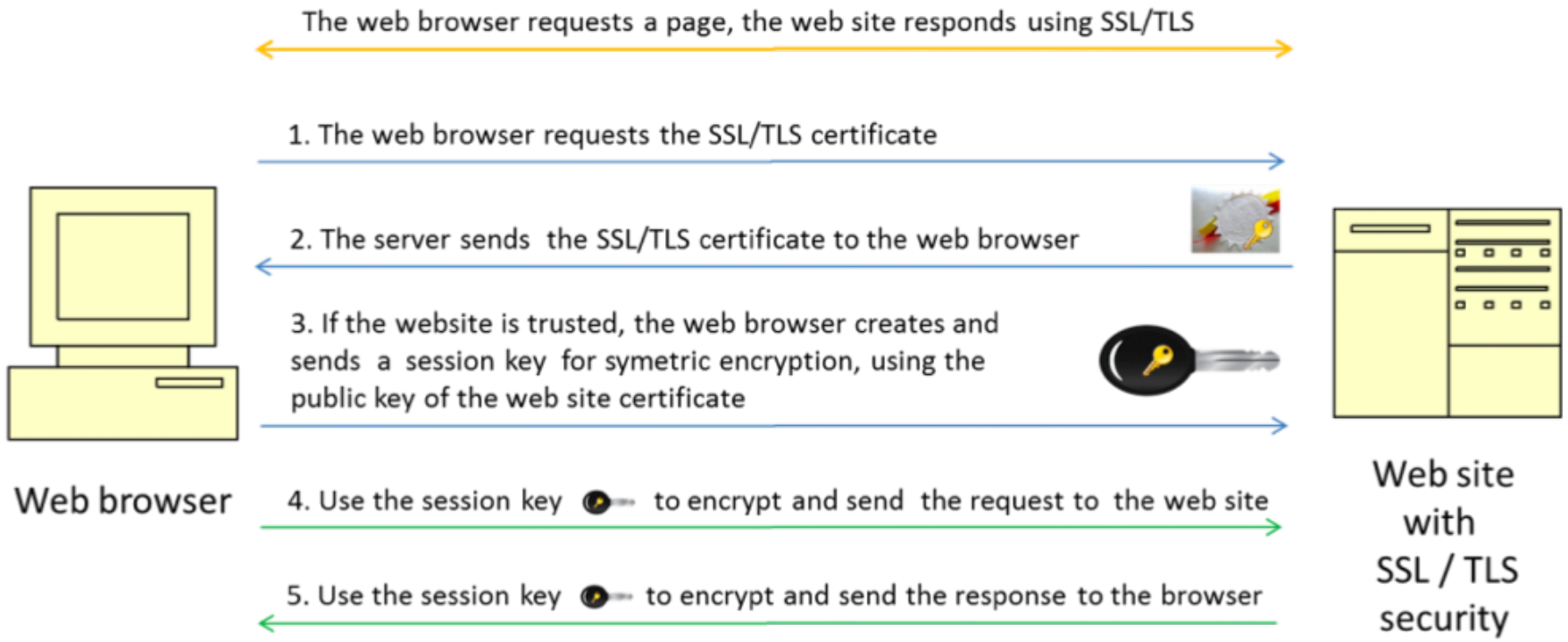
- PKCS #7
  - Cryptographic Message Syntax
  - Certificates
- PKCS #10
  - Certification Request
- PKCS #12
  - Personal Information Exchange Syntax
  - Private Keys
- Common file formats

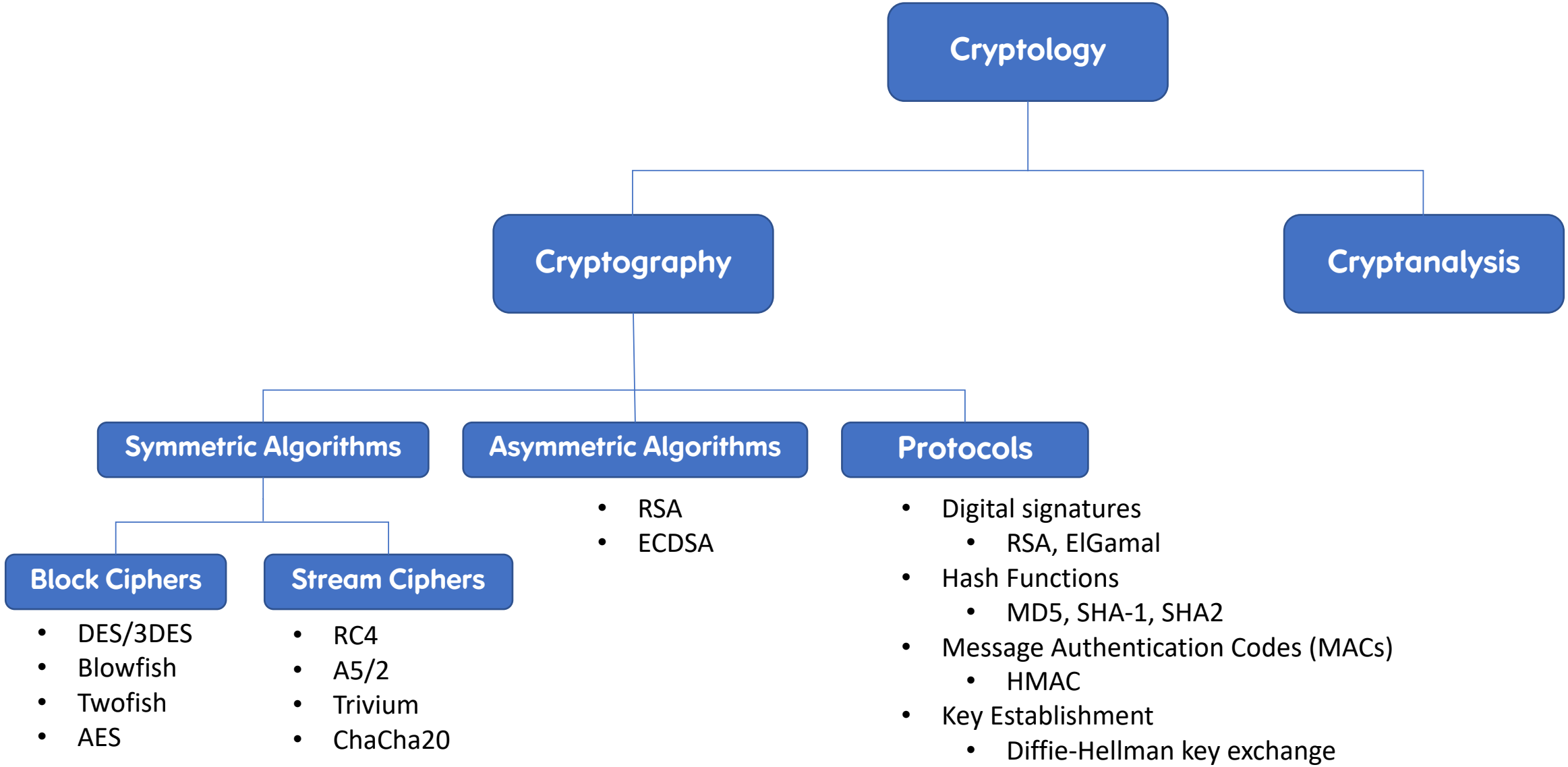


Openssl RSA demo

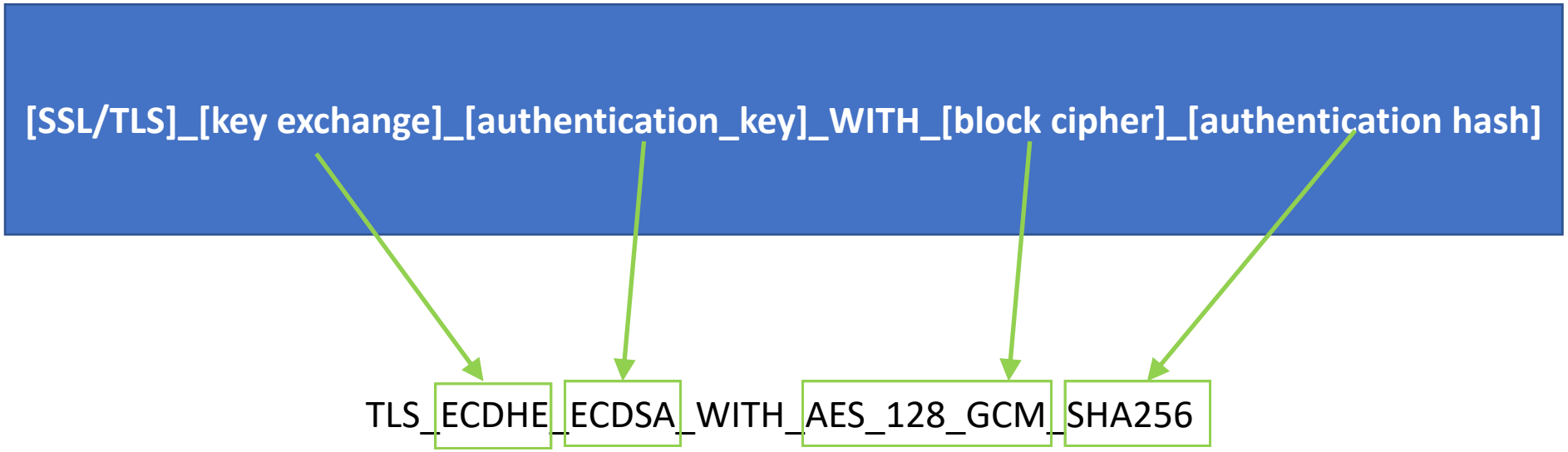
```
victoria@VicMac:~/pkidemo » openssl genrsa
Generating RSA private key, 2048 bit long modulus
...........+++
.............................+++
e is 65537 (0x10001)
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAkI1XnqQz0nQIhWkf39kJWkIwwvCIdG2BucPw6sY8cB6cZvPB
nU3R2Qs2tvgKFBxTJ7eoWZ52tnT/vvbbzBir/yS/nkBLfyPY1dHtHE9jTA4FFLmO
XmCqr7lG/Qw4F42CNWHewWCIT8Dxs2PaNhKpj620yZtJjGKwy/2+BnuBeFhFjuA4
d9OGfRwR3N0+feSCWC+dbXzfzv9MDC3eHqOT65NcFAj572ibdbKEIPSYvKoyYcWk
WQfJJuPFiZNoTzIsZW/ZnEgvW37CkwO0tqUTNeEgsviJuirqAXxiyYnERs9Nm6FY
AGbSa0NwIGiMXEWgwxlFCBj4HYby81pxb0NpVwIDAQABAoIBACbn0+hf247OKij2
JV0TRkAjxooBQUjM3HiKM0KNPmES7BeNjdPvyQY4zW4rjlLhWMQ82zPbwanUDa1x
NbFWBpzyYByQw4Ehvc+247GV8Gzc7SMp2hauv+Hx5RhgfIxiSuRhRhoYCDgm/ybA
PlAFb9u67gjBIeeF/H26+rU18p/YArt/I0IO1rdEZZ5oWeqBrY/7sFJLqankObdq
b7p+mXD5m8l/dbV8w9X+q+BBBHEmHYl2Y8AzM9thcXncgp2hEqGVn+alKO/6ZsmG
eSw3POtF7XALQtEU2oenAIjoA3EQ2MIT6AndFEEpAL7OyQsEnpnipbwVC+1/hX/+
F98jzQkCgYEAwK4u7+I/BoP2RGFecaL0F6nCY7De5lvSQytLTsvmKyxqeq2/fApZ
yO89ps1gcSOsVuKewH71DOnR8g7XpHf6jAZlI6unwnAr1bxdRqGpf+tssFRSZ+EB
6lG7iyUbqEM7nc8B181W60W2wVPxlfCpg/672R8l9CllQLZyvHKzMV0CgYEAwA44
/IRsiJe0Kkw518VzaafkXFnpkyM9AUdYvVEcENiK74xo8GfJakPkI83a0wa385Db
9G6WoZzaX3SqxwvNsKy2mV+15Y0+a9hqvb/8pbrVYJ+K0foF5ZHc2aztTIQpFtDW
NtaNfBfYYrGXx4Bw+EmuMYqZ/aHoRGTSk90llkMCgYB07wDDSl4PeTSSqTjk5A0Y
80hqseOEj8MaSoMIjgvgtbLlQt7Ly4hTvoV+3nMiFpE3pY7Mqf/T1283ZLSQNtQR
+KuWq01DXknsC8Xb4K56WIj8th/QtSgxaWrU9i9DvZHHouKucgrG++b+ixA5e/WU
ipq8GQAynpC5ZkBZh46ibQKBgHF3NaXCmtisP+4JcNcQuZn/bgQ9vqiGViFuRHg8
9kbhmFvkaY820iFtENkyYojwXDeTKtAjkezTXZStucO1LoWl1I+So0WGiA1xI8cN
KTfExRG70GkzvM4fugoEh4IARsJLCXQGmDprYBUW9SQRtyyJY1ezmjzlrfLSWjKv
oeg/AoGBAIradiOplQMmVfgHSl0OkF39cyGPxlHGcX0MoNGd48HZ0nbrc4QT81W/
x1soCzyYTTYzmY4kZLqlrIkAStf7hK/ywxMXy0AZIhwCsP/3t73iPjSoruANr2uo
193QKRFvcmAzuA0McHTJtz87W+13/1G/NGGDkEUkvAvurm3EBlj3
-----END RSA PRIVATE KEY-----
victoria@VicMac:~/pkidemo »
```

Secure connection using the SSL/TLS certificate




Cryptology
Cryptography
Cryptanalysis
Symmetric Algorithms
Asymmetric Algorithms
Protocols
Block Ciphers
Stream Ciphers
DES/3DES
Blowfish
Twofish
AES
RC4
A5/2
Trivium
ChaCha20
RSA
ECDSA
Digital signatures
RSA, ElGamal
Hash Functions
MD5, SHA-1, SHA2
Message Authentication Codes (MACs)
HMAC
Key Establishment
Diffie-Hellman key exchange

HTTPS


[SSL/TLS]_[key exchange]_[authentication_key]_WITH_[block cipher]_[authentication hash]
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256

Protocols

There are five protocols in the SSL/TLS family: SSL v2, SSL v3, TLS v1.0, TLS v1.1, and TLS v1.2:

- SSL v2 is insecure and must not be used. This protocol version is so bad that it can be used to attack RSA keys and sites with the same name even if they are on an entirely different servers (the DROWN attack).
- SSL v3 is insecure when used with HTTP (the POODLE attack) and weak when used with other protocols. It’s also obsolete and shouldn’t be used.
- TLS v1.0 is also a legacy protocol that shouldn't be used, but it's typically still necessary in practice. Its major weakness (BEAST) has been mitigated in modern browsers, but other problems remain.
- TLS v1.1 and v1.2 are both without known security issues, but only v1.2 provides modern cryptographic algorithms.

TLS v1.2 should be your main protocol because it's the only version that offers modern authenticated encryption (also known as AEAD). If you don't support TLS v1.2 today, your security is lacking.

Key exchange

- For the key exchange typically choice is between the classic ephemeral **Diffie-Hellman key exchange (DHE)** and its **elliptic curve variant, ECDHE**.

- There are other key exchange algorithms, but they're generally insecure in one way or another. The RSA key exchange is still very popular, but it doesn't provide forward secrecy.

- **Use Forward Secrecy (perfect forward secrecy)** - is a protocol feature that enables secure conversations that are not dependent on the server's private key. To enable PFS, the client and the server have to be capable of using a cipher suite that utilises the *Diffie-Hellman key exchange*. Importantly, the key exchange has to be *ephemeral*. This means that the client and the server will generate a new set of Diffie-Hellman parameters for each session. These parameters can never be re-used and should never be stored, the ephemeral part, and that's what offers the protection going forwards.

Key exchange – best practice

TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384

- Use only suites that use ECDHE and DHE (also referred to as EECDH and EDH) for the key exchange
- The EC variant is faster
- Both offer Perfect Forward Secrecy (PFS)
- TLSv1.3 will only support PFS capable key exchange

Authentication key algorithm

- Mostly RSA is used as widely supported
- But ECDSA is considerably faster
- Can be served both RSA and ECDSA certificates for the best of both worlds

With RSA certificate:
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

With ECDSA certificate:
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384

With hybrid RSA and ECDSA certificate:
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

Block cipher

- AES is the preferred algorithm and using a key size of 128bits is acceptable
- The GCM segment is the mode of the cipher
- GCM suites should be prioritised over non GCM suites (CBC). An example with the GCM and non-GCM versions of the same suite
- TLSv1.3 will only support AEAD suites

TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
TLS_DHE_RSA_WITH_AES_256_CBC_SHA

Authentication hash

- SHA-256 is now the industry-standard signature hash algorithm for SSL certificates
- SHA-256 provides stronger security and has replaced SHA-1 as the recommended algorithm
- SHA-256 is supported by all current browsers
- There are currently 6 different SHA2 variants including:
  - SHA-224
  - SHA-256
  - SHA-384
  - SHA-512
  - SHA-512/224
  - SHA-512/256

Putting together.. best practices

TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
TLS_DHE_RSA_WITH_AES_128_CBC_SHA
TLS_DHE_RSA_WITH_AES_256_CBC_SHA
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256

http security..

Security Report Summary

Site:	https://forse.no/
IP Address:	191.235.160.13
Report Time:	30 Jan 2018 15:52:36 UTC
Report Short URL:	https://schd.io/5rEC
Headers:	✖ Strict-Transport-Security ✖ Content-Security-Policy ✖ X-Frame-Options ✖ X-XSS-Protection ✖ X-Content-Type-Options ✖ Referrer-Policy

And at the end of a day..

- https://github.com/ssllabs/research/wiki/SSL-and-TLS-Deployment-Best-Practices
- https://scotthelme.co.uk/
- https://securityheaders.io/

- https://www.youtube.com/channel/UC1usFRN4LCMcfIV7UjHNuQg?pbjreload=10