4IZ110: HTTPS, TLS, šifrování

Ing. Michal Dočekal

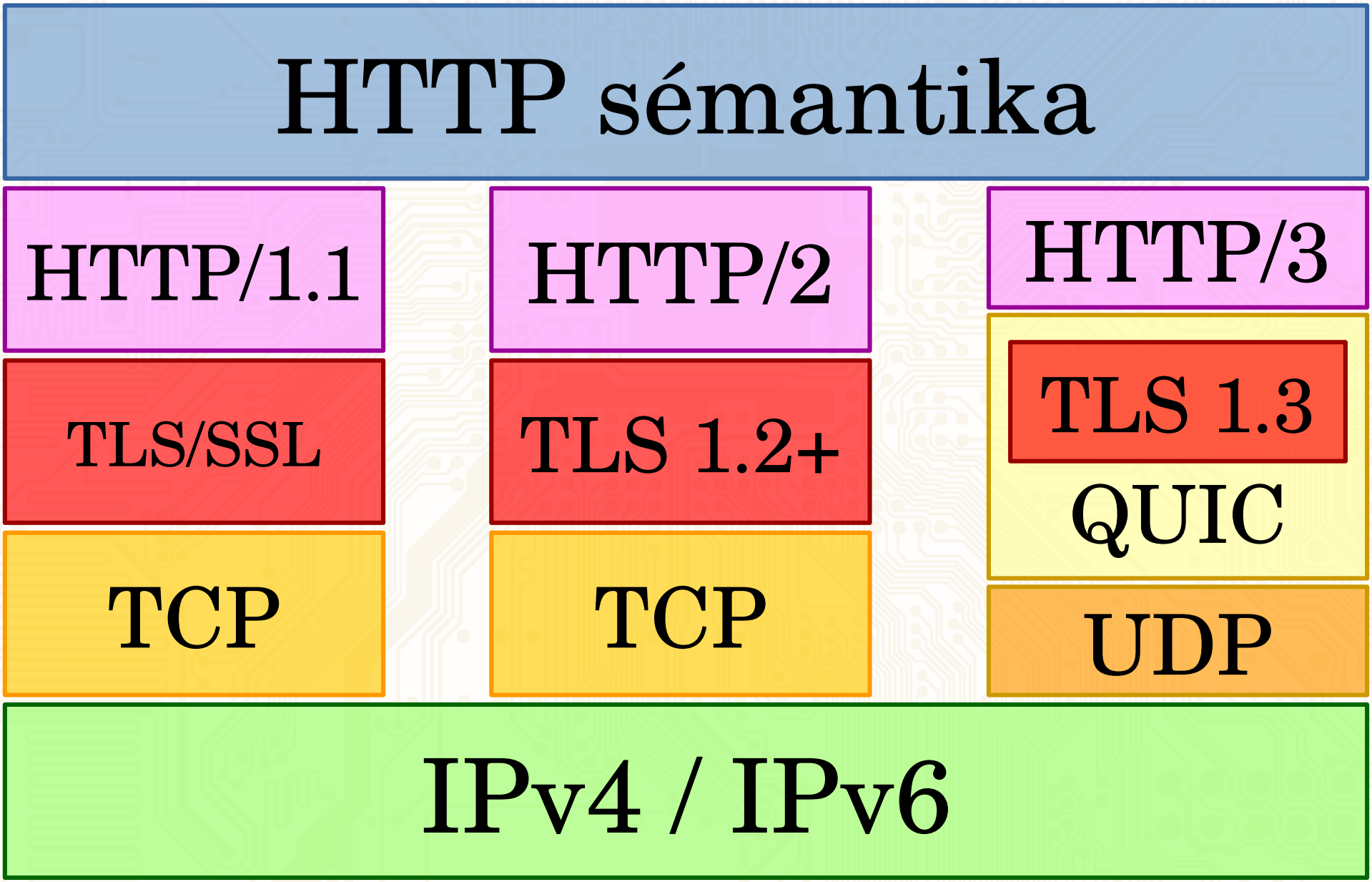
HTTPS

- **H**yper**T**ext **T**ransfer **P**rotocol **S**ecure
- **rozšíření** protokolu **HTTP**
- poskytuje **zabezpečení** ve formě:
  - **šifrování** komunikace
  - **ověření identity** webové stránky (PKI)
  - ochranu **soukromí** a **integrity** dat
  - chrání před MITM útoky, odposlechem a manipulací s přenášenými daty

HTTPS = HTTP over TLS

- protkol **TLS** zajistí vytvoření zabezpečeného spojení, přes které běží standardní **HTTP** protokol
- **HTTPS** využívá **TCP** port **443**
  - až do HTTP/3, pak využívá **QUIC** protokol nad **UDP**

Stacking protokolů



Verze zabezpečovacích protokolů

- **SSL, SSL2, SSL3: Secure Sockets Layer**
  - roky 1994, 1995, 1996
  - předchůdci TLS, již nejsou bezpeční
- **TLS 1.0** – 1999, zavrženo v RFC 8996
- **TLS 1.1** – 2006, zavrženo v RFC 8996
- **TLS 1.2** – 2008, zastaralé, jisté útoky možné
- **TLS 1.3 – 2018, aktuální verze, RFC 8446**

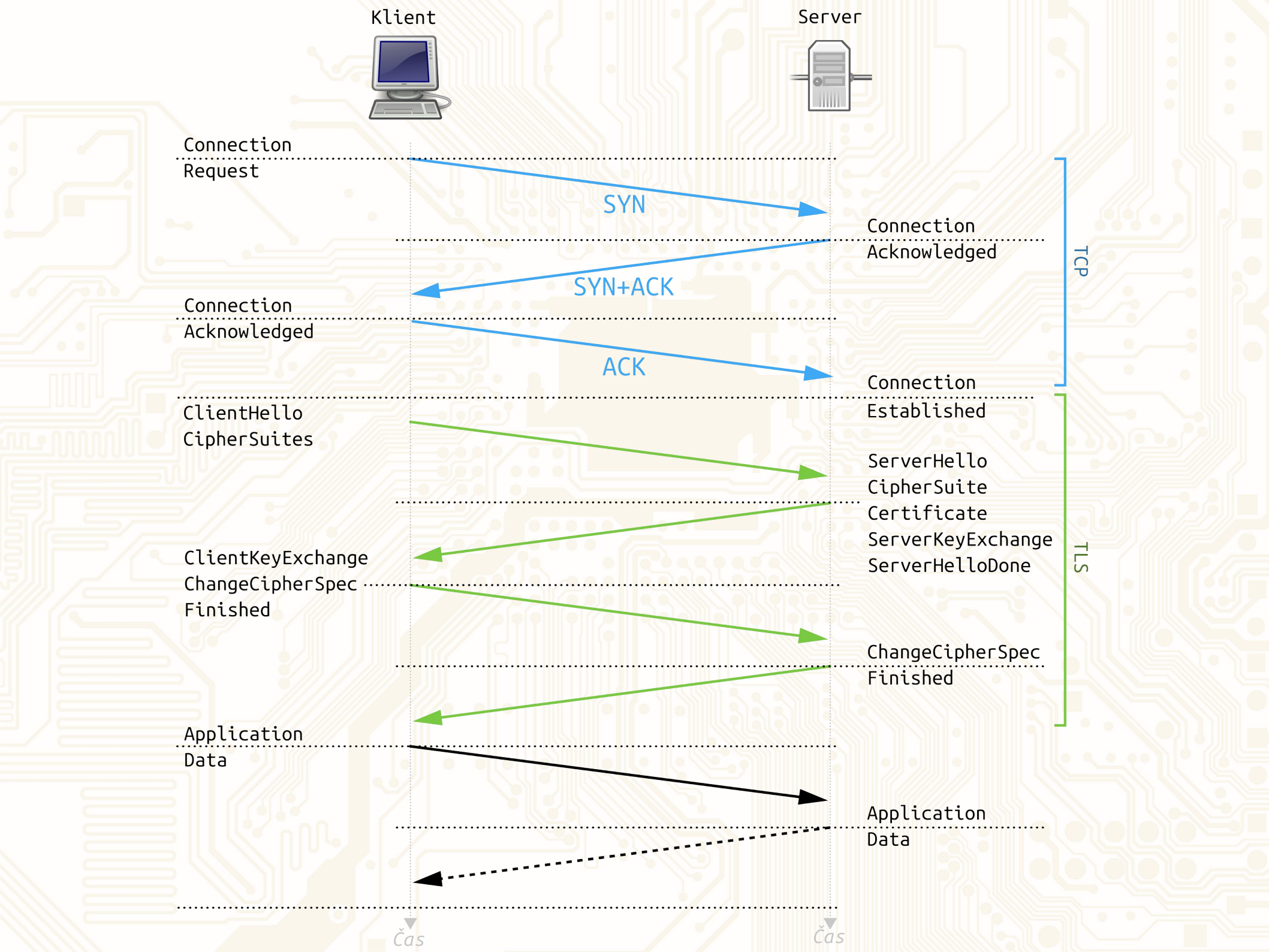
TLS: Transport Layer Security

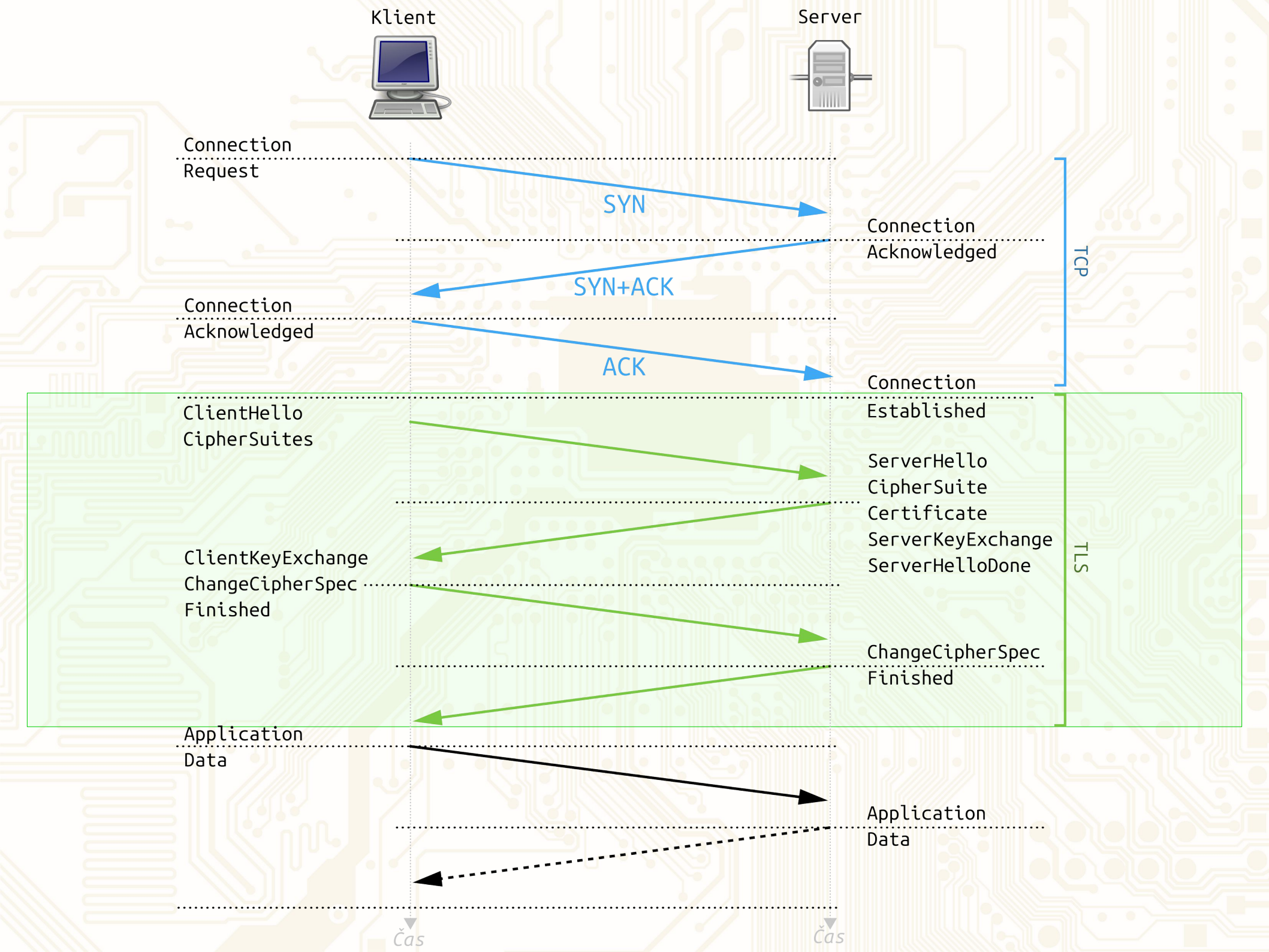
- využívá se k zabezpečení více služeb: **SMTP**, **IM** (instant messaging), **VoIP**, **HTTP**, **OpenVPN**, atd.
- běží v **prezentační** vrstvě OSI modelu
- sám má **TLS** protokol **dvě vrstvy**
  - **TLS handshake**
    - vytvoření zabezpečeného spojení
  - **TLS záznam** (record)
    - jednotka výměny dat (aplikační data + TLS zprávy)
    - udržování a zajištění integrity zabezpečeného spojení

aplikační
prezentační
relační
transportní
síťová
linková
fyzická

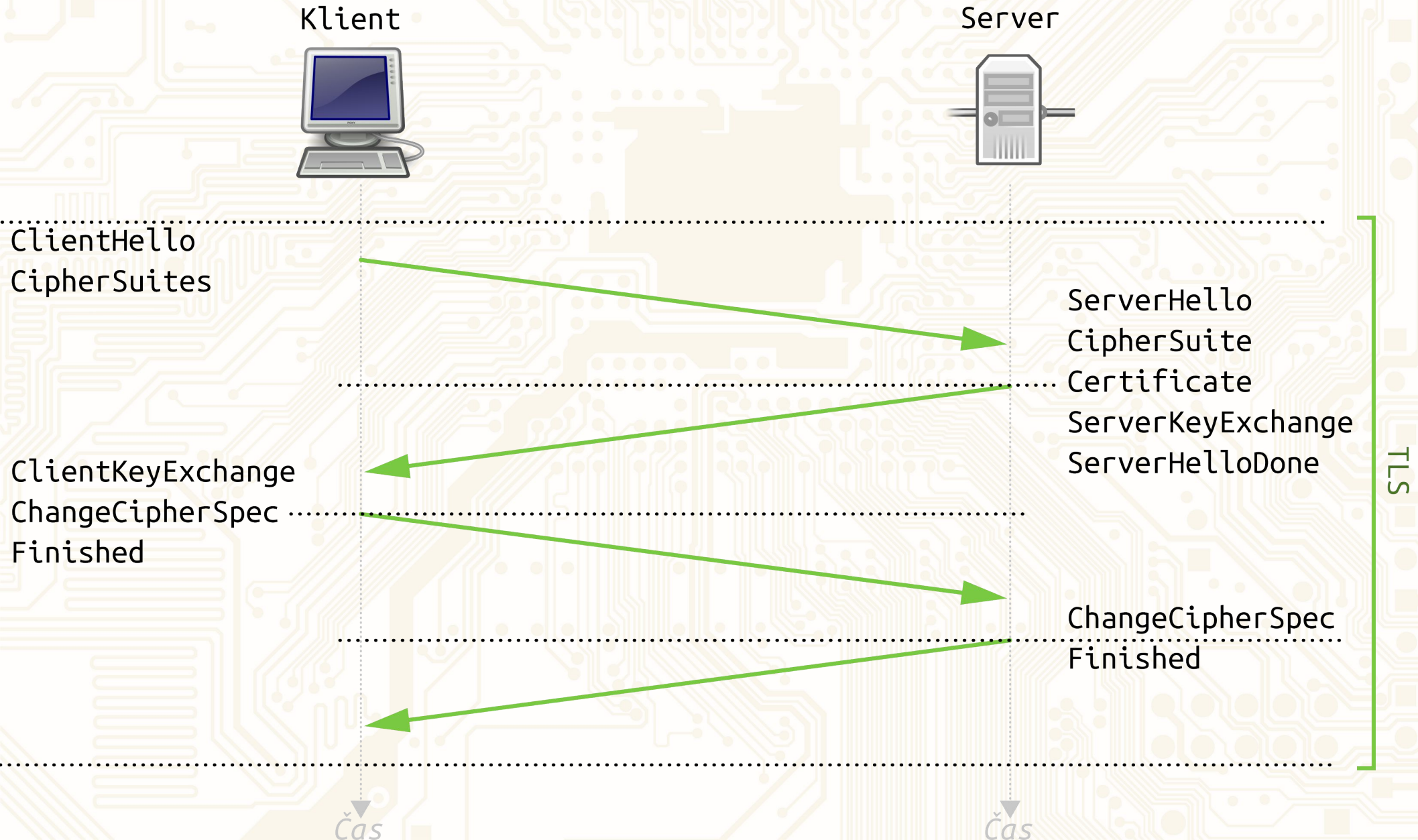
TLS Handshake

1. obě strany se domluví na šifrovacích metodách a verzi protokolu TLS
2. server se autentizuje pomocí certifikátu (klient může také, ale u většiny webů se to nepoužívá)
3. bezpečná výměna klíčů pro šifrování a zajištění integrity komunikace
4. zahájení šifrované a autentizované komunikace


Klient
Server
Connection
Request
SYN
Connection
Acknowledged
SYN+ACK
Connection
Acknowledged
ACK
Connection
Established
TCP
ClientHello
CipherSuites
ServerHello
CipherSuite
Certificate
ServerKeyExchange
ServerHelloDone
ClientKeyExchange
ChangeCipherSpec
Finished
ChangeCipherSpec
Finished
TLS
Application
Data
Application
Data
Čas
Čas


Klient
Server
Connection
Request
SYN
Connection
Acknowledged
SYN+ACK
Connection
Acknowledged
ACK
Connection
Established
TCP
ClientHello
CipherSuites
ServerHello
CipherSuite
Certificate
ServerKeyExchange
ServerHelloDone
ClientKeyExchange
ChangeCipherSpec
Finished
ChangeCipherSpec
Finished
TLS
Application
Data
Application
Data
Čas
Čas

TLSv1.2 Handshake



TLSv1.2 Handshake

- navázání TCP spojení
  - SYN → SYN/ACK → ACK

1. klient pošle **ClientHello** zprávu
   - nejvyšší podporovaná verze TLS protokolu
   - seznam doporučených šifrovacích a *kompresních* metod – komprese byla vzhledem k objeveným zranitelnostem vyřazena a zakázána v TLSv1.3

TLSv1.2 Handshake

2. server pošle
   - **ServerHello** zprávu
     - zvolená verze protokolu (nejvyšší podporovaná oběma stranami)
     - zvolená šifrovací a kompresní metoda
   - **Certifikát** pro ověření identity
   - **ServerKeyExchange** zprávu (výměna klíčů)
   - **ServerHelloDone** zprávu ukončující handshake domluvu

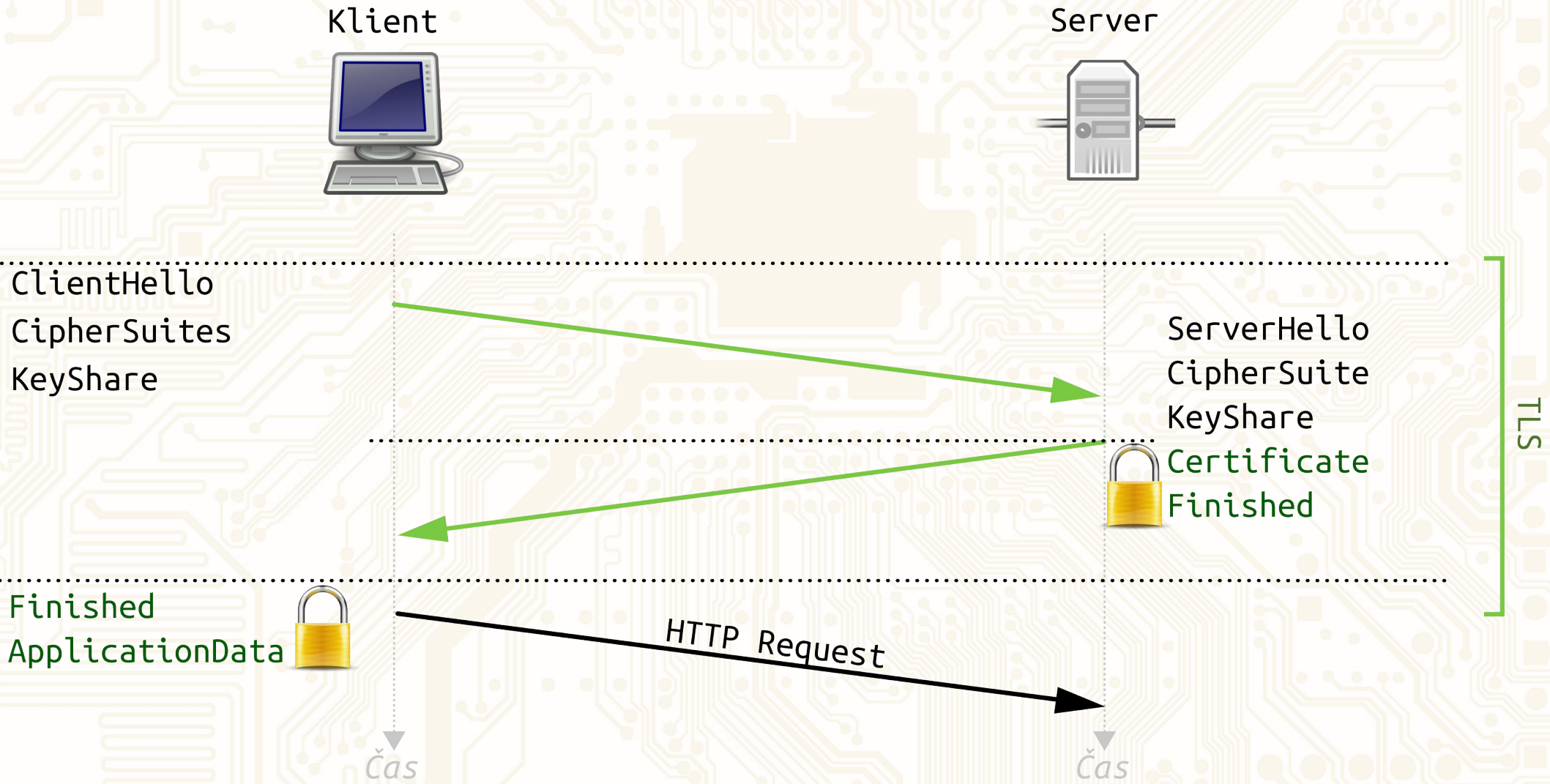
TLSv1.2 Handshake

3. klient odpoví
   - **ClientKeyExchange** zprávou (výměna klíčů)
   - **ChangeCipherSpec** zprávou, která instruuje server, že od teď se bude autentikovat a šifrovat
   - autentikovanou a zašifrovanou **Finished** zprávou
4. server pošle
   - **ChangeCipherSpec** zprávu, která instruuje klienta, že od teď se bude autentikovat a šifrovat
   - autentikovanou a zašifrovanou **Finished** zprávou

TLSv1.3 Handshake

- nešlo by to celé rychleji?
- TLSv1.2 Handshake
  - 3 pakety uvozující TCP spojení
  - 4 pakety TLS handshake
  - až potom lze přenášet data

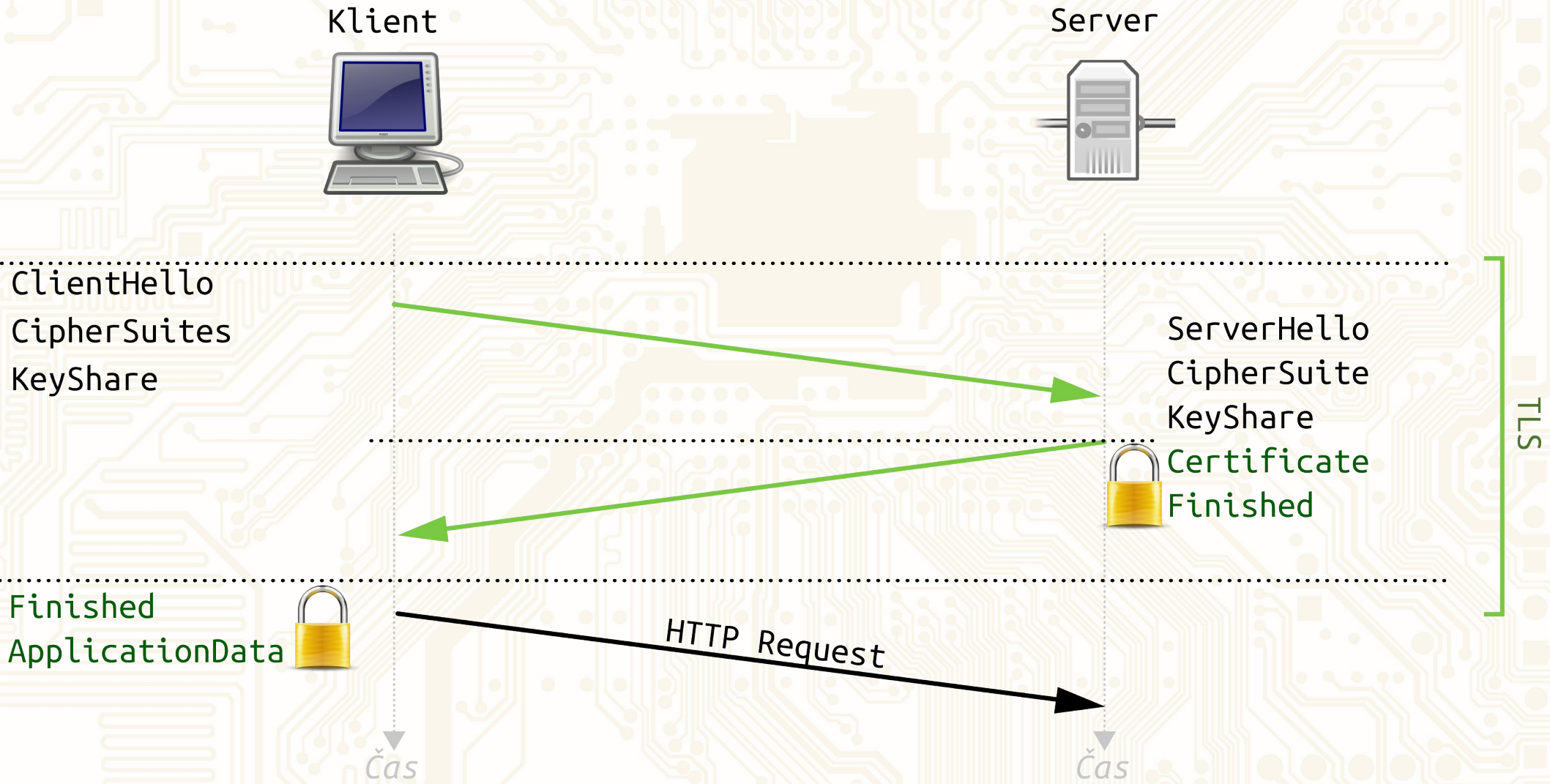
TLSv1.3 Handshake



TLSv1.3 Handshake

- nešlo by to celé rychleji?
- TLSv1.2 Handshake
  - 3 pakety uvozující TCP spojení
  - 4 pakety TLS handshake
  - až 8. paket může přenášet data
- TLSv1.3 Handshake
  - 3 pakety uvozující TCP spojení
  - 3 pakety TLS handshake, ale 3. paket už nese data
  - už 6. paket může přenášet data

TLSv1.3 Handshake



TLSv1.3 Handshake: ClientHello

- **CipherSuites**: klientem podporované šifrovací sady
- **KeyShare**: výměna klíčů začíná ještě před dohodou o použitých algoritmech – klient „hádá", jaké metody bude server preferovat, může dokonce poslat více návrhů pro výměnu klíčů
  - pokud by server nic z toho nepodporoval, pošle server HelloRetry zprávu s vlastním návrhem pro výměnu klíčů

TLSv1.3 Handshake: ServerHello

- **CipherSuite**: zvolená šifrovací sada
- **KeyShare**: dokončení výměny klíčů
- z těchto informací již klient sestaví klíče a může dešifrovat zbytek zprávy, který obsahuje:

Certificate: certifikát serveru k ověření klientem

Finished: zpráva serveru, že skončil handshake

TLSv1.3 ApplicationData

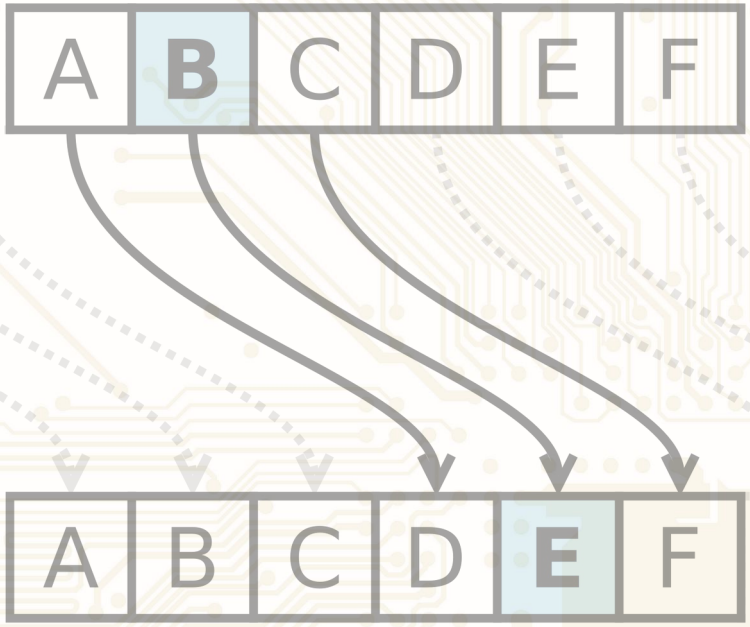
- na ServerHello zprávu odpovídá klient již šifrovaně

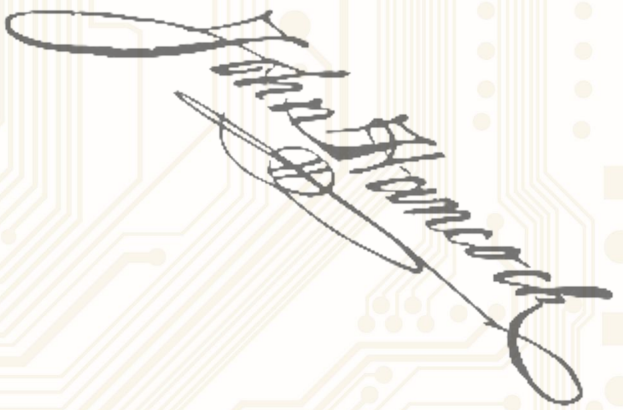
Finished: dokončení výměny klíčů

ApplicationData: zašifrované HTTP, tedy v tomto případě první HTTP request

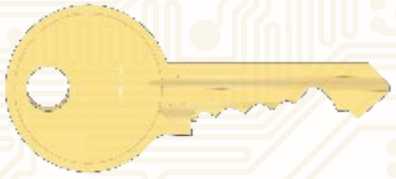
TLS Record

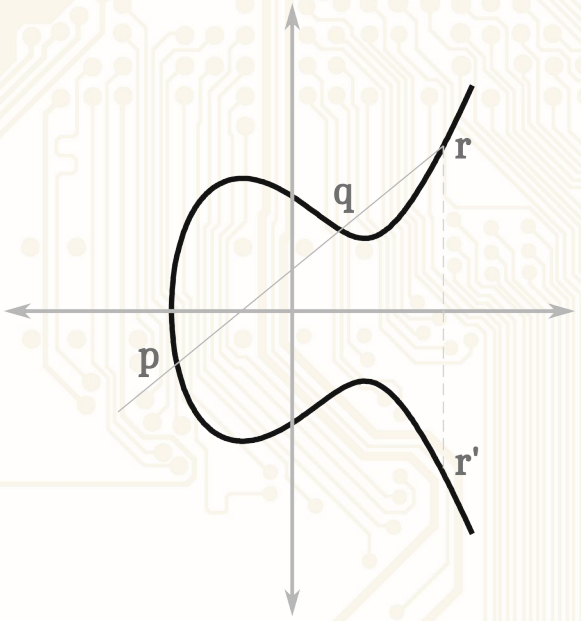
- Handshake protokol je zabalen do **Record Layer**
- Record Layer řídí **přenos a zabezpečení záznamů**
  - přijímá zprávy k přenosu, rozdělí data do zpracovatelných bloků, zabezpečuje záznamy a odesílá je
  - přijatá data jsou ověřena, dešifrována, znovu sestavena a doručena aplikační vrstvě
- kromě aplikačních dat a Handshaku přenáší Record protokol i **hlášení** (alert)
  - např. ukončování spojení musí vyvolat alert





Šifrování





b_0	b_4	b_8	b_{12}
b_1	b_5	b_9	b_{13}
b_2	b_6	b_{10}	b_{14}
b_3	b_7	b_{11}	b_{15}

Kryptografie

- věda o způsobech utajování smyslu informací převodem do podoby, která je čitelná jen se speciální znalostí (šifrovacího klíče)

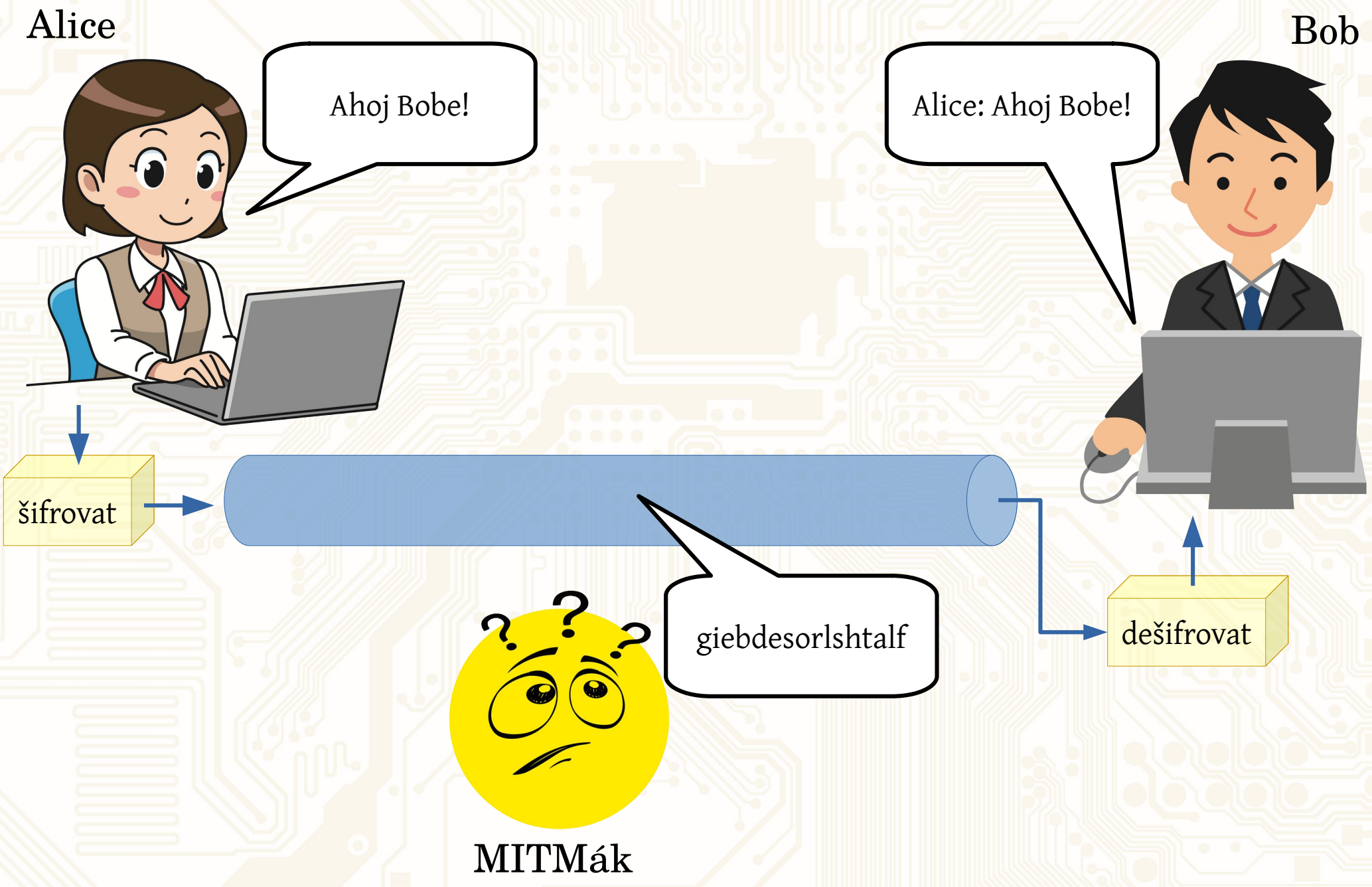
Kryptoanalýza

- věda zabývající se metodami získávání obsahu šifrovaných informací bez přístupu ke speciální znalosti (klíči)

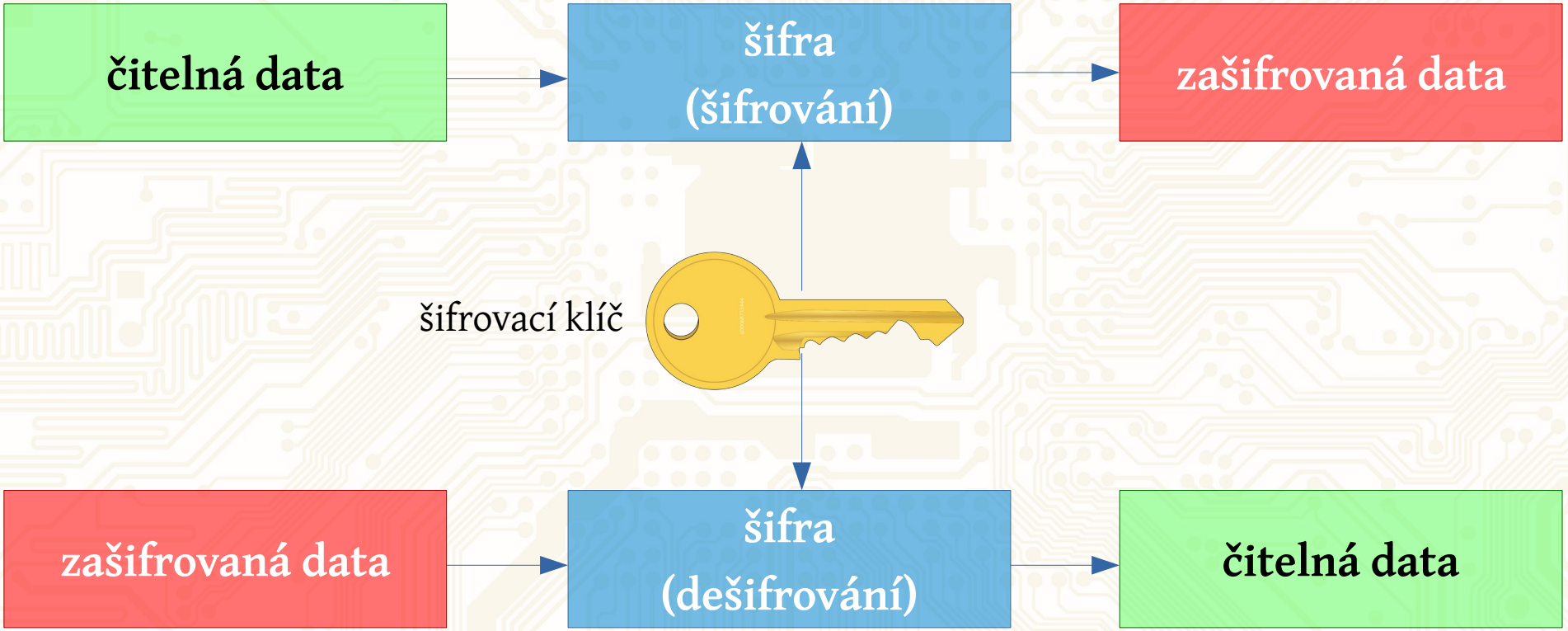
Kryptografie: základní termíny

- **plain text** (prostý text) – nezabezpečená data
- **cipher text** (šifrový text) – zašifrovaná data
- **šifra** – algoritmus, který převádí čitelná data za pomocí *klíče* na nečitelná (zašifrovaná)
- **dešifrování** – převod zašifrovaných dat na čitelná
- **klíč** – určuje transformaci čitelných dat na zašifrovaná
- **kryptosystém** – ucelená sada kryptografických algoritmů zajišťující konkrétní bezpečnostní službu

Šifrovaná komunikace



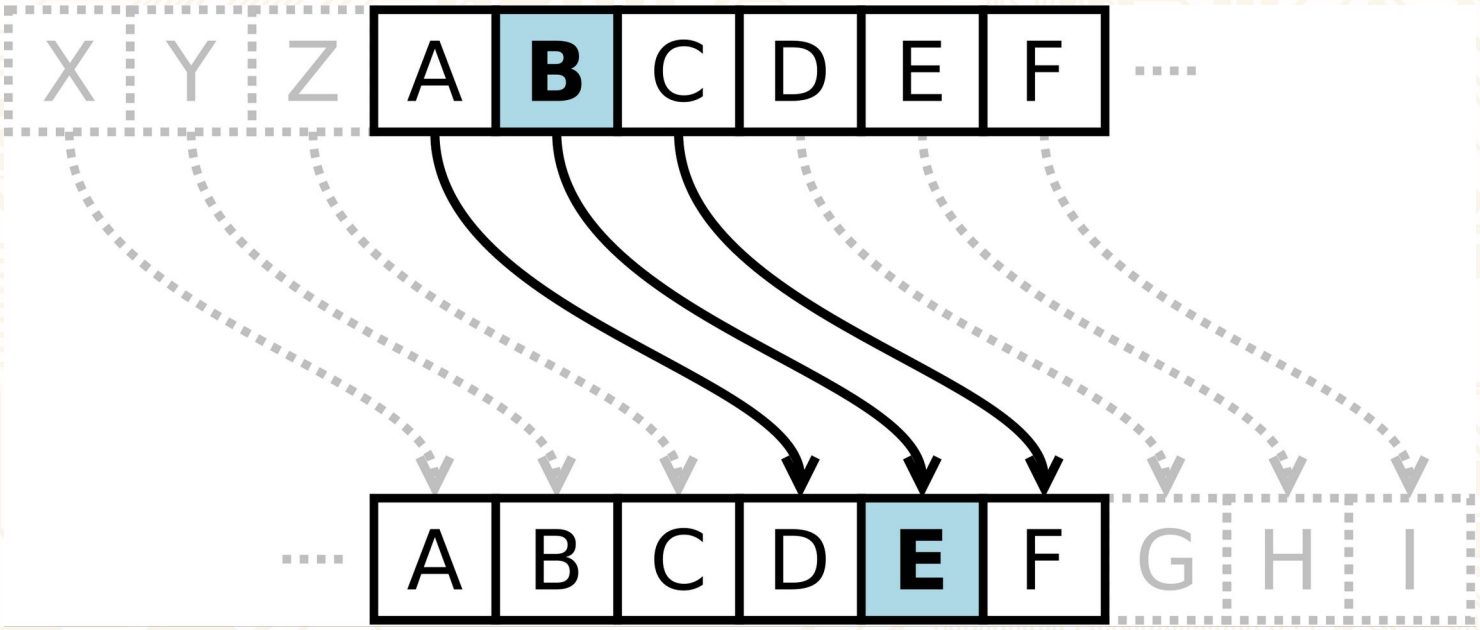
Symetrické šifrování



- pro šifrování a dešifrování je použit jeden a ten samý klíč

Caesarova šifra (57 př. n. l.)

- substituční šifra (záměna jedné množiny symbolů za jiné)
- posun každého písmena v pořadí abecedy o daný počet kroků (zde o 3:)



Caesarova šifra (příklad)

- Čistý text: AHOJ SVETE
- Klíč: 5
- Šifrový text: FMTO XAJYJ

Caesarova šifra

- `Čistý text:  AHOJ SVETE`
- `Klíč:  5`
- `Šifrový text: FMTO XAJYJ`

- používal Julius Caesar, hlavní slabinou je shoda čisté a zašifrované varianty konkrétních znaků
- zranitelné vůči **frekvenční analýze** a **útoku hrubou silou** (klíčem je číslo 1-26)

Moderní šifrovací algoritmy

- libovolný kus dat je v podstatě číslo
- šifry používají matematické metody a postupy
- šifry jsou implementovány softwarově
  - AES má HW akcelerátor v moderních procesorech
- využívají se postupy, kde cesta jedním směrem je podstatně jednodušší než cesta opačným směrem (*rozklad na prvočísla*, *diskrétní logaritmus*, atd.)
- **šifrový text by měl být neodlišitelný od náhodně vygenerovaných dat**

Entropie

- míra neurčitosti systému, „náhodnost"
- informační entropie (Shannonova entropie)
  - maximální pro rovnoměrné rozdělení, minimální pro zcela deterministický systém
  - smysluplná data: střední až nižší entropie
  - náhodná, zašifrovaná, komprimovaná data: vyšší entropie

Plain text

Once upon a midnight dreary, while I pondered, weak and weary,
Over many a quaint and curious volume of forgotten lore—
While I nodded, nearly napping, suddenly there came a tapping,
As of some one gently rapping, rapping at my chamber door.
"'Tis some visiter," I muttered, "tapping at my chamber door—
Only this and nothing more."

Ah, distinctly I remember it was in the bleak December;
And each separate dying ember wrought its ghost upon the floor.
Eagerly I wished the morrow;—vainly I had sought to borrow
From my books surcease of sorrow—sorrow for the lost Lenore—
For the rare and radiant maiden whom the angels name Lenore—
Nameless here for evermore.

The Raven, Edgar Allan Poe

```
Shannon entropy: 4.5145005560080955
```

Once upon a midnight dreary, while I pondered, weak and weary,
Over many a quaint and curious volume of forgotten lore—
While I nodded, nearly napping, suddenly there came a tapping,
As of some one gently rapping, rapping at my chamber door.
"'Tis some visiter," I muttered, "tapping at my chamber door—
Only this and nothing more."

Ah, distinctly I remember it was in the bleak December;
And each separate dying ember wrought its ghost upon the floor.
Eagerly I wished the morrow;—vainly I had sought to borrow
From my books surcease of sorrow—sorrow for the lost Lenore—
For the rare and radiant maiden whom the angels name Lenore—
Nameless here for evermore.

The Raven, Edgar Allan Poe

Shannon entropy: 4.5145005560080955

Once upon a midnight dreary, while I pondered, weak and weary,
Over many a quaint and curious volume of forgotten lore—
While I nodded, nearly napping, suddenly there came a tapping,
As of some one gently rapping, rapping at my chamber door.
"'Tis some visiter," I muttered, "tapping at my chamber door—
Only this and nothing more."

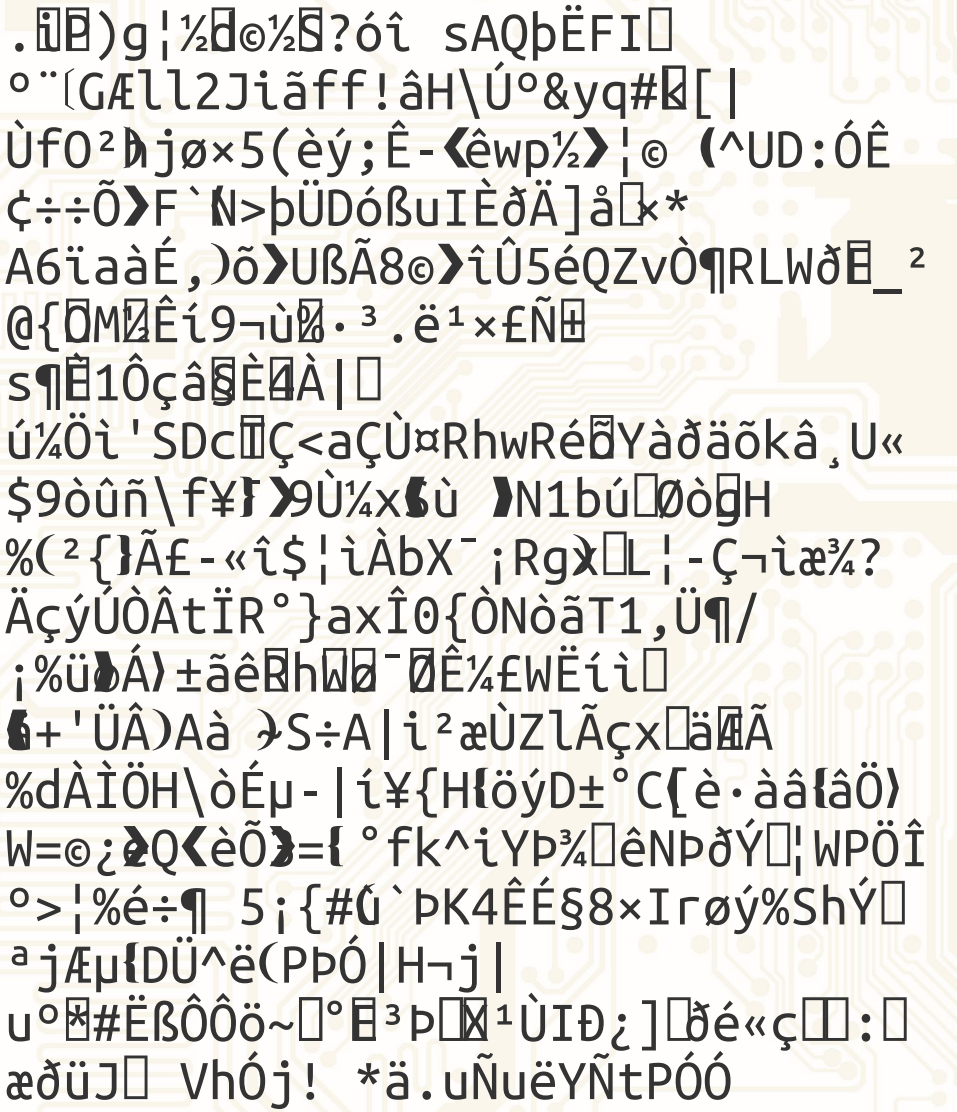
Ah, distinctly I remember it was in the bleak December;
And each separate dying ember wrought its ghost upon the floor.
Eagerly I wished the morrow;—vainly I had sought to borrow
From my books surcease of sorrow—sorrow for the lost Lenore—
For the rare and radiant maiden whom the angels name Lenore—
Nameless here for evermore.

AES Encrypt

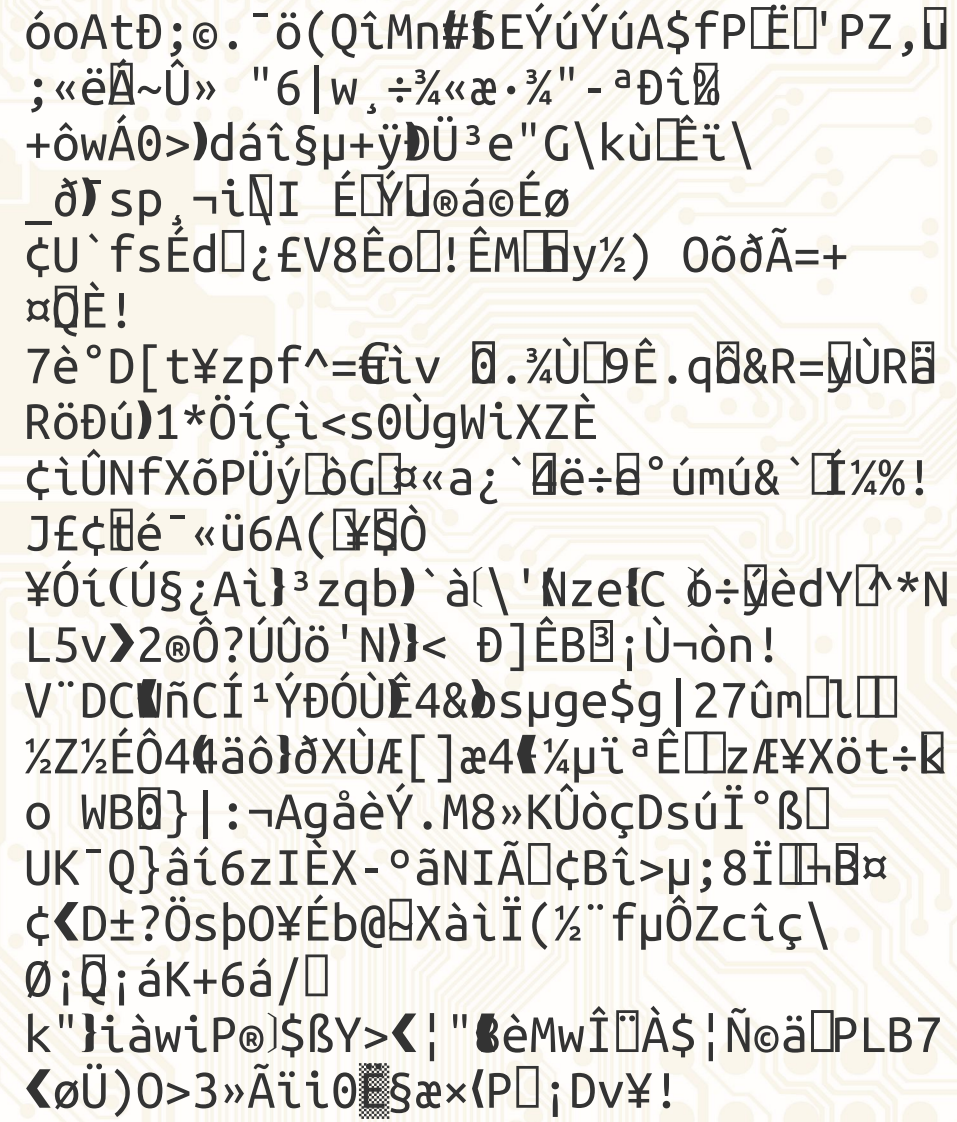
Key		IV	
jksadhnwtumkdi…	LATIN1	53c8975399f427b2…	HEX

Mode	Input	Output
GCM	Raw	Raw

Náhodná data / šifrový text?



vzorek 1



vzorek 2

Shannon entropy: 7.7326250085529615

vzorek 1

.ﬕÞ)g¦½▯©½S?óî sAQþËFI°¨▯GÆll2Jiãff!âH\Ú°&yq#▯[|ÙfO²▯jø×5(èý;Ê-▯
êwp½❮❯¦©(^UD:ÓÊ¢÷÷Õ❯F`N>þÜDóßuIÈðÄ]å▯×*
A6ïaàÉ,)õ❯UßÃ8©❯îÛ5éQZvÒ¶RLWðE_²@{▯M▯Êí9¬ù▯·³.ë¹×£Ñ▯ s¶▯1ÔçâSÈ▯À|▯
ú¼Öì'SDc▯Ç<aÇÙ¤RhwRé❯Yàðäõkâ¸U«$9òûñ\f¥❮❯9Ù¼x▯Sù▯N1bú▯Øò▯H
%(²{❯Ã£-«î$¦ìÀbX¯¡Rg❯x▯L¦-Ç¬ìæ¾?ÄçýÚÒÂtÏR°}axÎ0{ÒNòãT1,Ü¶/
¡%ü❯Á❯±ãê❮h❯❯¯▯Ê¼£WËíì▯+'ÜÂ▯▯Aà▯S÷A|i²æÙZlÃçx▯ä▯Ã%dÀÌÖH\òÉµ-|í¥
{H❮öýD±°C❮è·àâ❮âÖ❯W=©¿ ❯Q❮èÕ❯=❮°fk^iYÞ¾▯êNÞðÝ▯¦WPÖÎ°>¦%é÷¶
5¡{#Ú`ÞK4ÊÉ§8×Irøý%ShÝ❮ªjÆµ(DÜ^ë▯PÞÓ|H¬j|
u°❯#ËßÔÔö~❯ ❮³ÞK¹ÙIÐ¿]▯ ðé«ç▯▯:▯æðüJ▯ VhÓj! *ä.uÑuëYÑtPÓÓ

vzorek 2

óoAtÐ;©.¯ö(QîMn#▯EÝúÝúA$fP▯Ë▯'PZ,▯;«ë▯~Û» "6|w¸÷¾«æ·¾"-ªÐî▯
+ôwÁ0>❯dáî§µ+ÿ❯Ü³e"G\kù▯ï_ð ❯sp¸¬i❯I É▯Ý▯®á©Éø¢U`fsÉd▯¿£V8Êo▯!
ÊM(▯y½) OõðÃ=+¤QÈ!7è°D[t¥zpf^=❯ìv ▯.¾Ù▯9Ê.q▯&R=▯ÙR▯RöÐú▯1*ÖíÇì<s0ÙgWiXZÈ
¢ìÛNfXõPÜý(òG❯¤«a¿`4ë÷è°úmú&`▯Í¼%!J£¢▯é¯«ü6A(▯¥▯Ò
¥Óí(Ú§¿Aì❯³zqb❯`à(\'Nze❮C ð÷❯èdY❯^*NL5v❯2®Ô?ÚÛö'N▯▯< Ð]ÊB▯¡Ù¬òn!
V¨DC❮ñCÍ¹ÝÐÓÙ▯4&❯sµge$g|27ûm▯l▯▯
½Z½ÉÔ4❮äô❯ðXÙÆ[]æ4❮¼µï³Ê▯▯zÆ¥Xöt÷▯o WB▯}|:¬Agåè Ý.M8»KÛòçDsúÏ°ß▯
UK¯Q}âí6zIÈX-ºãNIÃ❮¢Bî>µ;8Ï❯¬▯¤¢▯D±?ÖsÞO¥Éb@▯Xàì Ï(½¨fµÔZcîç\Ø¡▯¡áK+6á/▯

Shannon entropy: 7.663401155185399

Shannon entropy: 7.7326250085529615

vzorek 1

šifrový text

vzorek 2

náhodná data

Shannon entropy: 7.663401155185399

Symetrické šifrování: příklad

Čistý text: KILROY WAS HERE

Hex: 4b49 4c52 4f59 2057 4153 2048 4552 4500

Klíč: SIFROVANI RULEZ.

Šifra: AES (Rijndael)

šifrování

Šifrový text: ýCÆ ¶ ; x"ÅS

Hex: fd1a 43c6 9211 b698 3b96 9878 22c5 a79c

Symetrické šifrování: příklad

Šifrový text: ýCÆ ¶ ; x"ÅS

Hex: fd1a 43c6 9211 b698 3b96 9878 22c5 a79c

Klíč: SIFROVANI RULEZ.

Šifra: AES (Rijndael)

dešifrování

Čistý text: KILROY WAS HERE

Hex: 4b49 4c52 4f59 2057 4153 2048 4552 4500

Přehozený bit v klíči

Šifrový text: ýCÆ ¶ ; x"ÅS

Hex: fd1a 43c6 9211 b698 3b96 9878 22c5 a79c

Klíč: SIFROVANI RULDZ.

Šifra: AES (Rijndael)

dešifrování

Čistý text: ûHO)æã¢¯ª9 <Rr

Hex: fb48 4f1b 8be6 e3a2 afaa 390e 003c 5272

Src: 4b49 4c52 4f59 2057 4153 2048 4552 4500

Algoritmy a utajení

- snaha tajit algoritmy samotné byla překonána, dnes považujeme za bezpečné výhradně ty zveřejněné
- **Kerckhoffův princip** (19. století): bezpečnost kryptosystému má záviset na utajení klíče, ne algoritmu
- spoléhat na utajení algoritmu je **security by obscurity** *(bezpečnost skrze neznalost)* a špatný bezpečnostní postup, lepší je **secure by design**

„Kdokoliv může vynalézt bezpečnostní systém tak chytrý, že si sám neumí představit, jak ho zlomit. Bezpečné jsou ty, které neumí zlomit nikdo jiný ani po mnoha letech pokusů." – Bruce Schneier

Úvod do kryptoanalýzy

- **útoky na šifru**: zlomení nebo oslabení algoritmu
- **útoky na klíč**:
  - **brute force** – zkoušení všech možností
  - **slovníkový útok** – databáze slov, nebo ještě častěji databáze nejčastějších hesel nebo kombinací loginů a hesel
  - **oslabení klíče + brute force**
    - např. víme, že v klíči je slovo „Jan", má 8 znaků a 2 z toho jsou čísla

Úvod do kryptoanalýzy

- **útoky na implementaci**: i nejlepší šifrovací algoritmy implementované špatně mohou útočníkovi umožnit zlomit šifrování – WEP, SSL, starší TLS
  - **CVE-2008-0166**: OpenSSL knihovna omylem modifikována vývojáři Debianu tak, že omezila zdroj náhodnosti pro generování klíčů na 32-bitové číslo, tedy jen 4×10^9 variant místo alespoň 10^{308})
  - **zadní vrátka (backdoor)**: má-li kryptosystém zadní vrátka, stačí, aby na ně někdo přišel
    - utajený mechanismus umožňující získat přístup k danému systému (nejen u kryptografie)
    - např. dešifrovací klíč nenápadně připojený k datům
    - Dual_EC_DRBG – CSPRNG „upravený“ NSA

Úvod do kryptoanalýzy

- **útoky postranním kanálem**: kryptoanalýza za pomoci informací zjištěných z provozu kryptosystému původně nepředpokládaným způsobem
  - např. délka operace, zatížení procesoru, spotřeba energie, apod. nám mohou prozradit informace vedoucí k prolomení nebo oslabení šifrování
- **sociální inženýrství**: manipulace oběti s cílem obejít zabezpečení (a získat heslo)
- **hadicová kryptoanalýza**: použití psychického či fyzického nátlaku, aby oběť prozradila heslo

Co pomáhá kryptoanalýze?

1. nejtěžší situace: máme jen šifrový text
2. ke konkrétnímu šifrovému textu známe **čistý text**
3. námi **zvolený čistý text** necháme zašifrovat
   a dostaneme se k odpovídajícímu šifrovému textu
4. námi **zvolený šifrový text** necháme dešifrovat
   a dostaneme se k odpovídajícímu čistému textu
5. máme k dispozici body 3. a 4.

Frekvenční analýza

- slova a jazyk mají svoje specifické vzory, které se u substitučních šifer mohou projevovat i v šifrovém textu *(např. v angličtině je nejčastěji zastoupeno písmeno E)*
- na základě frekvenční analýzy (analýzy četnosti konkrétních znaků) pak lze dešifrovat konkrétní písmena, slova až celý text
- závislé na použité abecedě a jazyce

Útok hrubou silou (brute force)

- kryptoanalytická metoda, kde postupně zkoušíme všechny varianty klíče, náročné na výkon i čas
- reálné u **krátkých klíčů** nebo při **oslabení šifry**
- u **silného hesla** a dnešních šifrovacích algoritmů nereálné (128-bitový klíč: 2^{128} variant: $3x10^{38}$)

  340282366900000000000000000000000000
- existují metody zvyšující nároky na brute force a zesilující slabá hesla (např. PBKDF2, Argon2)
- u služeb (např. webových) mnohočetné zadávání hesla v krátké době za sebou bývají dekována a zastavena

Slovníkový (dictionary) útok

- testují se **pouze pravděpodobná hesla** z předpřipraveného seznamu
- v reálném světě **velmi časté** typy útoků
- relativně **snadné** a **nenáročné** na provedení
- obranou je **silné heslo**
- naopak problémem jsou zapomenuté testovací účty (uživatel *test*, heslo *test*) nebo ledabyle nakonfigurovaná zařízení (uživatel *admin*, heslo *admin*) či ponechání uživatele a hesla od výrobce
- na služby lze útočit nenápadně (dlouhý čas mezi pokusy)

Diferenciální kryptoanalýza

- nauka o tom, jak odlišnosti v informačním vstupu mohou ovlivnit výsledek kryptosystému na výstupu, tedy jak se např. změna 1 bitu šíří algoritmem a jak se projeví
- hledá se, kde se algoritmus nechová nahodile
- optimálně není možné zjistit, kde se změna projeví
- pokud to možné zjistit je, může to pomoci oslabit nebo prolomit šifrování

Vliv chyb na šifrování (AES)

Šifrový text: ýCÆ ¶ ; x"ÅS

Hex: fd1a 43c6 9211 b698 3b96 9878 22c5 a79c

Klíč: SIFROVANI RULEZ.

Šifra: AES (Rijndael)

dešifrování

Čistý text: KILROY WAS HERE

Hex: 4b49 4c52 4f59 2057 4153 2048 4552 4500

Přehozený bit v šifrovém textu

Šifrový text: ýCÆ ¶ ; y"ÅS

Hex: fd1a 43c6 9211 b698 3b96 9879 22c5 a79c

Klíč: SIFROVANI RULEZ.

Šifra: AES (Rijndael)

dešifrování

Čistý text: ¼ ý_Ì°Újh á ¹Ä

Hex: 9d9a bc88 fd5f ccb0 da6a 6887 e18b b9c4

Src: 4b49 4c52 4f59 2057 4153 2048 4552 4500

Typy symetrických šifer

- **bloková (block)**: šifruje se po blocích
  - zástupci šifer: **Rijndael** (AES), **Serpent**, **Twofish**
  - AES šifruje po blocích 128 bitů = 16 bytů
  - stejný blok na vstupu povede ke stejnému bloku na výstupu → šifrovací režimy (mode of operation)
- **proudová (stream)**: šifruje se po bitech
  - **keystream**: pseudonáhodný generátor čísel založený na hesle, XORuje se s čistým textem
  - zástupce: **ChaCha20**

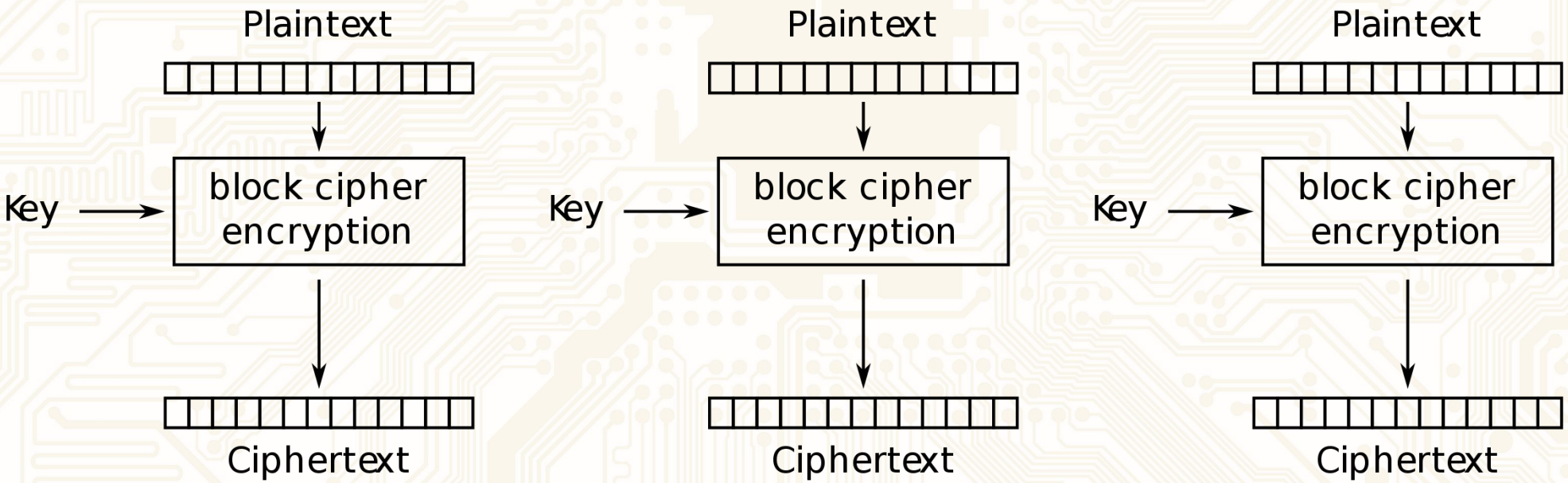
Mode of operation

- **provozní režim** blokových šifer
  - jak se šifrují jednotlivé bloky při šifrování většího množství dat než kolik se vejde do 1 bloku
  - jak se data vyplní do bloku, pokud ho nejsme schopni vyplnit celý
- cílem je udržet bezpečnost zašifrovaných dat a zabránit kryptoanalýze
- zástupci: ECB, CBC, CFB, CTR, GCM, SIV,…

ECB: Electronic Code Book

- nulový režim, každý blok se šifruje samostatně
- stejná vstupní data v dalším bloku povedou ke stejnému výstupnímu zašifrovanému bloku
  - silně napomáhá kryptoanalýze, protože často umožňuje odhadnout, co mohlo být ve vstupních datech
  - žádná autentizace dat, možno podvrhnout blok dat útočníkem
- jakýkoliv moderní kryptosystém se snaží zajistit, aby každý blok šifrového textu byl jiný, a bránit se změnám šifrovaných dat nebo jejich podvržení

ECB: Electronic Code Book



Electronic Codebook (ECB) mode encryption

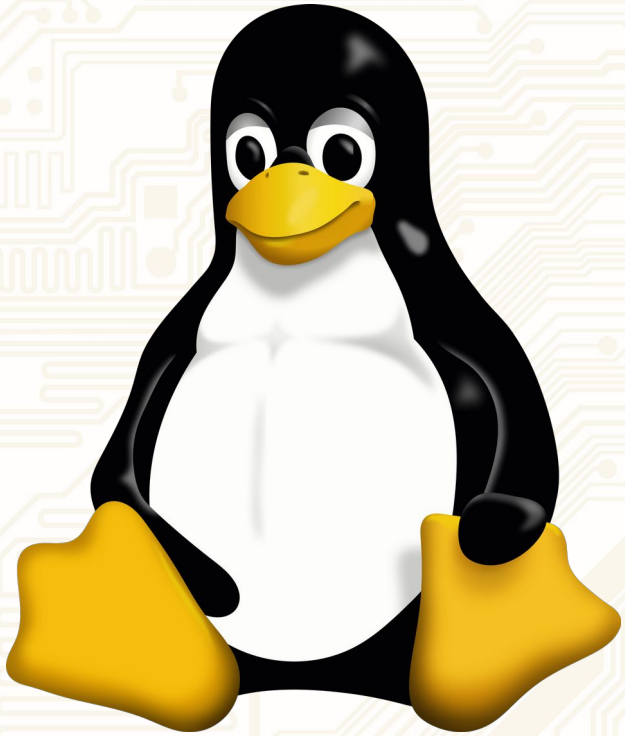
2 bloky, mód ECB

```
Hex:   0000 0000 0000 0000 0000 0000 0000 0000
       0000 0000 0000 0000 0000 0000 0000 0000
Klíč: SIFROVANI RULEZ.
```

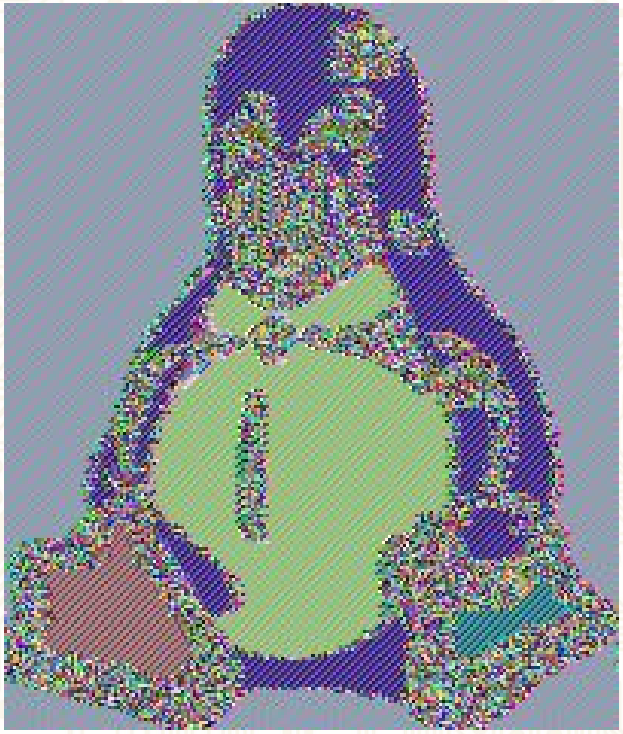
šifrování

```
Hex: 7e85 e871 5919 9215 34d2 5fb1 f7f2 a8a6
     7e85 e871 5919 9215 34d2 5fb1 f7f2 a8a6
```

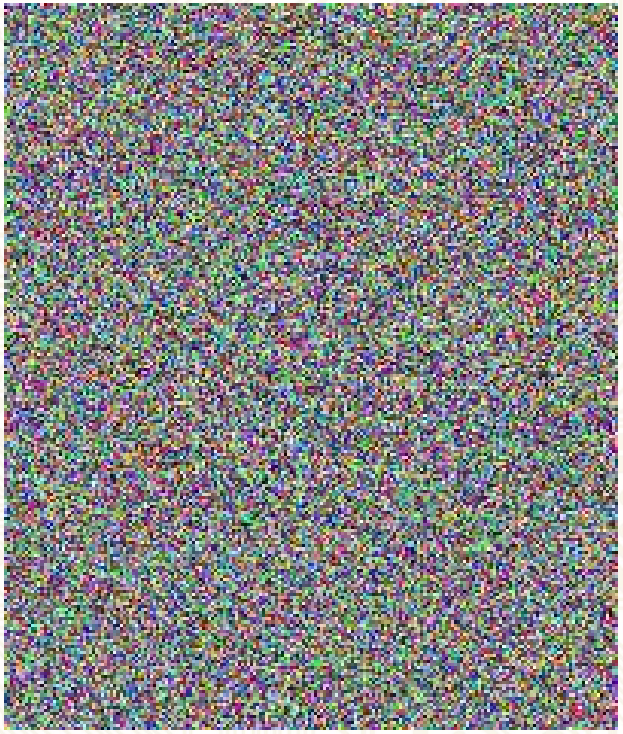
ECB: Vizualizace



Původní obrázek



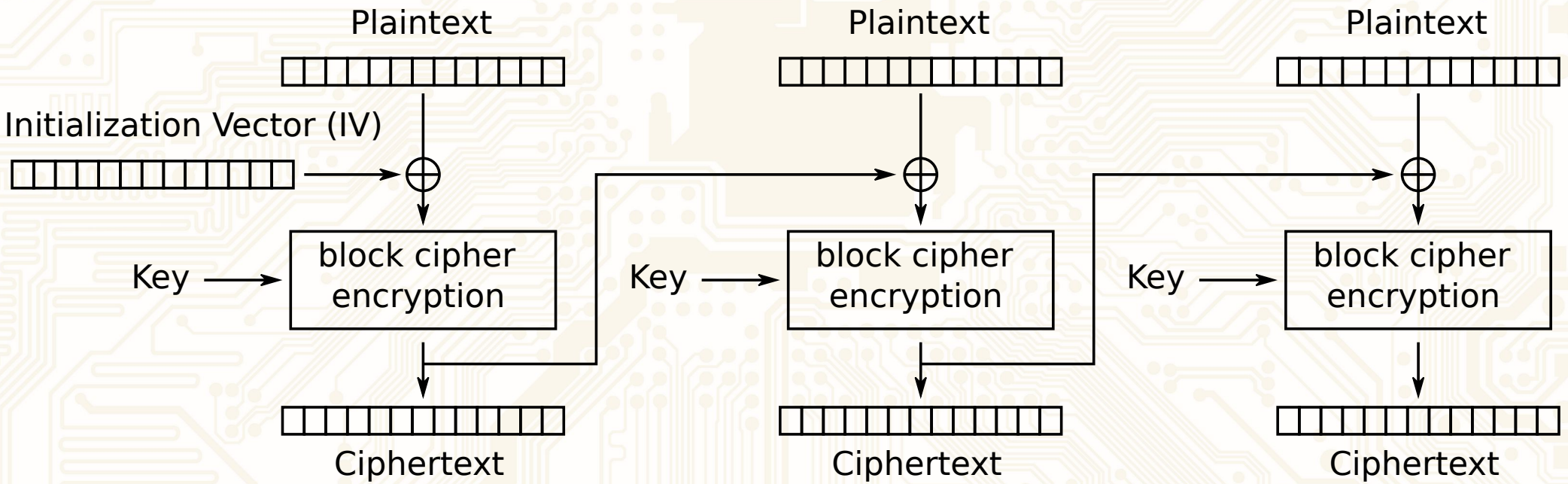
ECB



Cokoliv jen ne ECB

Tučňák Tux: Larry Ewing, Simon Budig, Garrett LeSage

CBC: Cipher Block Chaining



Cipher Block Chaining (CBC) mode encryption

2 bloky, mód CBC

```
Hex:   0000 0000 0000 0000 0000 0000 0000 0000
       0000 0000 0000 0000 0000 0000 0000 0000
Klíč: SIFROVANI RULEZ.
IV:    0000 0000 0000 0000 0000 0000 0000 0000
```

šifrování

```
Hex: 7e85 e871 5919 9215 34d2 5fb1 f7f2 a8a6
     787b 6784 6f28 9b0a 5ace 5015 8013 9482
```

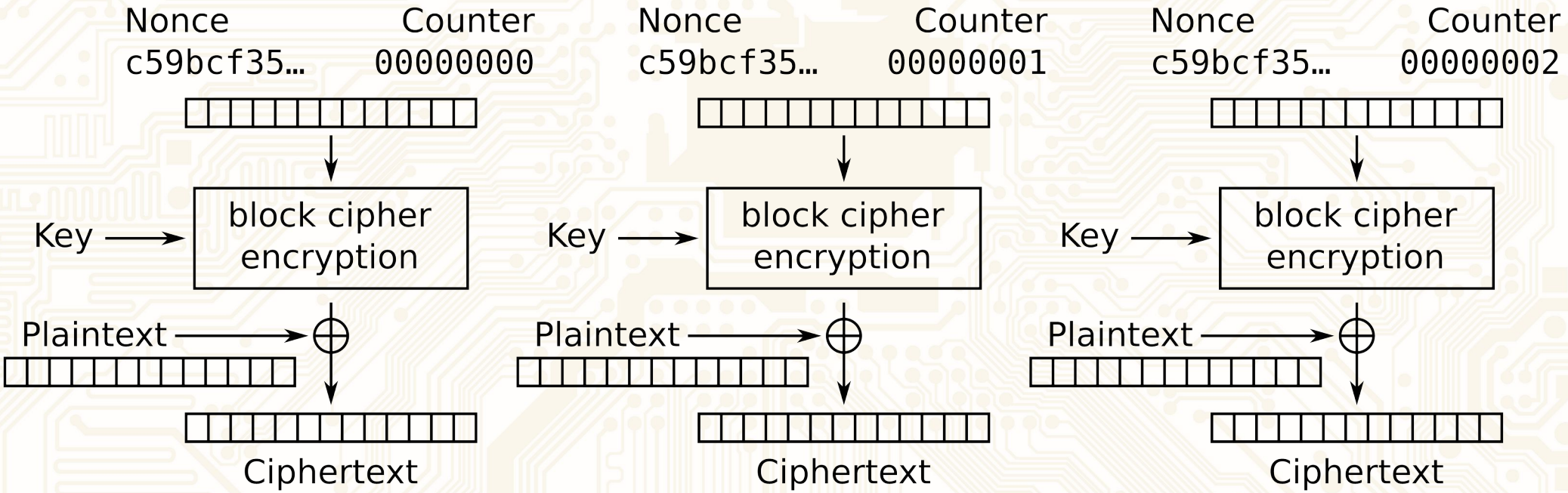
CBC: Cipher Block Chaining

- každý následující blok plaintextu je XORován s šifrovým textem předchozího bloku
- první blok plaintextu je XORován s tzv. inicializačním vektorem (náhodně vygenerovaná data o velikosti bloku)
- nevýhodou je závislost na předchozím bloku, nemožnost paralelizace pro zašifrování
- předvídatelný IV může být využit k vytvoření speciálního plaintextu, který po zašifrování zanechá detekovatelný vzor – problém diskového šifrování
- stále neřeší útoky zaměřené na změny zašifrovaných dat nebo podvržení zašifrovaných dat útočníkem

Counter (CTR) mode

- dělá z blokové šifry proudovou
- z „nonce" (ekvivalent IV) a pořadového čísla pomáhají spolu se šifrou tvořit keystream
- keystream se XORuje s plaintextem
- režim podporuje paralelní operaci (nemusí se nejprve šifrovat předchozí bloky)
- neřeší se integrita dat

Counter (CTR) mode



Counter (CTR) mode encryption

2 bloky, mód CTR

```
Hex:   0000 0000 0000 0000 0000 0000 0000 0000
       0000 0000 0000 0000 0000 0000 0000 0000
Klíč: SIFROVANI RULEZ.
IV:    0000 0000 0000 0000 0000 0000 0000 0000
```

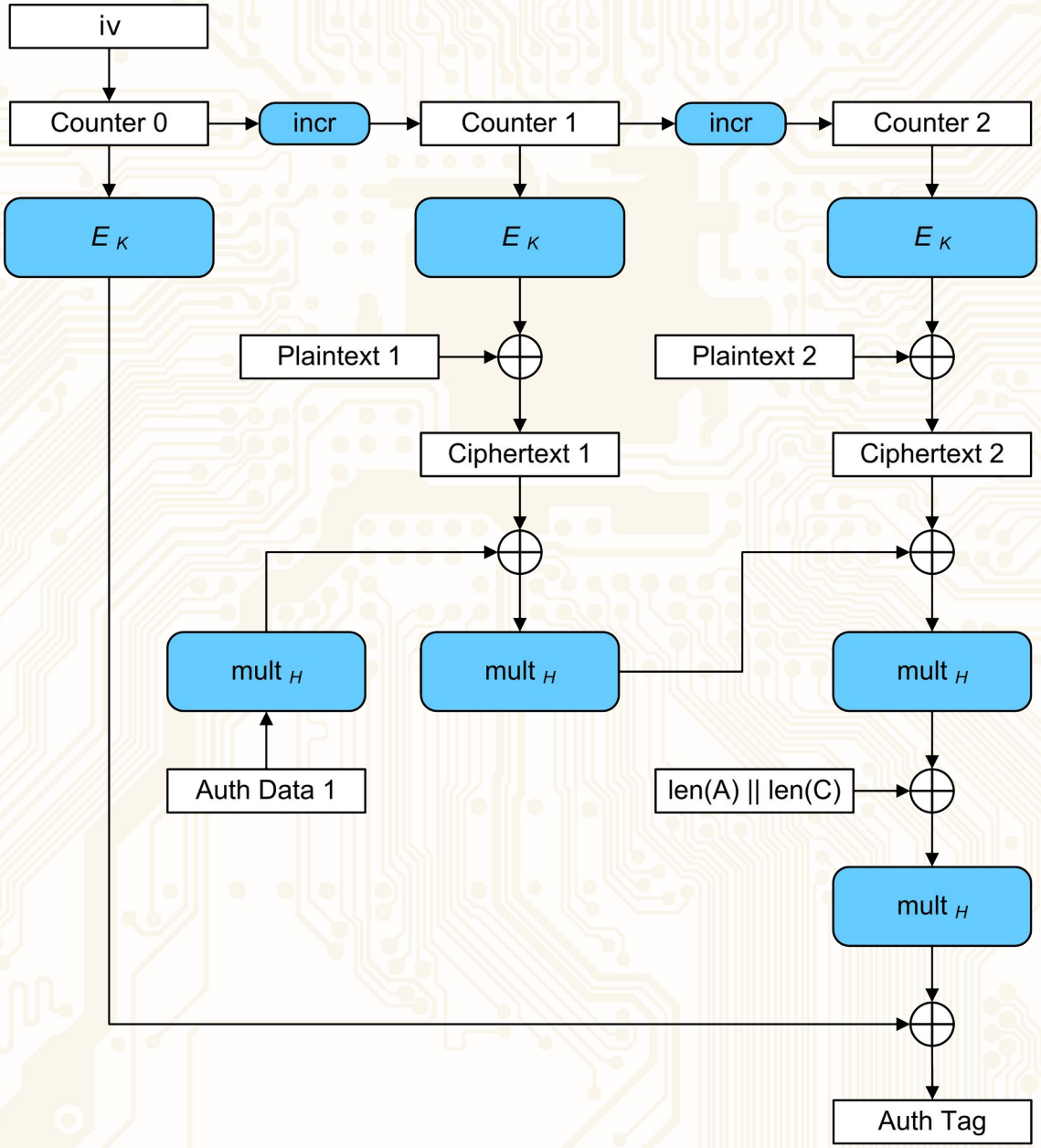
šifrování

```
Hex: 7e85 e871 5919 9215 34d2 5fb1 f7f2 a8a6
     6629 c126 4cd8 4081 f540 7f87 1e74 61e6
```

Galois/counter mode (GCM)

- varianta CTR provozního režimu s autentizací Galois
- plná podpora paralelního zpracování
- podporuje verifikaci integrity zašifrovaných dat i připojených nešifrovaných dat (AEAD: Authenticated encryption with associated data)
- používá se v TLS 1.2 i současném TLS 1.3 a na dalších místech (SSH, OpenVPN, atd.)

Galois/counter mode (GCM)

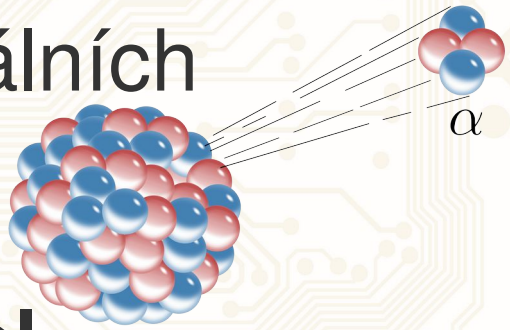


Náhodná čísla

- bezpečnost nejen v kryptografii stojí na nepředvídatelnosti náhodně vygenerovaných čísel (např. náhodně vygenerovaný klíč)
- skutečně náhodná čísla jsou založena na náhodných jevech (např. počet rozpadů radioaktivní látky)
- ačkoliv existují hardwarové generátory náhodných čísel, u počítačů převládají generátory pseudonáhodných čísel

Generátory náhodných čísel

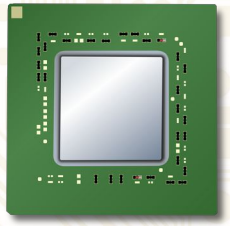
- **hardwarové generátory náhodných čísel** – založené na nepředvídatelných fyzikálních jevech (např. radioaktivní přeměna)



- **generátory pseudonáhodných čísel**
  - algoritmus je třeba nakrmit (SEED) nějakými (pokud možno náhodnými) daty
  - z těchto dat se pak generují náhodná čísla
  - stejné krmení = stejná vygenerovaná čísla
  - po určité době se čísla mohou opakovat

Generátory náhodných čísel

- **generátory pseudonáhodných čísel**
  - v OS bývají poměrně kvalitní generátory pseudonáhodných čísel (**`/dev/urandom`**)
  - jako krmení (**SEED**) se dají využít HW generátory i nahodilé jevy (doba mezi kliknutími myši či stisky kláves, doba vystavení hlavičky pevného disku, nejmoderněji je to **CPU execution timing jitter**)



  - **pozor na slabé generátory!**
    - při vývoji aplikací užívejte „cryptographically secure" generátory (CSPRNG)

Vernamova šifra (1917)

- šifrování i dešifrování provádí ***exkluzivní bitovou disjunkci*** (**XOR**) čistého textu s klíčem
- dokonalá a bezpečná šifra za předpokladu, že:
  1. Klíč je **stejně dlouhý** jako čistý text
  2. Klíč je skutečně **náhodný** (žádné pseudo)
  3. Klíč není **nikdy použit znovu**

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

```
Čistý text:  01011100        Šifrový text:  10110001
Klíč:        11101101        Klíč:          11101101
XOR výstup:  10110001        XOR výstup:    01011100
```

Důsledky a souvislosti

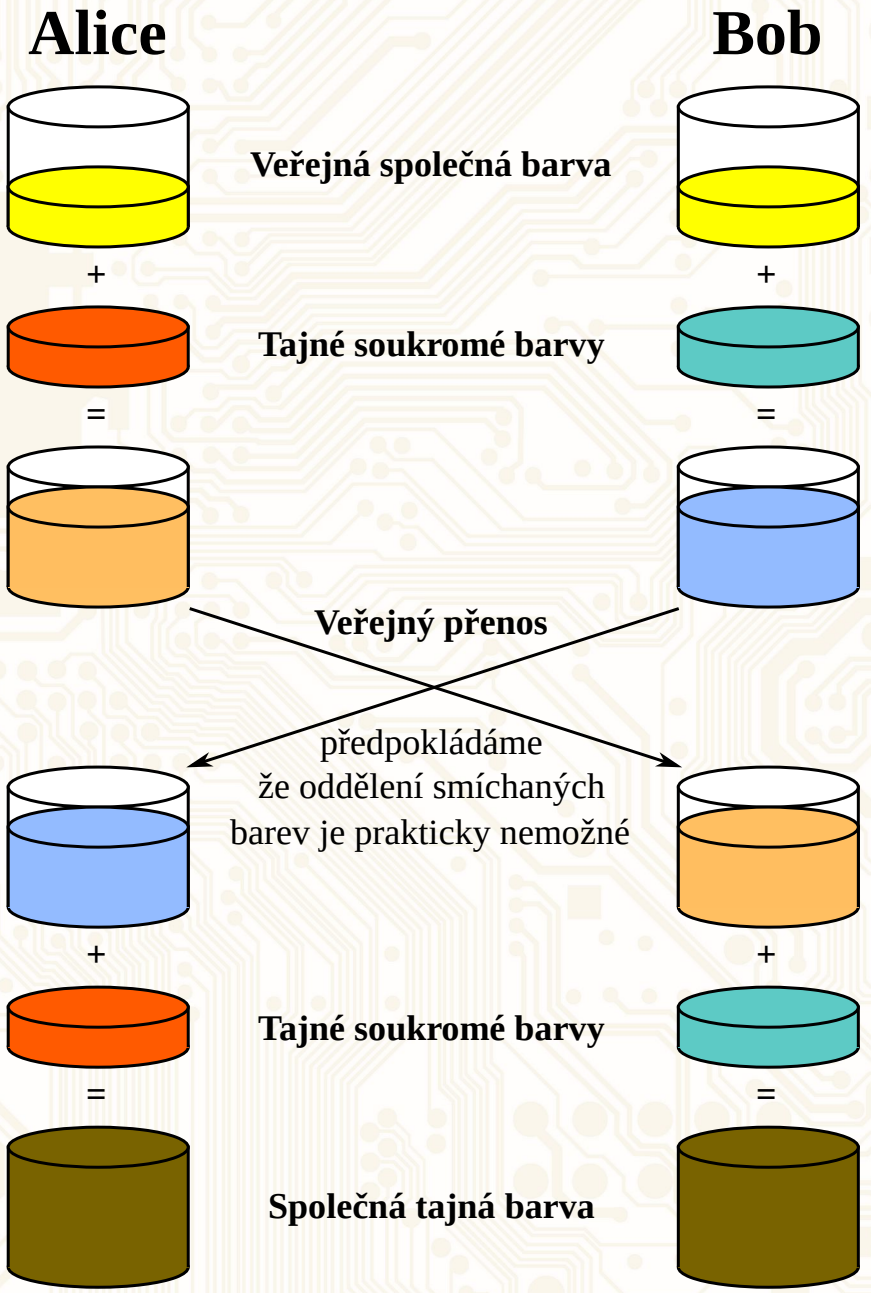
- symetrické šifry mají klíč kratší než data
- princip XORování dat s klíčovými daty je využíván v proudových šifrách a šifrovacích módech pro blokové šifry
- v obou případech však již nejsou splněny podmínky Vernamovy šifry, a oba postupy jsou tedy zranitelné

TLS: kde vzít klíče pro šifrování?

- **cíl**: chceme komunikaci zašifrovat
- **kde vzít klíč**?
  - vygenerujeme si je generátorem náhodných čísel
- **problém**: jak předáme protistraně tajný klíč, když zatím komunikujeme nešifrovaně?
- **řešení**: Diffie-Hellmanova výměna klíčů

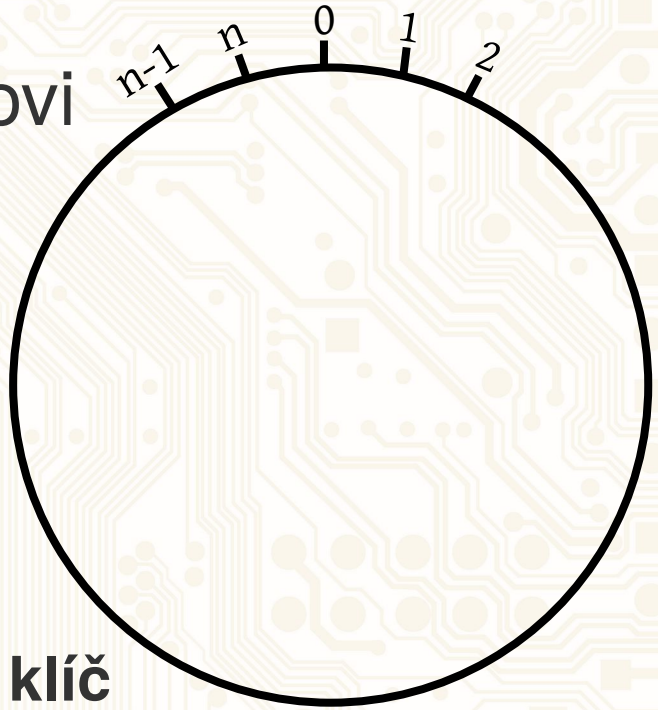
Diffie-Hellmanova výměna klíčů

- kryptografický protokol pro vytvoření klíče mezi komunikujícími stranami přes nezabezpečený kanál
- založený na **teorii grup** a **modulo** aritmetice
- modulo… zbytek po celočíselném dělení



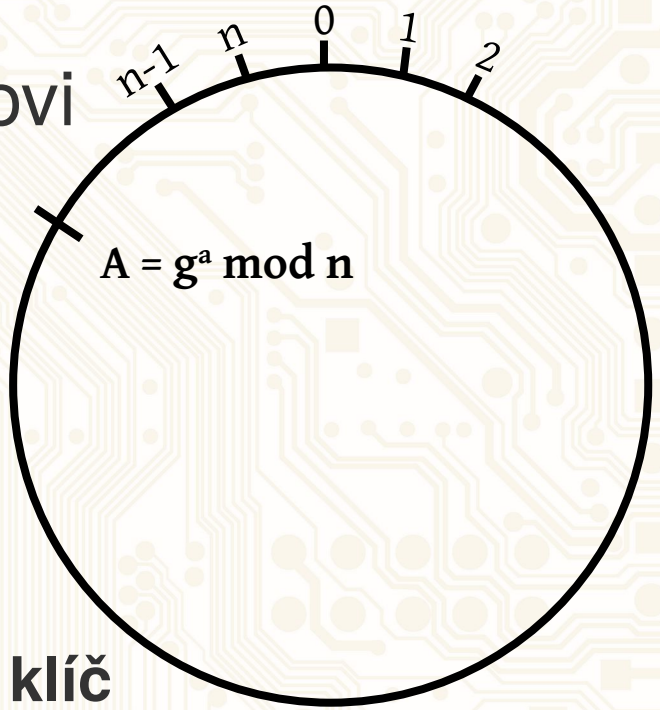
Diffie-Hellmanova výměna klíčů

- někdo z účastníků zveřejní malé prvočíslo ***g*** a velké ***n*** (*n* je dnes min. 2048 bitů – tj. 600 číslic a více)
- Alice si zvolí tajné přir. číslo ***a*** takové, že **1 < a < n**
- Bob si zvolí tajné přir. číslo ***b*** takové, že **1 < b < n**
- Alice spočte $\mathbf{A}=\mathbf{g}^{\mathbf{a}}\ \mathbf{mod\ n}$, předá Bobovi
- Bob spočítá $\mathbf{B}=\mathbf{g}^{\mathbf{b}}\ \mathbf{mod\ n}$, předá Alici
- Alice spočítá $(\mathbf{B})^{\mathbf{a}}\ \mathbf{mod\ n}$
- Bob spočítá $(\mathbf{A})^{\mathbf{b}}\ \mathbf{mod\ n}$
- $(\mathbf{B})^{\mathbf{a}}\ \mathbf{mod\ n} = (\mathbf{A})^{\mathbf{b}}\ \mathbf{mod\ n}$ = **dohodnutý klíč**



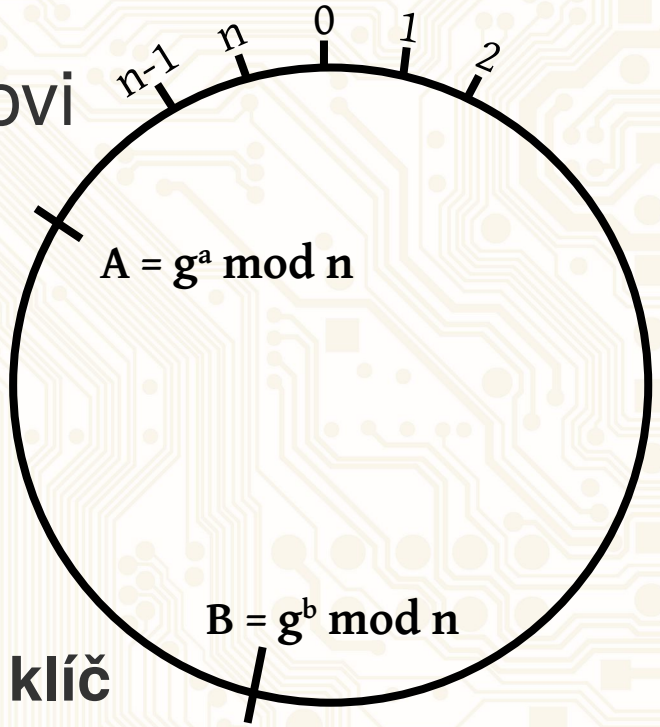
Diffie-Hellmanova výměna klíčů

- někdo z účastníků zveřejní malé prvočíslo ***g*** a velké ***n*** (*n* je dnes min. 2048 bitů – tj. 600 číslic a více)
- Alice si zvolí tajné přir. číslo ***a*** takové, že **1 < a < n**
- Bob si zvolí tajné přir. číslo ***b*** takové, že **1 < b < n**
- Alice spočte $\mathbf{A}=\mathbf{g}^{\mathbf{a}}\ \mathbf{mod\ n}$, předá Bobovi
- Bob spočítá $\mathbf{B}=\mathbf{g}^{\mathbf{b}}\ \mathbf{mod\ n}$, předá Alici
- Alice spočítá $(\mathbf{B})^{\mathbf{a}}\ \mathbf{mod\ n}$
- Bob spočítá $(\mathbf{A})^{\mathbf{b}}\ \mathbf{mod\ n}$
- $(\mathbf{B})^{\mathbf{a}}\ \mathbf{mod\ n} = (\mathbf{A})^{\mathbf{b}}\ \mathbf{mod\ n}$ = **dohodnutý klíč**



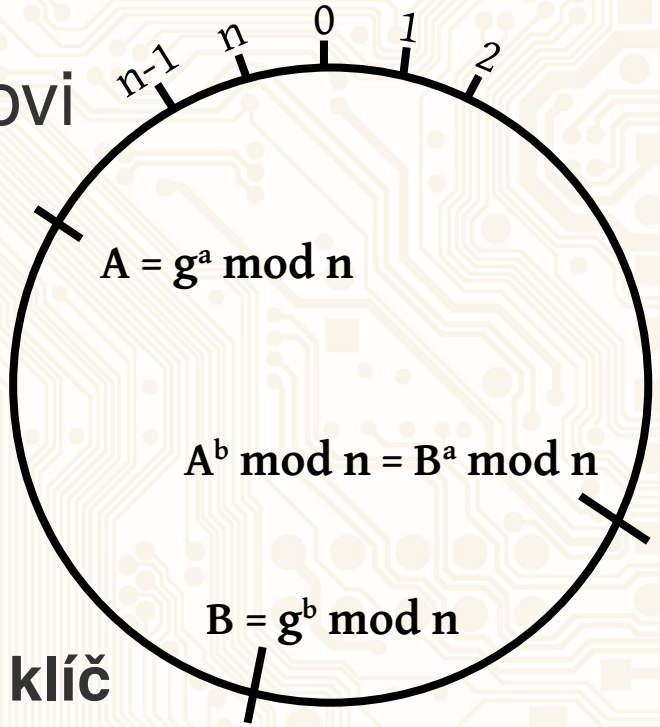
Diffie-Hellmanova výměna klíčů

- někdo z účastníků zveřejní malé prvočíslo ***g*** a velké ***n*** (*n* je dnes min. 2048 bitů – tj. 600 číslic a více)
- Alice si zvolí tajné přir. číslo ***a*** takové, že **1 < a < n**
- Bob si zvolí tajné přir. číslo ***b*** takové, že **1 < b < n**
- Alice spočte $\mathbf{A}=\mathbf{g}^{\mathbf{a}}$ **mod n**, předá Bobovi
- Bob spočítá $\mathbf{B}=\mathbf{g}^{\mathbf{b}}$ **mod n**, předá Alici
- Alice spočítá $(\mathbf{B})^{\mathbf{a}}$ **mod n**
- Bob spočítá $(\mathbf{A})^{\mathbf{b}}$ **mod n**
- $(\mathbf{B})^{\mathbf{a}}$ **mod n** = $(\mathbf{A})^{\mathbf{b}}$ **mod n** = **dohodnutý klíč**



Diffie-Hellmanova výměna klíčů

- někdo z účastníků zveřejní malé prvočíslo ***g*** a velké ***n*** (*n* je dnes min. 2048 bitů – tj. 600 číslic a více)
- Alice si zvolí tajné přir. číslo ***a*** takové, že **1 < a < n**
- Bob si zvolí tajné přir. číslo ***b*** takové, že **1 < b < n**
- Alice spočte $\mathbf{A}=\mathbf{g}^{\mathbf{a}}\ \mathbf{mod\ n}$, předá Bobovi
- Bob spočítá $\mathbf{B}=\mathbf{g}^{\mathbf{b}}\ \mathbf{mod\ n}$, předá Alici
- Alice spočítá $(\mathbf{B})^{\mathbf{a}}\ \mathbf{mod\ n}$
- Bob spočítá $(\mathbf{A})^{\mathbf{b}}\ \mathbf{mod\ n}$
- $(\mathbf{B})^{\mathbf{a}}\ \mathbf{mod\ n} = (\mathbf{A})^{\mathbf{b}}\ \mathbf{mod\ n}$ = **dohodnutý klíč**



Diffie-Hellmanova výměna klíčů

- veřejná data: **g = 7**, **n = 23**
- Alice si zvolí tajný klíč **a** = **4**
  - spočte **A** = $\mathbf{g^a}$ **mod n** = $\mathbf{7^4}$ **mod 23** = **9**, pošle Bobovi
- Bob si zvolí tajný klíč **b** = **3**
  - spočte **B** = $\mathbf{g^b}$ **mod n** = $\mathbf{7^3}$ **mod 23** = **21**, pošle Alici
- Alice: $\mathbf{B^a}$ **mod n** = 21^4 mod 23 = **16**
- Bob: $\mathbf{A^b}$ **mod n** = 9^3 mod 23 = **16**

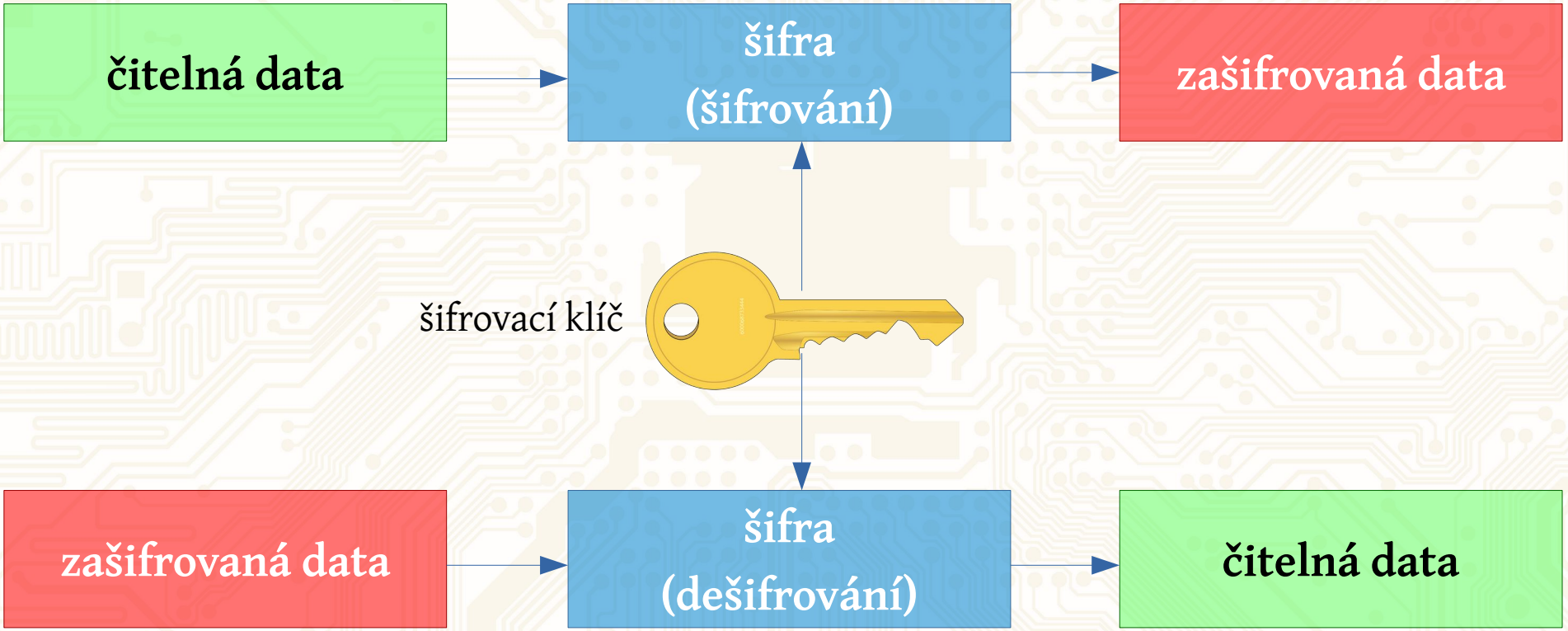
Diffie-Hellmanova výměna klíčů

- založena na:

$$(g^{a} \bmod n)^{b} \bmod n = (g^{b} \bmod n)^{a} \bmod n$$

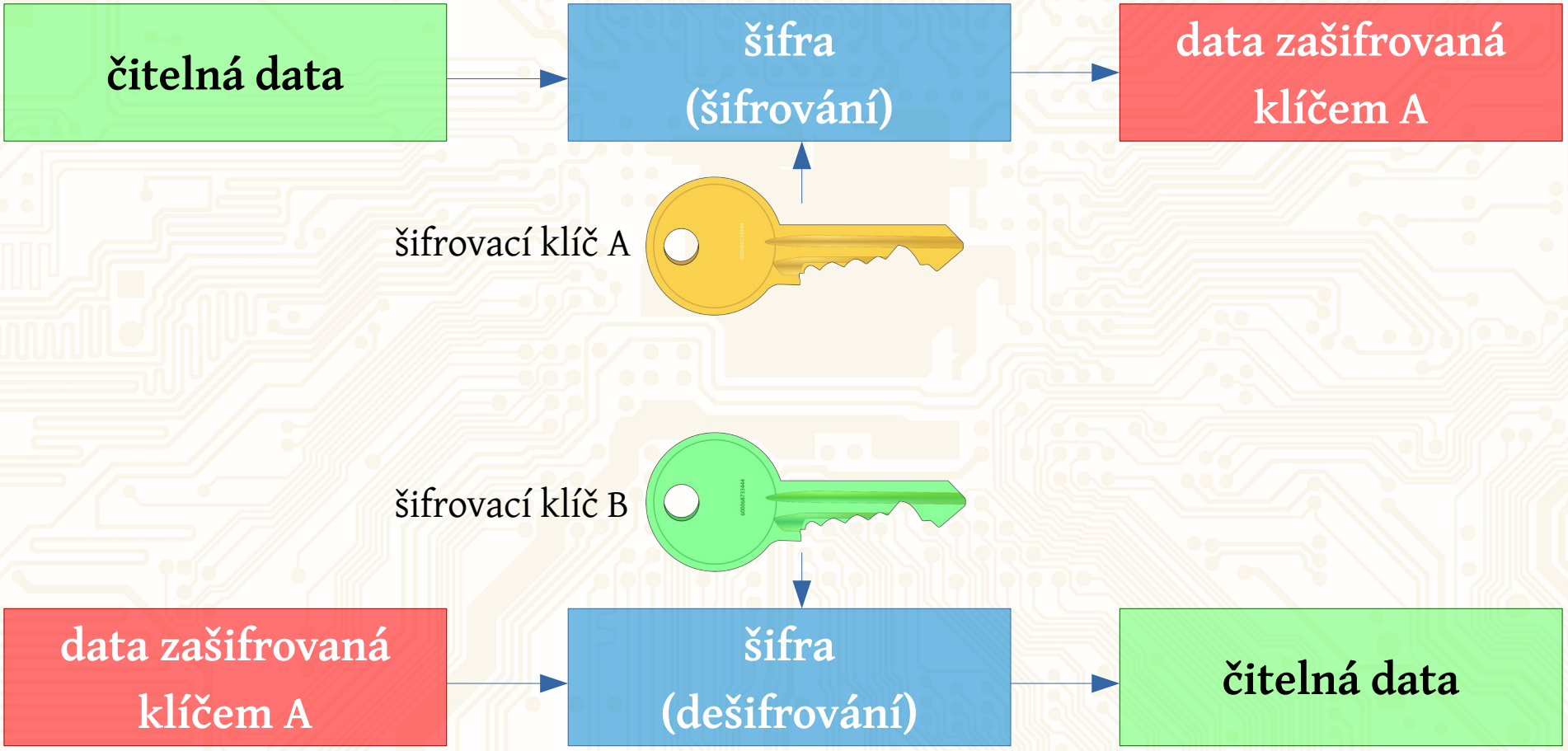
- z čísel posílaných veřejně nelze spočítat finální klíč (problém diskrétního logaritmu)
- existuje i varianta Diffie-Hellmana založená na eliptických křivkách: ECDH
- **slabina**: zranitelné vůči MITM útokům – jak víme, že nám A či B poslal skutečně Bob nebo Alice?!
- **řešení**: ověříme identitu protistrany (autentizace)

Symetrické šifrování



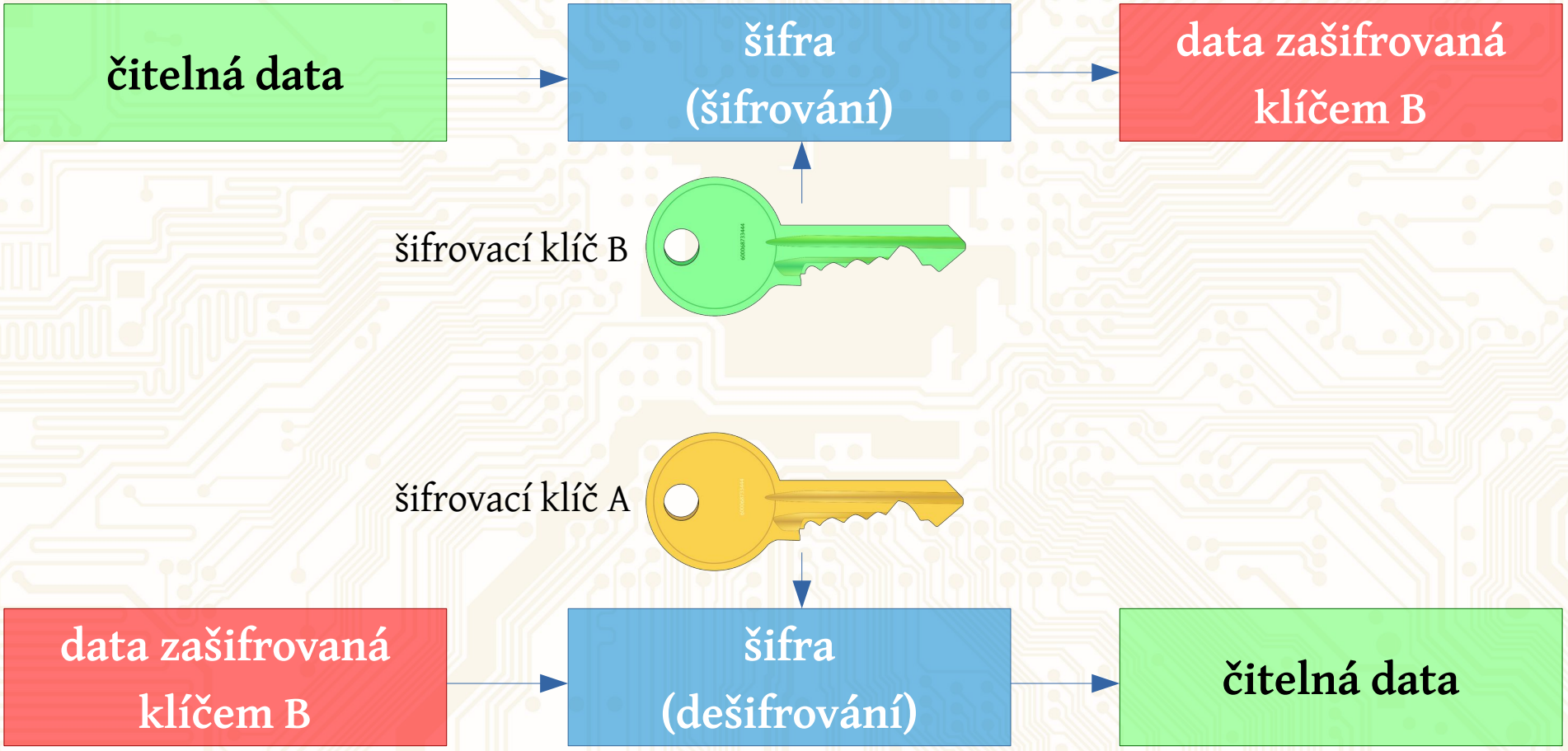
- pro šifrování a dešifrování je použit jeden a ten samý klíč

Asymetrické šifrování



- mám **pár klíčů**: co zašifruji jedním dešifruji jen druhým a naopak

Asymetrické šifrování



- mám **pár klíčů**: co zašifruji jedním dešifruji jen druhým a naopak

Asymetrické šifrování

- mám **pár klíčů**, které jsou matematicky sestavené tak, aby to, co zašifruji jedním, šlo dešifrovat jen druhým a naopak
- to, co zašifruji jedním klíčem nemohu dešifrovat tím samým klíčem
- příklady:
  - **Diffie-Hellman** + jeho eliptická křivková varianta
  - **RSA**, **DSA**, **ElGamal**
  - Eliptické křivky: **ECDSA**, **ECDH**
  - atd.

Soukromý a veřejný klíč

- v každém páru klíčů jsou klíče funkčně ekvivalentní, ale z hlediska použití se dělí na
  - **soukromý klíč**, který střežím jako oko v hlavě
  - **veřejný klíč**, který sdílím se světem
- někdo mi chce poslat zašifrovaná data – získá můj veřejný klíč, zašifruje jím data a pošle mi je
- data může dešifrovat jen vlastník mého soukromého klíče, tedy já

RSA (1977)

1. zvolíte si dvě různá velká náhodná prvočísla ***p*** a ***q***
2. spočítáte jejich součin ***n***=*pq*
3. spočítáte hodnotu Eulerovy funkce $\varphi(n)=(p-1)(q-1)$
4. zvolíte celé číslo ***e*** < $\varphi(n)$ a nesoudělné s $\varphi(n)$
5. naleznete číslo ***d*** tak, aby $de \equiv 1 \pmod{\varphi(n)}$
   - → je-li *d* prvočíslo, pak $d=(1+\boldsymbol{r}\times\varphi(n))/e$, kde:
   - → $\boldsymbol{r}=(e-1)\varphi(n)^{(e-2)}$

RSA (1977)

- **veřejným klíčem** je (*n*,*e*), kde ***n*** je modul a ***e*** veřejný exponent
- **soukromým klíčem** je (*n*,*d*), kde ***d*** je soukromý exponent

- **šifrování zprávy**: $c \equiv m^e \bmod n$, kde *m* je zpráva převedená na číslo, a toto číslo musí být menší *n*
- **dešifrování zprávy**: $m \equiv c^d \bmod n$

RSA: příklad

- $p = 79$, $q = 97$ (náhodná prvočísla, soukromá)
- $n = pq = 7663$ (modul – veřejný)
- $e = 5$ (veřejný exponent)
- $d = 4493$ (soukromý exponent)

- zašifrování zprávy m=42:
  - $m^e \bmod n = 42^5 \bmod 7663 = 6430$
- dešifrování:
  - $c^d \bmod n = 6430^{4493} \bmod 7663 = 42$

Zdroj: Kamilla Amirová

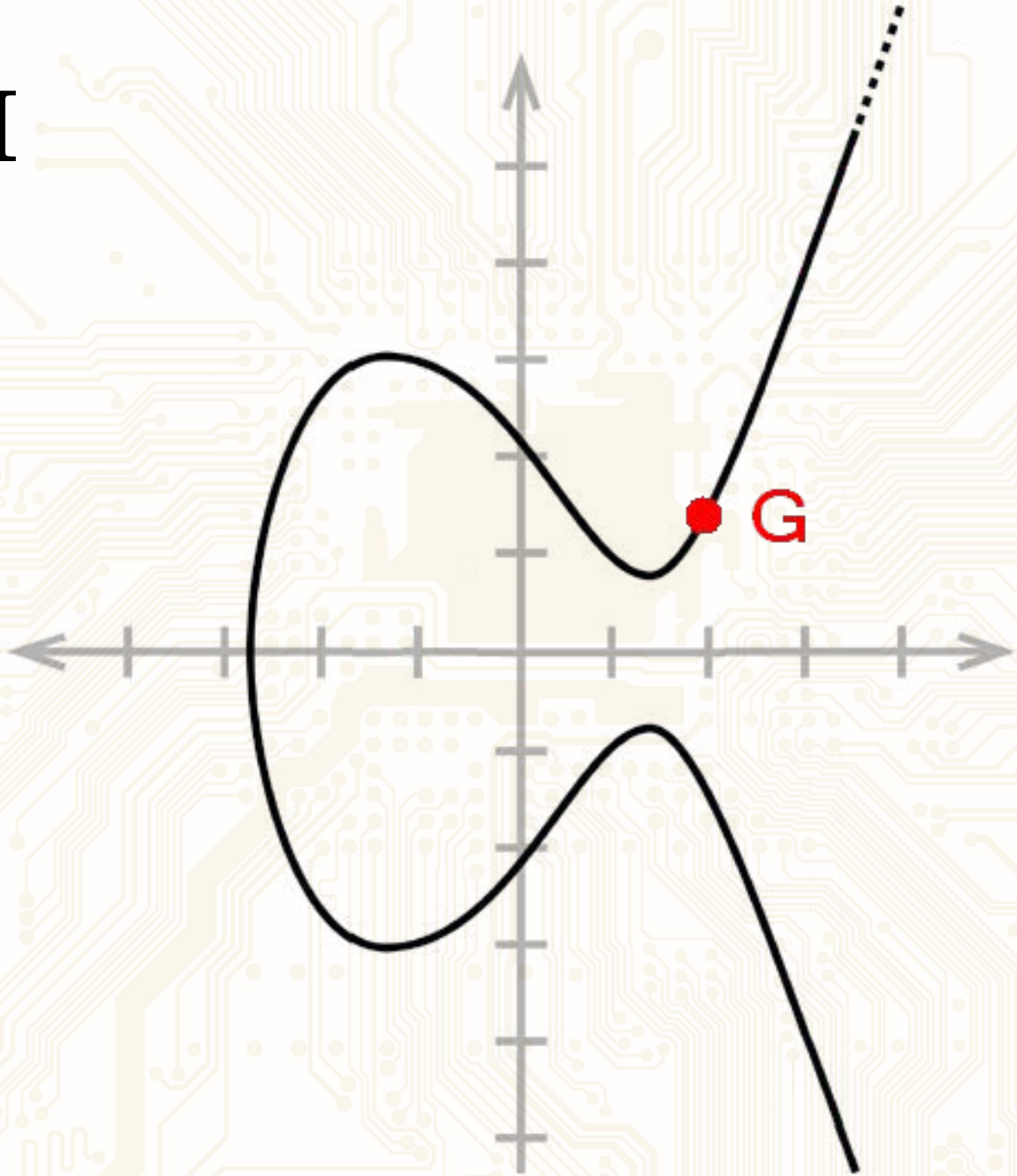
RSA: co si odnést

- co zašifrujeme jedním klíčem dešifrujeme druhým a naopak
  - zašifrování: $m^d \bmod n = \mathbf{42}^{4493} \bmod 7663 = 5175$
  - dešifrování: $c^e \bmod n = 5175^5 \bmod 7663 = \mathbf{42}$
- pokud něco zašifrujeme něčím klíčem, nemáme mechanismus, jak se dobrat původních dat (problém rozkladu součinu prvočísel na součinitele)

Eliptické křivky

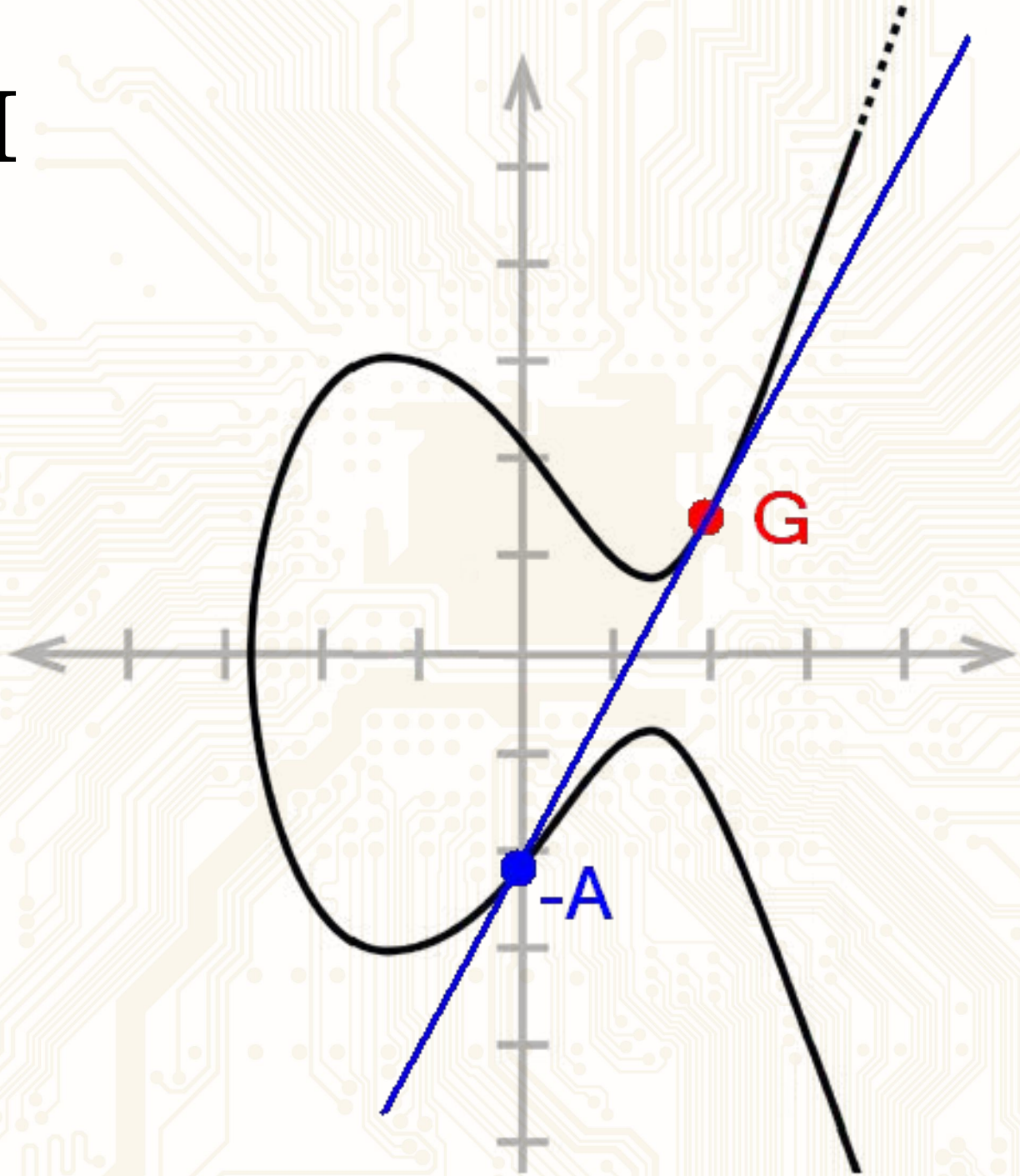
- **RSA**, **DSA**, **DH** algoritmy vyžadují dlouhé klíče 2048 – 4096 bitů, aby byly bezpečné
- EC algoritmy umí stejnou bezpečnost zajistit s kratšími klíči (nižší stovky bitů)
  - efektivnější přenos (zvažte mobilní data!)
  - rychlejší operace (zvažte mobilní platformy)
- ECDH, ECDSA
  - popis ECDH (video)

ECDH



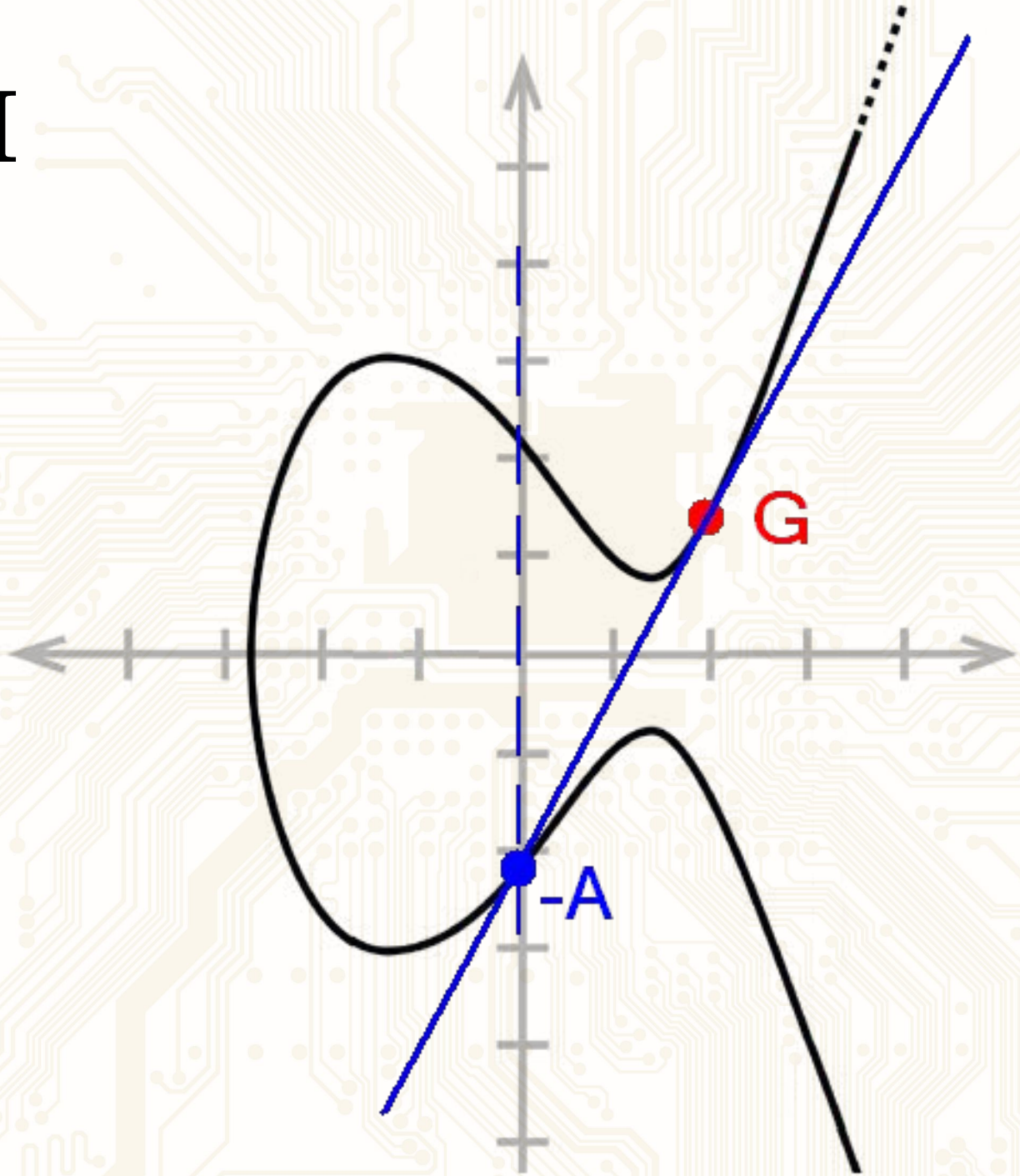
Byla zvolena eliptická křivka, bod G.

ECDH



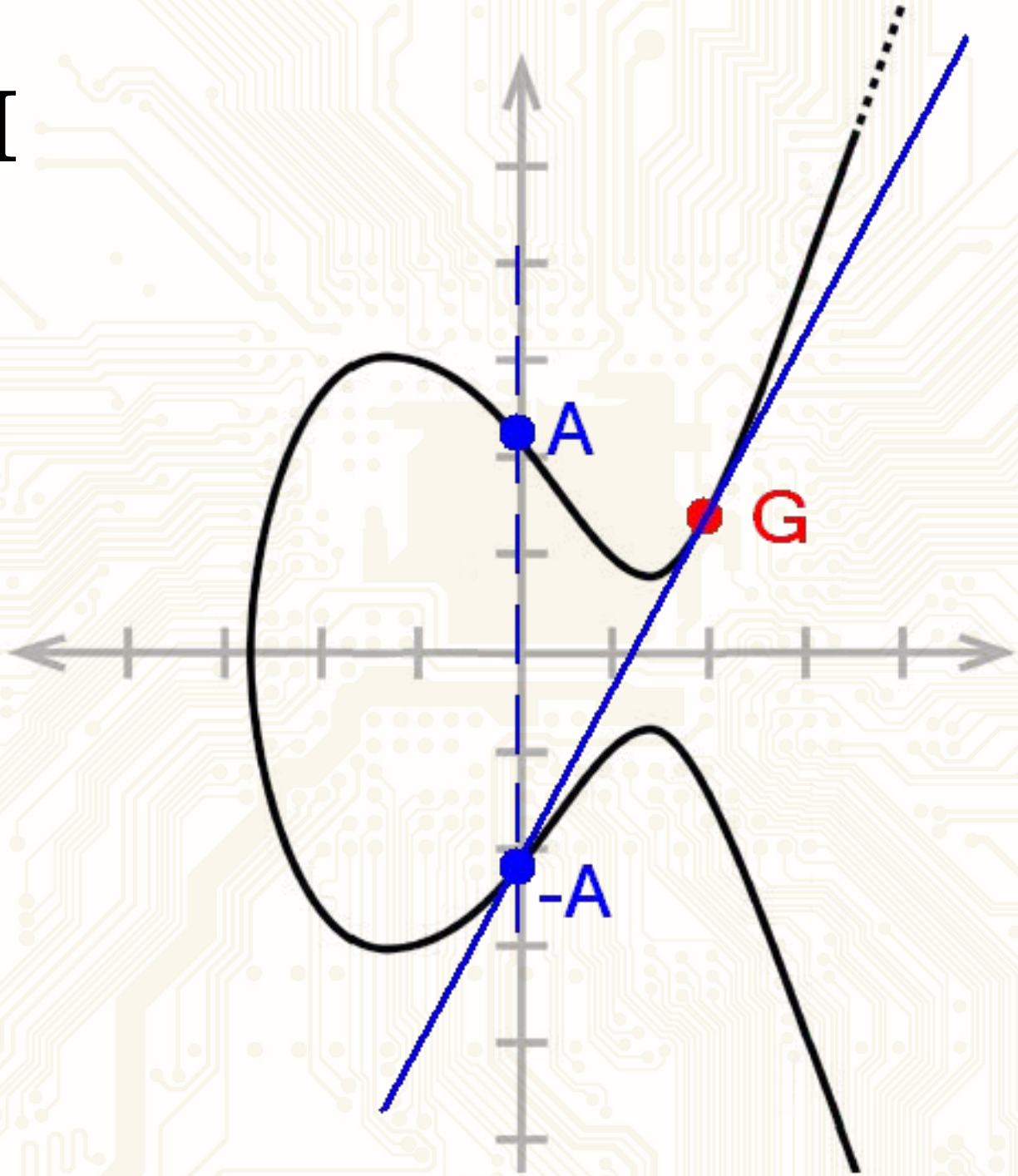
Alice tajně zvolila a=2,
zdvojnásobuje bod G.

ECDH



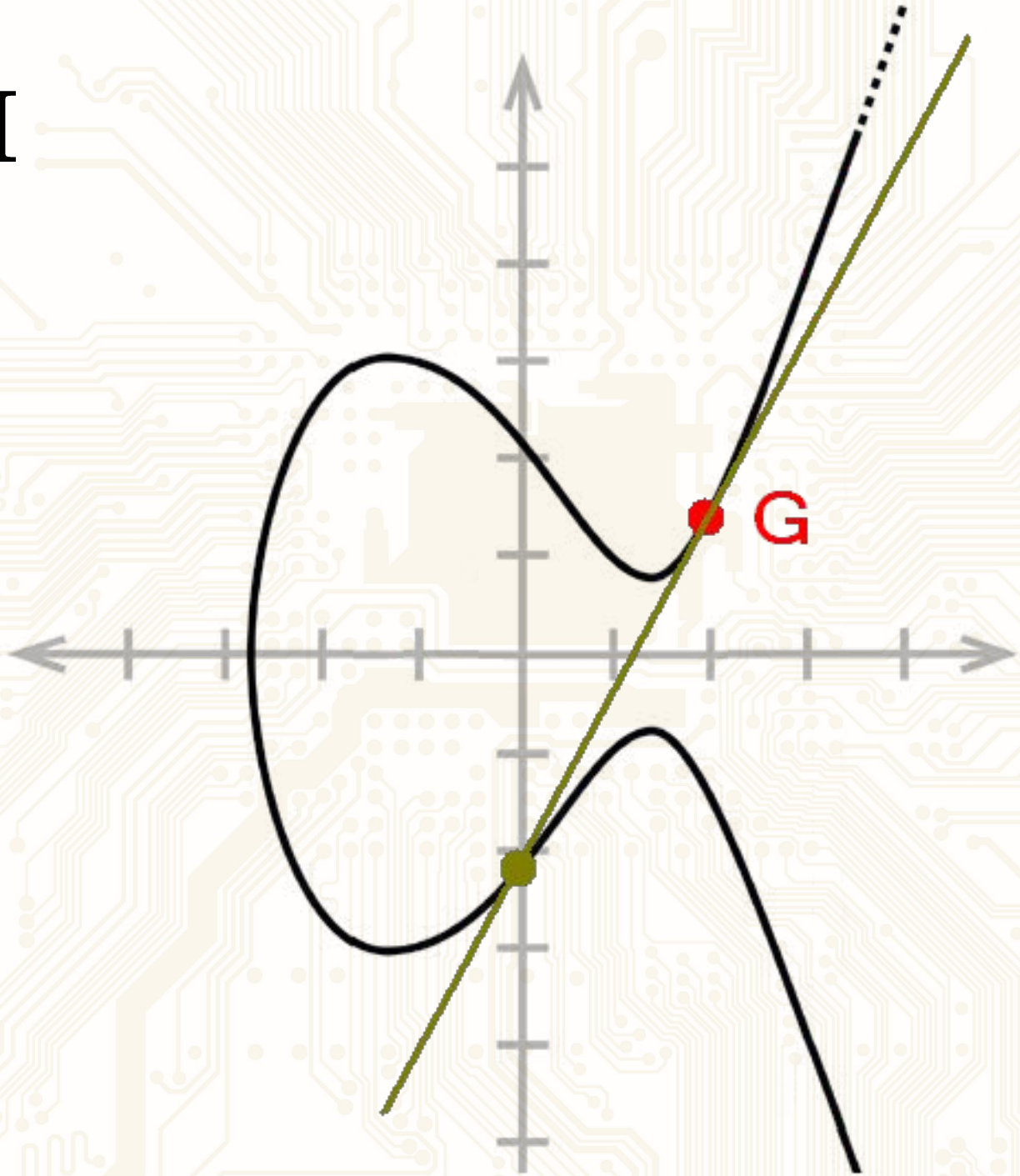
Alice tajně zvolila a=2,
zdvojnásobuje bod G.

ECDH



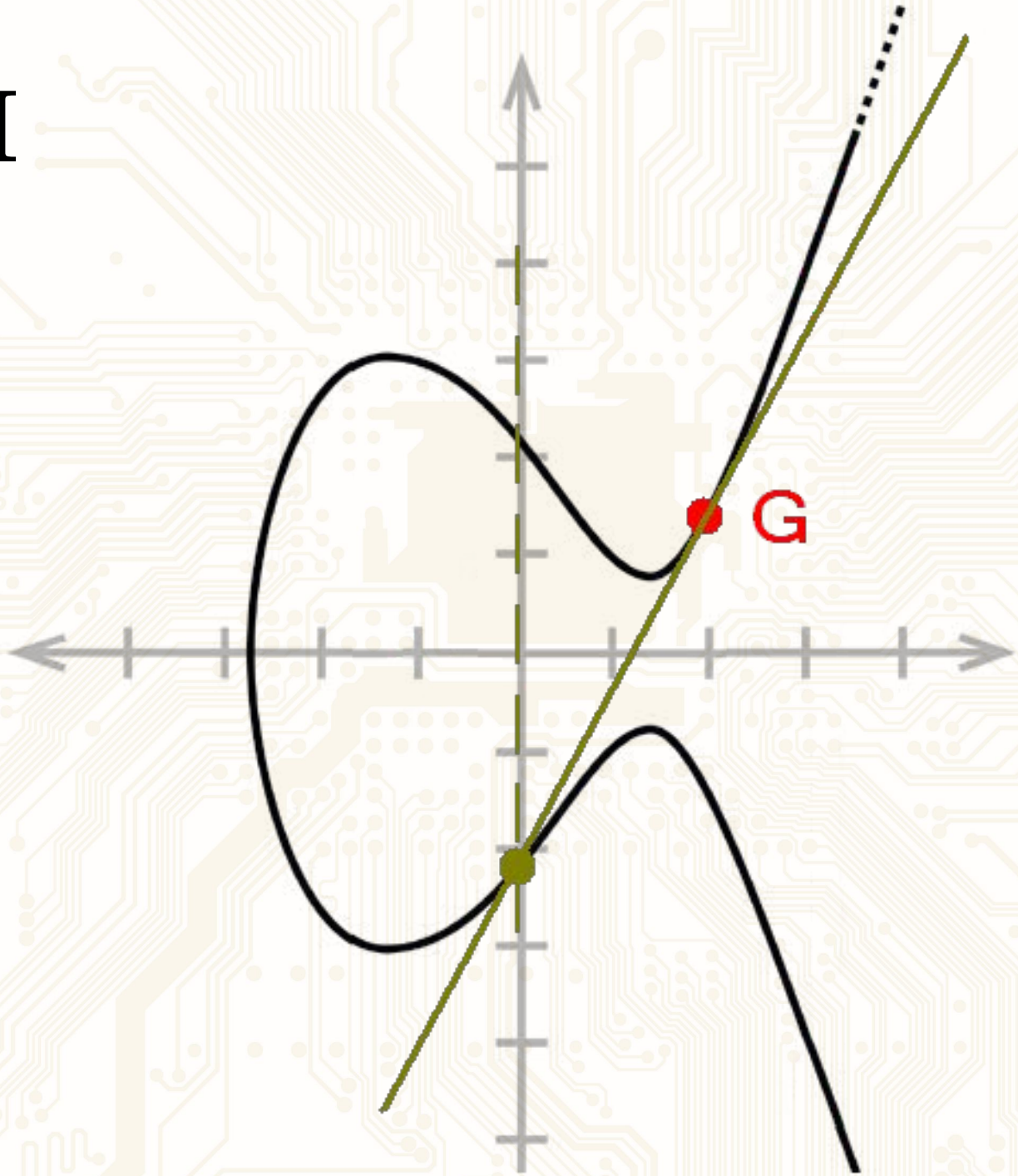
Alice nalezla bod A.

ECDH



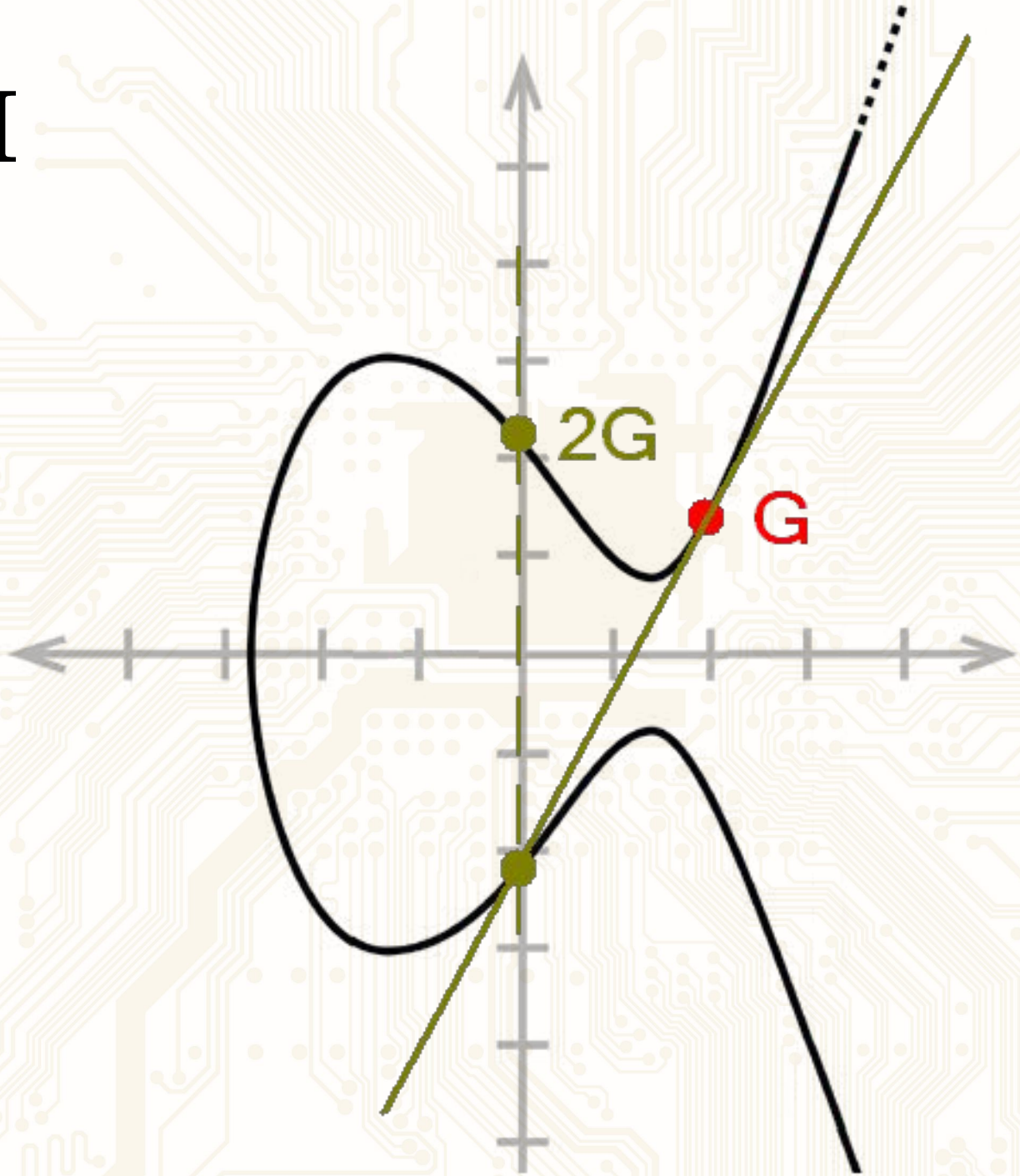
Bob tajně zvolil b=3,
ztrojnásobuje bod G.

ECDH



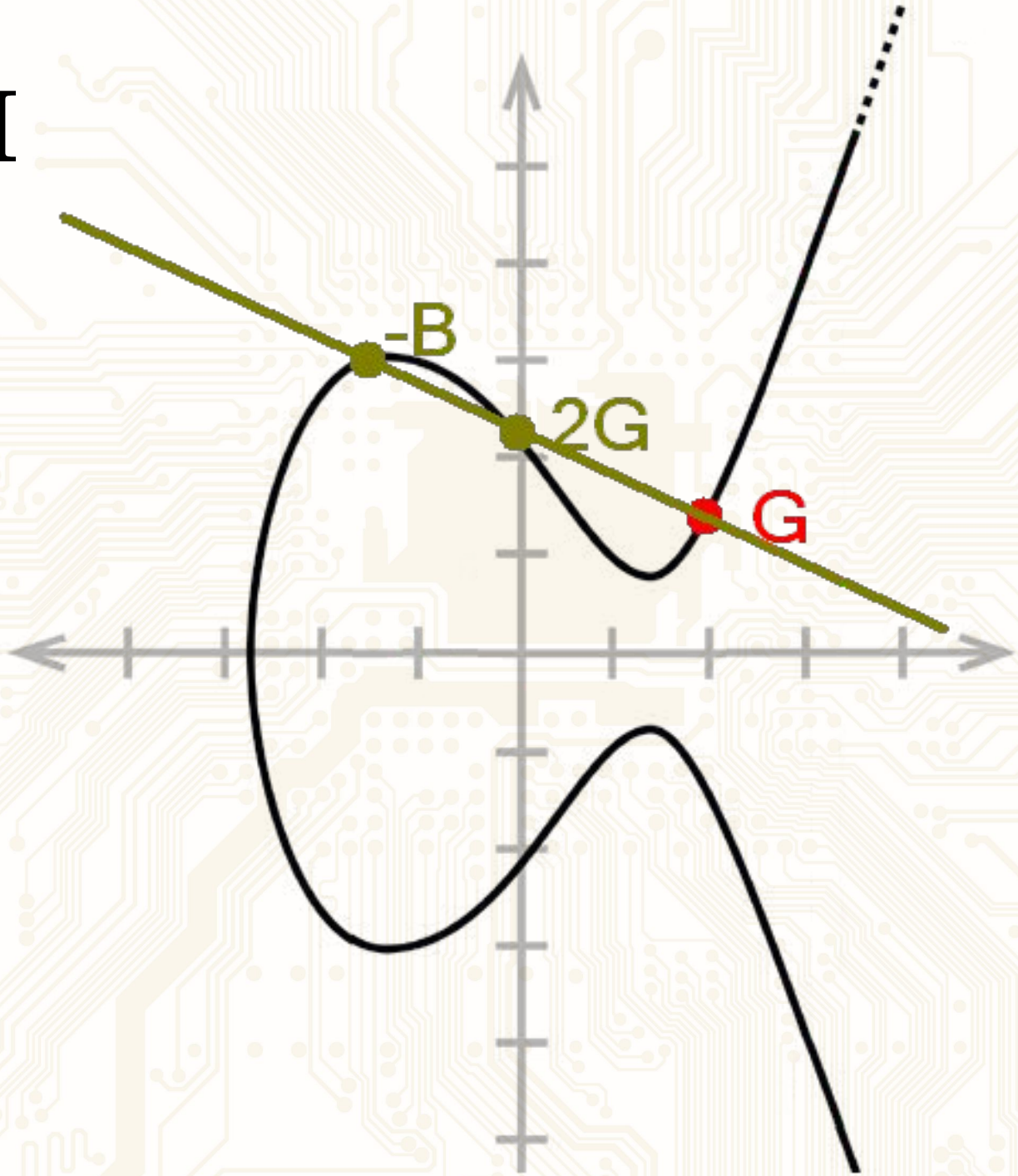
Bob tajně zvolil b=3,
ztrojnásobuje bod G.

ECDH



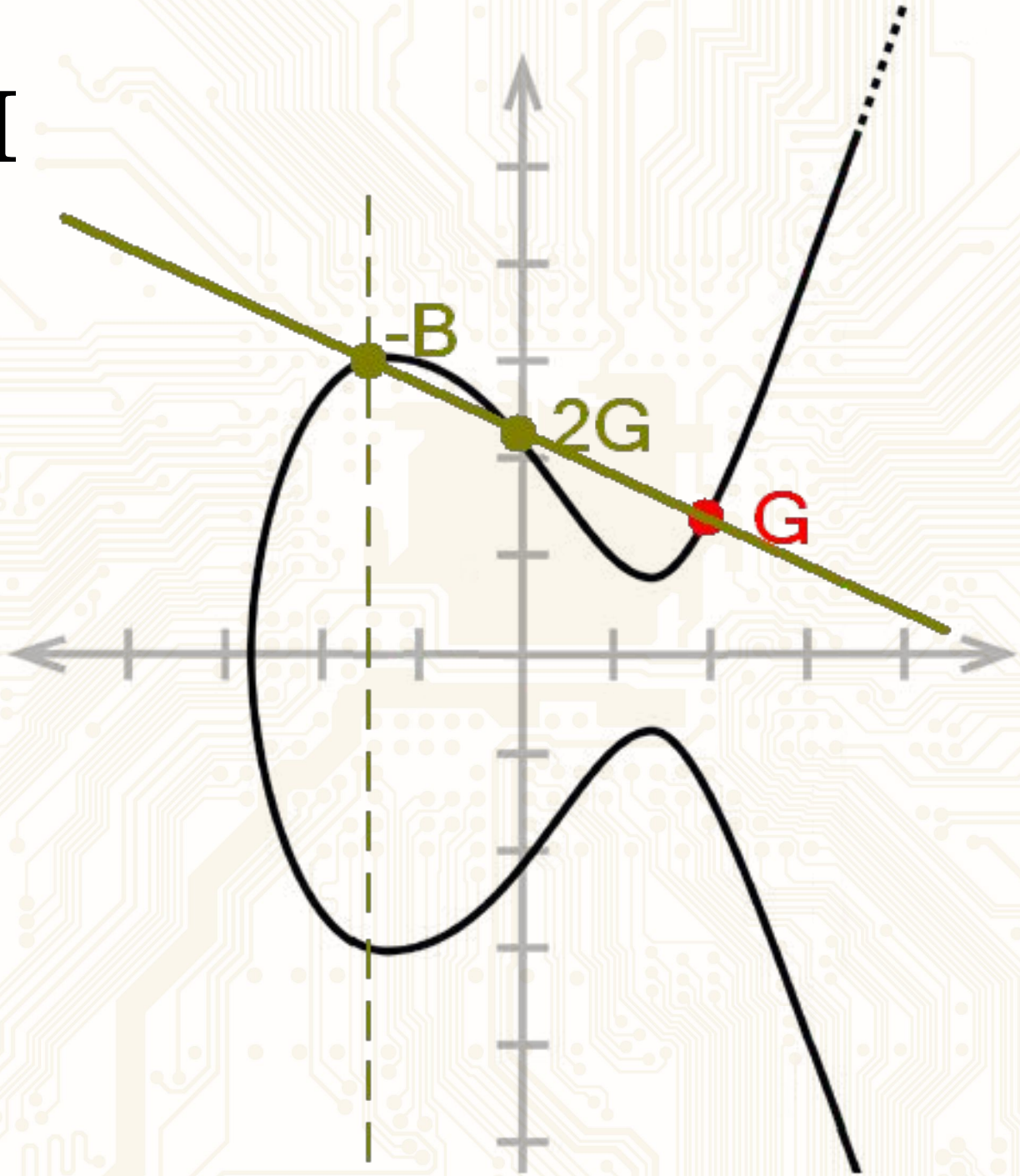
Bob tajně zvolil b=3,
ztrojnásobuje bod G.

ECDH



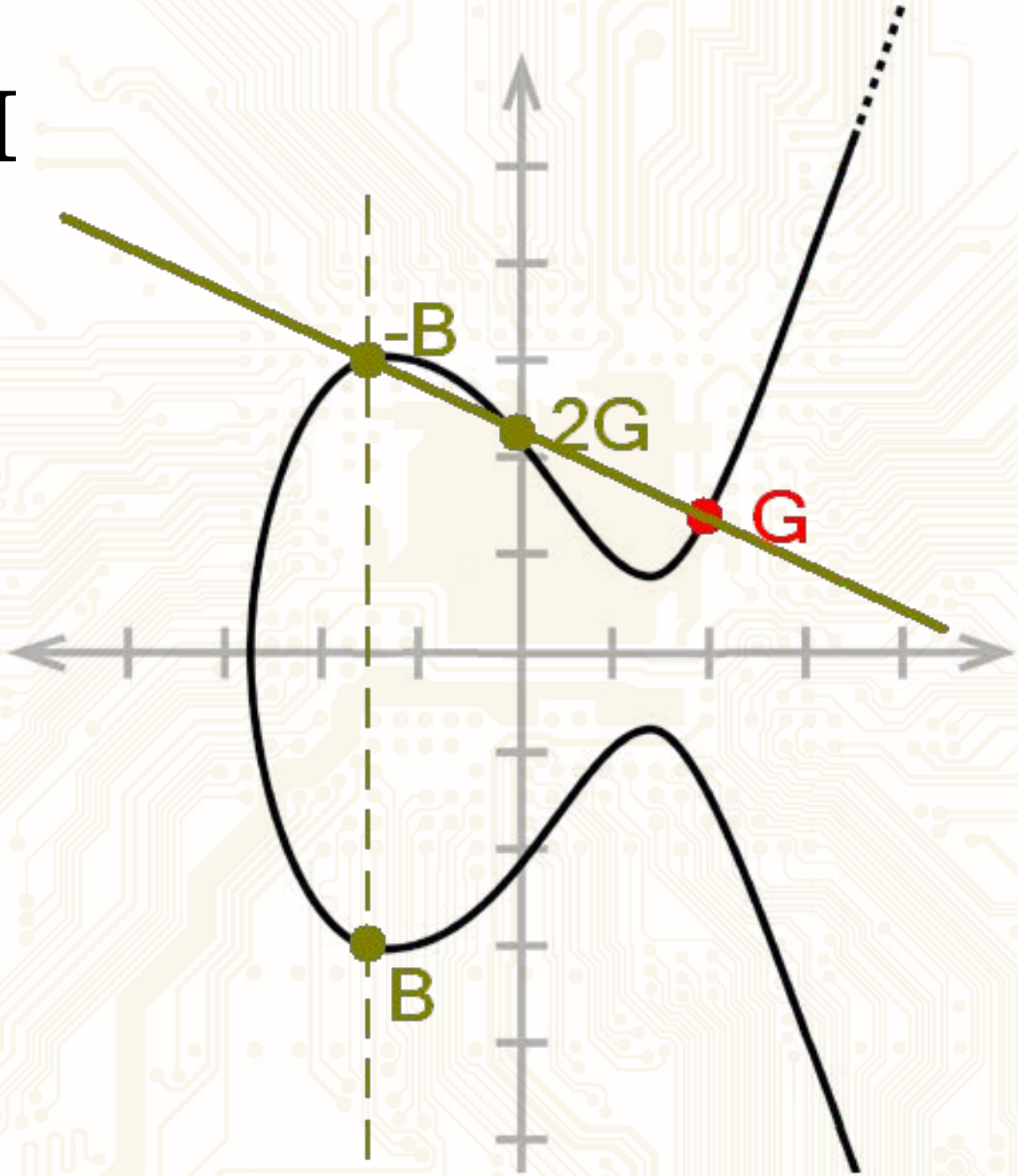
Bob tajně zvolil b=3,
ztrojnásobuje bod G.

ECDH



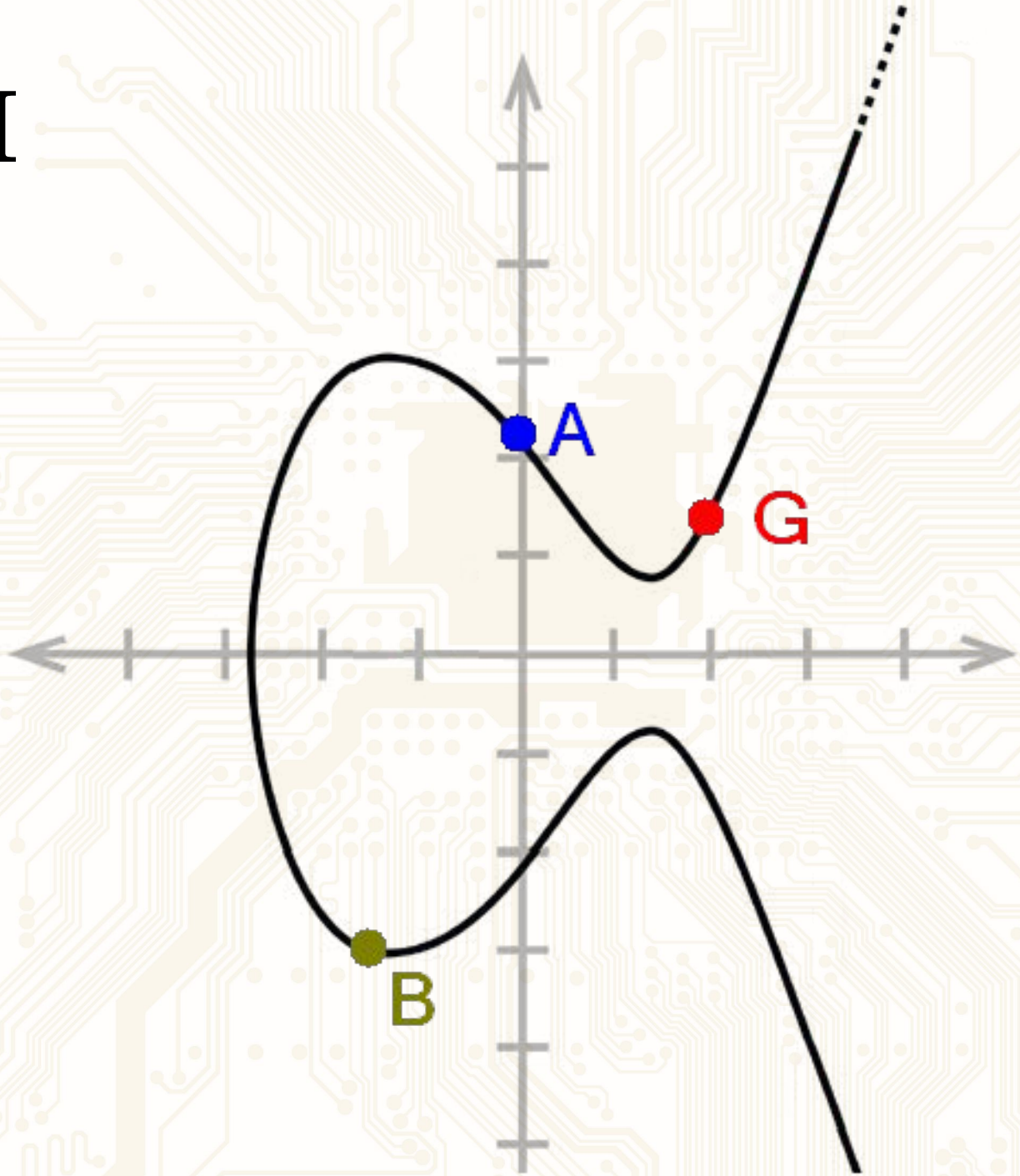
Bob tajně zvolil b=3,
ztrojnásobuje bod G.

ECDH



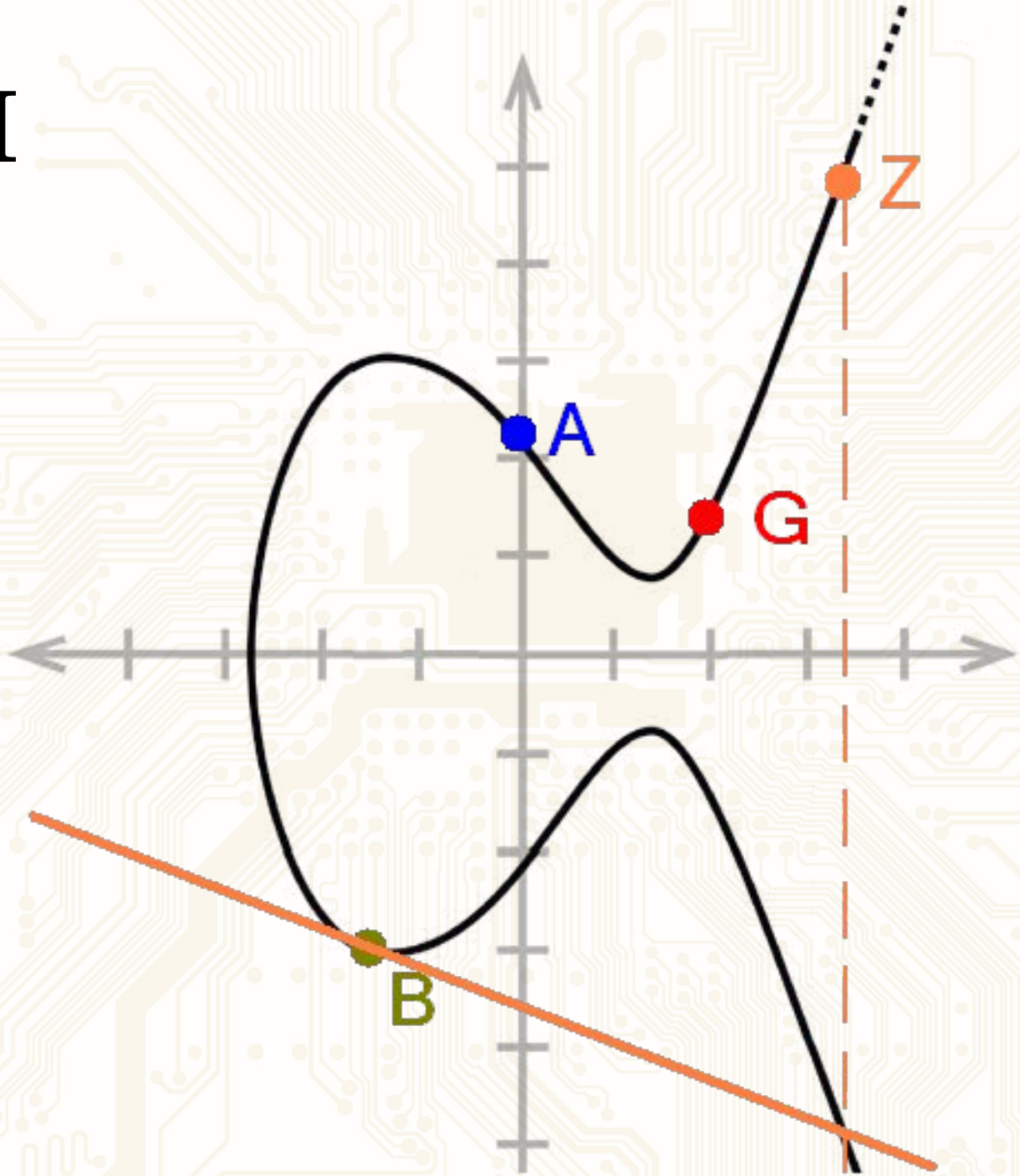
Bob nalezl bod B.

ECDH



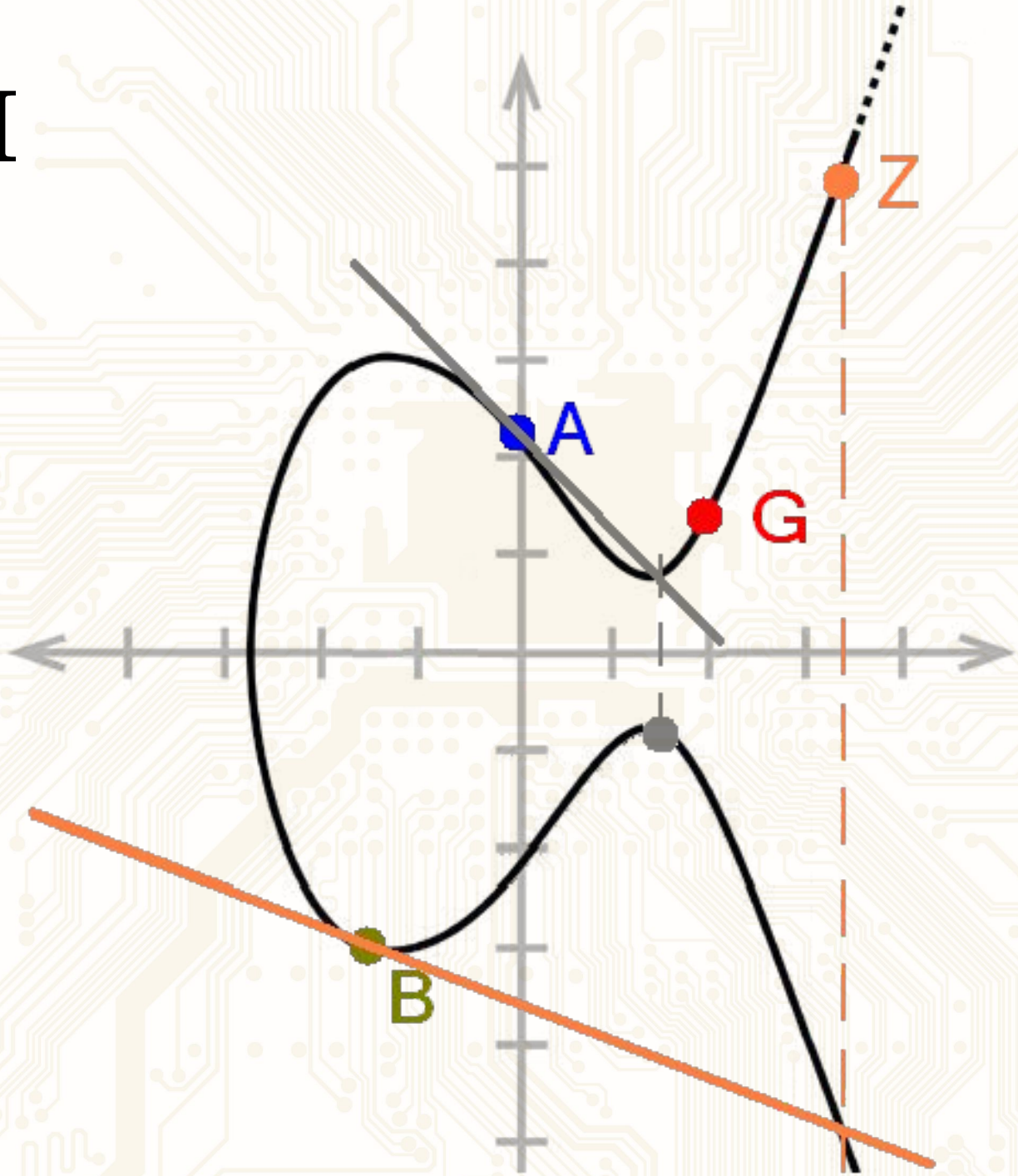
Body A, B se zveřejní,
a, b zůstávají tajná.

ECDH



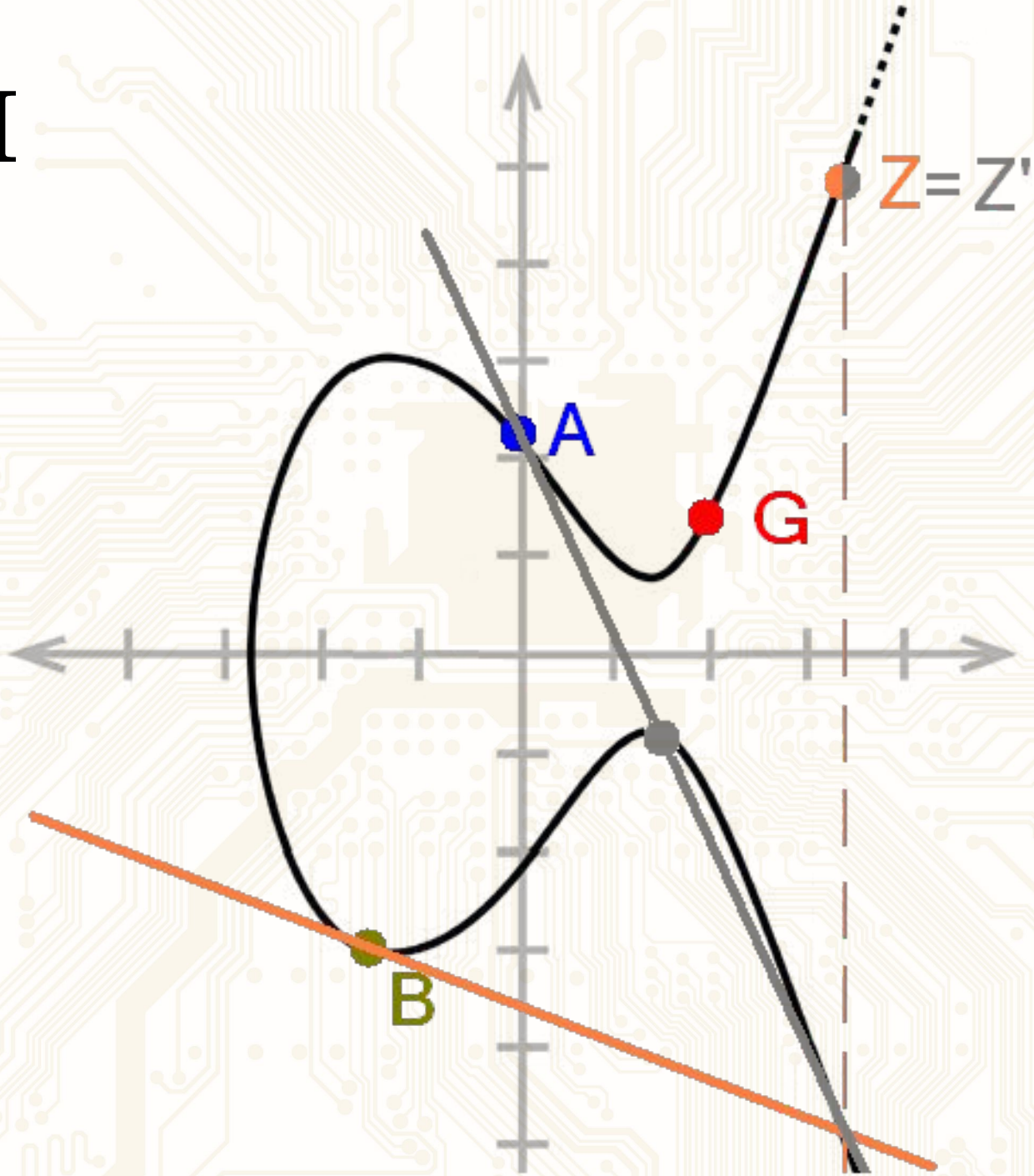
Alice může nalézt bod Z,
Z=aB.

ECDH



Bob může nalézt Z',
Z'=bA.

ECDH



Bob může nalézt Z',
Z'=bA.

Hashovací funkce

- matematická funkce, která produkuje pro data libovolné délky stejně dlouhý hash (nebo také otisk či fingerprint)
- vlastnosti
  - malá změna vstupních dat vyvolá velkou změnu v hashi
  - z hashe nelze získat původní data (na rozdíl od šifrování)
  - je minimálně pravděpodobné, že různým datům odpovídá stejný hash (hashí lze tedy identifikovat konkrétní data)

Hashovací algoritmy (příklady)

- **MD5 (rok 1991) – od r. 1996 doporučován odklon**
- **SHA-1 (rok 1995), od r. 2005 doporučován odklon**
- **SHA-2 (rok 2001)**
- **SHA-3 (rok 2015)**
- **Whirlpool (rok 2000)**, založeno na šifře Rijndael
- **Tiger (rok 1995)**
- **RIPEMD-160 (rok 1996)**, původní RIPEMD zranitelný
- …

Hashovací funkce

```
Data:   Hashuji, tedy jsem.
Hex:    48617368756a692c207465647920a73656d2e
Hashovací algoritmus: SHA-1
Hash:   2d70d1104d7bbf3192b9f07fb38136415a1247ee
```

Hashovací funkce

```
Data:   Hashuji, tedy jsem.
Hex:    48617368756a692c2074656479206a73656d2e
Hashovací algoritmus: SHA-1
Hash:   2d70d1104d7bbf3192b9f07fb38136415a1247ee

Data:   Hashujh, tedy jsem.
Hex:    48617368756a682c2074656479206a73656d2e
Hashovací algoritmus: SHA-1
Hash:   c45ff9b653a56671290f67d0476ce5fbf5ee01b7
```

Detail

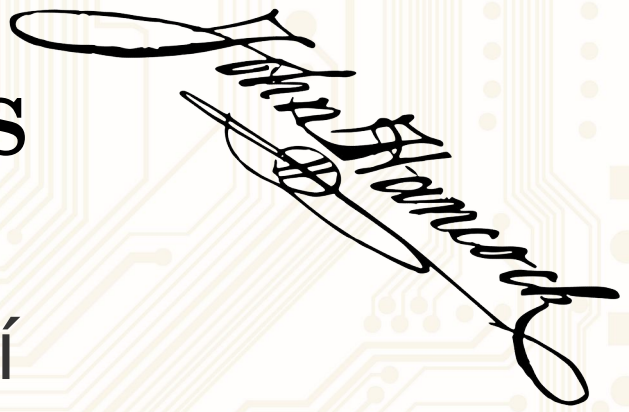
```
2d70 d110 4d7b bf31 92b9 f07f b381 3641 5a12 47ee
c45f f9b6 53a5 6671 290f 67d0 476c e5fb f5ee 01b7
```

Hashovací funkce

```
Data:   Tak jako ostatní na rotorech založené šifrovací stroje je Enigma kombinací elektrického a mechanického systému.

Hash:   145560ede2c379d63d8c1eb19ece10fd9145c415


Hash:   2d70d1104d7bbf3192b9f07fb38136415a1247ee

Data:   Hashuji, tedy jsem.
```

Zdroj: Wikipedia

Digitální podpis



- vezmu **data** připravená k odeslání
- zhotovím jejich **hash**
- **zašifruji** tento hash svým **soukromým klíčem** a pošlu jej spolu s daty

---

- všichni mají můj **veřejný klíč**
- veřejným klíčem mohou **dešifrovat hash**
- a **porovnat jej s hashem**, který si sami zhotoví
- pokud souhlasí, data nebyla cestou pozměněna

Jak ověřím identitu protistrany?

- pomocí asymetrické kryptografie mohu šifrovat i podepisovat
- jak ale vím, že něčí veřejný klíč, který se někde povaluje na internetu, nebo se jím prokáže webový server při TLS handshaku, patří skutečně tomu, za koho se dotyčný vydává?

PKI: Public Key Infrastructure

- hierarchická struktura certifikačních autorit
- **Certifikační Autorita** (CA): **ověří identitu** žadatele a vydá mu **certifikát**
- **Certifikát**: certifikační autoritou **podepsaný veřejný klíč žadatele** (formát certifikátu: standard X.509)
  - má danou časovou platnost (od kdy, do kdy, typicky rok)
- věřím-li autoritě, mohu věřit i jí vydaným certifikátům (*přenos důvěry*)
- alternativa k **PKI**: Web of Trust

Struktura X.509 verze 3

- Certifikát
  - verze certifikátu
  - sériové číslo certifikátu
  - ID podpisového algoritmu
  - jméno vydavatele
  - platnost
    - od – neplatné před datem:
    - do – neplatné po datu:
  - jméno subjektu (majitele certifikátu)
  - informace o veřejném klíči subjektu
    - algoritmus veřejného klíče
    - veřejný klíč
  - unikátní identifikátor vydavatele (volitelně)
  - unikátní identifikátor subjektu (volitelně)
  - rozšíření (volitelné)
    - ...
- Algoritmus podpisu certifikátu
- Podpis certifikátu

- Certificate
  - Version Number
  - Serial Number
  - Signature Algorithm ID
  - Issuer Name
  - Validity period
    - Not Before
    - Not After
  - Subject name
  - Subject Public Key Info
    - Public Key Algorithm
    - Subject Public Key
  - Issuer Unique Identifier (optional)
  - Subject Unique Identifier (optional)
  - Extensions (optional)
    - ...
- Certificate Signature Algorithm
- Certificate Signature

PKI: Public Key Infrastructure

- ke zvážení
  - v reálu se vás nikdo neptá – výrobce prohlížeče nebo vašeho operačního systému rozhodne, kterým CA věří (máte možnost změnit, ale kdo to dělá?)
  - na seznam důvěryhodných CA se dostávají ty, které zaplatí tučný příspěvěk (ale reputace výrobce)
  - mohou třípísmenkové organizace tlačit na CA k vydání podvržených certifikátů?
  - *hierarchický systém*

Hierarchický systém CA

- jedna CA může podepsat jinou CA
  - *přenos důvěry*: věříte-li CA, věříte i jí podepsaným CA
- OS/prohlížeče důvěřují zpravidla kořenovým CA
- takže musíte ověřovat certifikát webu, mezilehlé certifikáty a certifikační autoritu
- vytváří to komplikovanou a ne zcela jasnou síť vztahů CA

PKI: Public Key Infrastructure

- zvůle nebo kompromitace CA může mít dalekosáhlé následky
- umožňuje vytvářet **důvěryhodné parazitní certifikáty** umožňující triviálně snadné MITM útoky na šifrovaná spojení libovolné služby
- na druhou stranu když se na to přijde, CA ztratí důvěru a zpravidla končí (DigiNotar, StartCom)

TLS: ověření serveru

- v rámci TLS handshake je ověřen certifikát serveru
- certifikát serveru je vydán pro konkrétní **doménové jméno** a **na omezenou dobu**
- je-li certifikát serveru **platný** a podepsán **důvěryhodnou CA**, je spojení považováno za zabezpečené
- v opačném případě vás prohlížeč varuje

Použití sym/asym šifrování

- **asymetrické šifrování** se zřídkakdy používá k šifrování vlastních dat nebo komunikace, neboť je mnohem **pomalejší**
- asymetrické šifrování je využíváno k **autentizaci** nebo k **výměně klíčů**
- vlastní data/komunikace se pak šifrují **symetricky**

Máme šifrované spojení, co dál?

- dohodli jsme se na **širovacích metodách**
- **ověřili jsme identitu** protistrany
- bezpečně jsme si s ní **vyměnili klíče**
- zahájili jsme **šifrovanou komunikaci**
- ale co když...

Útoky na TLS

- mějme šmíráka **Š**, který zaznamenává veškerou šifrovanou komunikaci **K** serveru **S** po delší dobu
- předpokládejme, že spojení je zabezpečené pomocí privátního klíče **P**, který server používá delší dobu
- teď je klíč bezpečný, ale časem může být prolomen nebo kompromitován
  - *popř. šmírák získá přístup ke kvantovému počítači s dostatečným počtem qbitů k prolomení asymetrické šifry Shorovým algoritmem*

Revokace pub. klíče / certifikátu

- kompromitovaný klíč je možné **revokovat**
- účinky mohou být relativně okamžité (OCSP)
- nejsou však retroaktivní
- pokud někdo komunikaci zaznamenal a nyní se dostal k soukromému klíči, může zaznamenanou komunikaci dešifrovat
- **řešení**: dopředná bezpečnost

Perfect forward secrecy

- **dopředná bezpečnost** – techniky znemožňující kompromitovat relační klíče i při používání dlouhodobých tajemství (např. soukromý klíč k serverovému certifikátu)
- **ECDH<u>E</u>**: Elliptic Curve Diffie-Hellman **<u>E</u>phemeral**
  - Ephemeral (pomíjivé) varianty key exchange protokolů
- tajemství používaná pro výměnu klíčů jsou čerstvě generovaná pro každou relaci, nejsou znovupoužity a soukromý klíč serveru (dlouhodobé tajemství) je omezen na provádění autentizace

Dopředná bezpečnost a TLS

- TLS 1.2 podporuje šifrové sady s dopřednou bezpečností, ale nevyžaduje jejich použití
- TLS 1.3 vyžaduje dopřednou bezpečnost

MITM útoky na probíhající TLS

- kryptosystém se musí bránit útokům, které mohou oslabit nebo prolomit šifrování
- jak víme, že data dorazila v pořádku a nebyla na cestě pozměněna?
- šifrování samotné neřeší situace jako prohozený bit nebo data podvržená útočníkem

MITM útoky na probíhající TLS

- inu dobrá, tak šifrovaná data zahashujeme a pošleme hash spolu s nimi – stačí?

MITM útoky na probíhající TLS

- inu dobrá, tak šifrovaná data zahashujeme a pošleme hash spolu s nimi – stačí? NE
- co když útočník podvrhne vlastní data nebo data pozmění a zahashuje je sám?

MITM útoky na probíhající TLS

- inu dobrá, tak šifrovaná data zahashujeme a pošleme hash spolu s nimi – stačí? NE
- co když útočník podvrhne vlastní data nebo data pozmění a zahashuje je sám?
- inu dobrá, tak k datům, co budeme hashovat, přidáme sdílené tajemství (šifrovací klíč) – ten útočník nezná – stačí?

MITM útoky na probíhající TLS

- inu dobrá, tak šifrovaná data zahashujeme a pošleme hash spolu s nimi – stačí? NE
- co když útočník podvrhne vlastní data nebo data pozmění a zahashuje je sám?
- inu dobrá, tak k datům, co budeme hashovat, přidáme sdílené tajemství (šifrovací klíč) – ten útočník nezná – stačí? NE

MITM útoky na probíhající TLS

- co když útočník vyřadí zprávu z doručení a vloží ji do komunikace později?

MITM útoky na probíhající TLS

- co když útočník vyřadí zprávu z doručení a vloží ji do komunikace později?
- zohledníme i číslo sekvence, abychom zabránili replay útokům

Není MAC jako MAC

- **M**edium **A**ccess **C**ontrol: fyzická adresa (ethernet, linková vrstva)
- **M**essage **A**uthentication **C**ode, také **MAC**, ale v kryptografii (prezentační vrstva)

Message Authentication Code

- Message Authentication Code (či autentizační značka) jsou využívány k ověření autenticity a integrity každého zašifrovaného paketu
- **HMAC** – forma MAC kombinující hash s klíčem
- **TLS 1.3** využívá **AEAD** (authenticated encryption with associated data) algoritmy (*GCM*, *ChaCha20-Poly1305*, atd.): autentizace šifrovaných i nešifrovaných dat ve zprávě (např. číslo sekvence)

Příklad šifrové sady TLS

- Cipher Suite: TLS_AES_256_GCM_SHA384
  - **AES**: šifrovací algoritmus: AES, Rijndael
  - **256**: varianta AES s 256-bitovým klíčem
  - **GCM**: Galois/Counter Mode, šifrovací mód s AEAD
  - **SHA384**: hashovací algoritmus rodiny SHA-2
- Key Share: x25519
  - **Diffie-Hellman**ova výměna klíčů
  - použije se „osvědčená" **eliptická křivka 25519**

PKI, certifikáty, podpisy a stát

- elektronický podpis zrealizujete s jakýmkoliv certifikátem, ale stát vám pro komunikaci s ním obvykle jen tak nějaký neakceptuje ČR/EU nejsou výjimka
- konvenční komerční CA stát neuznává
- státem uznávané CA naopak nemusí být důvěryhodné ve vašem prohlížeči
  - dnes je snad situace lepší

Kvalifikovaný certifikát

- certifikát vydaný **kvalifikovanou** CA
  - musí splňovat zákonné požadavky
  - v ČR momentálně asi 3 subjekty
- pro elektronickou komunikaci s úřady v ČR je vyžadován **kvalifikovaný certifikát**

Elektronický podpis a legislativa

- **Prostý** elektronický podpis 
  - patička e-mailu, zaškrtnutí souhlasu na webu, atd.
- **Zaručený** elektronický podpis
  - asymetrické šifrování, jakýkoliv certifikát, i self-signed
- **Uznávaný** elektronický podpis (stát uznává)
  - zaručený el. podpis založený na kvalifikovaném certifikátu
- **Kvalifikovaný** elektronický podpis (nejvyšší úroveň)
  - založený na kvalifikovaném certifikátu a kvalifikovaném prostředku (schválená čipová karta, apod.)

SSL strip útok

- většina webů dnes využívá **HTTPS**, ale prohlížeče se při prvním přístupu na webovou stránku standardně připojují na port **80** přes nezabepzečené **HTTP**
- webový server klienty přistupující na port 80 po zadání požadavku přesměruje na **HTTPS** variantu webu

Konfigurace webserveru (Apache)

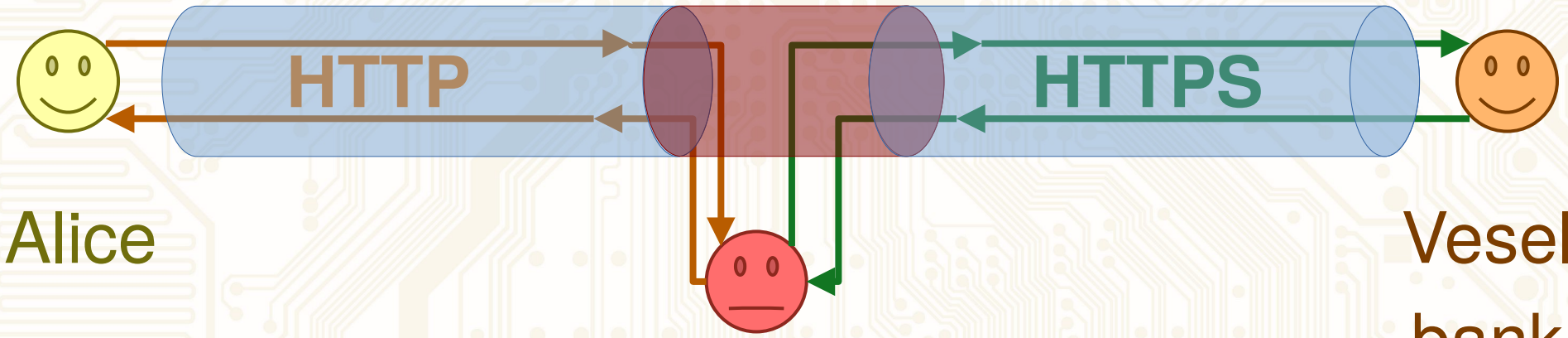
```
<VirtualHost *:80>
        ServerName 4iz110.vse.cz
        RewriteEngine On
        RewriteRule ^(.*) https://4iz110.vse.cz/$1 [R=301,L]
        ...
</VirtualHost>
```

HTTP komunikace (port 80)

```
GET / HTTP/1.1
Host: krkavec.net

HTTP/1.1 301 Moved Permanently
Location: https://krkavec.net/
```

SSL strip útok

- co když ale MITM útočník převezme roli webového serveru a místo přesměrování na šifrovanou variantu nechá obět komunikovat nešifrovaně?



HTTP Strict Transport Security

- součástí odpovědi na HTTP request je hlavička HSTS
- hlavička specifikuje, jak dlouho se má přistupovat na cílový web přes HTTPS a zda-li to platí i na subdomény
- prohlížeč si to zapamatuje, HTTPS pak vyžaduje a nepovolí downgrade na HTTP za žádných okolností
- příklad hlavičky HSTS:

```
Strict-Transport-Security: max-age=31536000
```

HTTP Strict Transport Security

- **problém**: stále musí proběhnout alespoň jeden útočníkem nenarušený cyklus Request → Response (úplně první spojení je zranitelné)
- ***řešení***: některé prohlížeče mají navíc zadrátovaný seznam domén využívajících HSTS, do seznamu bývá možné se veřejně přidat

SNI: Server Name Indication

- HTTPS spojení **nejprve** navazuje zabezpečené spojení a teprve poté proběhne HTTP **Request**
- konkrétní **certifikát** je vázán na doménu
- hostingové služby provozují stovky/tisíce domén na **jediném** serveru (a tedy jediné IP adrese)
- hlavička **Host:** je povinná, ale uplatní se až **při HTTP Requestu**, zatímco certifikát je potřeba již při navazování zabezpečeného spojení

SNI: Server Name Indication

- Server Name Indication je **rozšíření TLS** umožňující serveru sdělit doménové jméno požadované webové stránky
- server dle toho vrátí příslušný certifikát vydaný pro danou doménu
- ověření identity proběhne korektně
- **problém**: toto jméno se přenáší nešifrovaně (information leak)

TLSv1.3 Client Hello (RAW data)

```
0000   00 16 3c 46 3b 45 cc e1 7f c6 a0 01 08 00 45 00   ..<F;E........E.
0010   01 3b bc c7 40 00 38 06 63 d2 95 48 b0 f5 1f 07   .;..@.8.c..H....
0020   bb de cd 54 00 19 43 8e 69 30 84 27 5e 1c 80 18   ...T..C.i0.'^...
0030   00 3c 99 03 00 00 01 01 08 0a c8 f2 d9 e2 30 9a   .<............0.
0040   fa b1 16 03 01 01 02 01 00 00 fe 03 03 eb 75 66   ..............uf
0050   d2 74 47 3a 66 60 96 14 1c 4f b3 27 a4 5d 3b cc   .tG:f`...O.'.];.
0060   af bf 75 2e 9a 96 82 d6 59 85 be 34 9b 20 83 94   ..u.....Y..4. ..
0070   ce 9d 29 af a2 bd af 76 6e 1a e6 05 07 7a 73 3e   ..).....vn....zs>
0080   e2 a4 9c 63 12 37 33 1d 12 62 d0 44 85 60 00 28   ...c.73..b.D.`.(
0090   c0 2b c0 2f c0 2c c0 30 cc a9 cc a8 c0 09 c0 13   .+./.,.0........
00a0   c0 0a c0 14 00 9c 00 9d 00 2f 00 35 c0 23 c0 27   ........./.5.#.'
00b0   00 3c 13 01 13 02 13 03 01 00 00 8d 00 00 00 10   .<..............
00c0   00 0e 00 00 0b 6b 72 6b 61 76 65 63 2e 6e 65 74   .....krkavec.net
00d0   00 05 00 05 01 00 00 00 00 00 0a 00 0a 00 08 00   ................
00e0   1d 00 17 00 18 00 19 00 0b 00 02 01 00 00 0d 00   ................
00f0   1a 00 18 08 04 04 03 08 07 08 05 08 06 04 01 05   ................
0100   01 06 01 05 03 06 03 02 01 02 03 ff 01 00 01 00   ................
0110   00 12 00 00 00 2b 00 07 06 03 04 03 03 03 02 00   .....+..........
0120   33 00 26 00 24 00 1d 00 20 08 f1 60 64 1b 8e 4c   3.&.$... ..`d..L
0130   0d d1 67 9f fc b1 60 1b 96 09 a4 4c d2 48 76 ea   ..g...`....L.Hv.
0140   c4 cd 22 42 54 ee b5 06 61                        ..."BT...a
```

TLSv1.3 Client Hello (RAW data)

```
0000   00 16 3c 46 3b 45 cc e1 7f c6 a0 01 08 00 45 00   ..<F;E........E.
0010   01 3b bc c7 40 00 38 06 63 d2 95 48 b0 f5 1f 07   .;..@.8.c..H....
0020   bb de cd 54 00 19 43 8e 69 30 84 27 5e 1c 80 18   ...T..C.i0.'^...
0030   00 3c 99 03 00 00 01 01 08 0a c8 f2 d9 e2 30 9a   .<............0.
0040   fa b1 16 03 01 01 02 01 00 00 fe 03 03 eb 75 66   ..............uf
0050   d2 74 47 3a 66 60 96 14 1c 4f b3 27 a4 5d 3b cc   .tG:f`...O.'.];.
0060   af bf 75 2e 9a 96 82 d6 59 85 be 34 9b 20 83 94   ..u.....Y..4. ..
0070   ce 9d 29 af a2 bd af 76 6e 1a e6 05 07 7a 73 3e   ..).....vn....zs>
0080   e2 a4 9c 63 12 37 33 1d 12 62 d0 44 85 60 00 28   ...c.73..b.D.`.(
0090   c0 2b c0 2f c0 2c c0 30 cc a9 cc a8 c0 09 c0 13   .+./.,.0........
00a0   c0 0a c0 14 00 9c 00 9d 00 2f 00 35 c0 23 c0 27   ........./.5.#.'
00b0   00 3c 13 01 13 02 13 03 01 00 00 8d 00 00 00 10   .<..............
00c0   00 0e 00 00 0b 6b 72 6b 61 76 65 63 2e 6e 65 74   .....krkavec.net
00d0   00 05 00 05 01 00 00 00 00 00 0a 00 0a 00 08 00   ................
00e0   1d 00 17 00 18 00 19 00 0b 00 02 01 00 00 0d 00   ................
00f0   1a 00 18 08 04 04 03 08 07 08 05 08 06 04 01 05   ................
0100   01 06 01 05 03 06 03 02 01 02 03 ff 01 00 01 00   ................
0110   00 12 00 00 00 2b 00 07 06 03 04 03 03 03 02 00   .....+..........
0120   33 00 26 00 24 00 1d 00 20 08 f1 60 64 1b 8e 4c   3.&.$... ..`d..L
0130   0d d1 67 9f fc b1 60 1b 96 09 a4 4c d2 48 76 ea   ..g...`....L.Hv.
0140   c4 cd 22 42 54 ee b5 06 61                        ..."BT...a
```

TLS Encrypted Client Hello

- rozšíření TLS protokolu umožňující šifrovat Client Hello včetně dat rozšíření Server Name
- zatím ale jen draft
- cenzorům se toto rozšíření nelíbí již dnes, a blokují ho

Steganografie

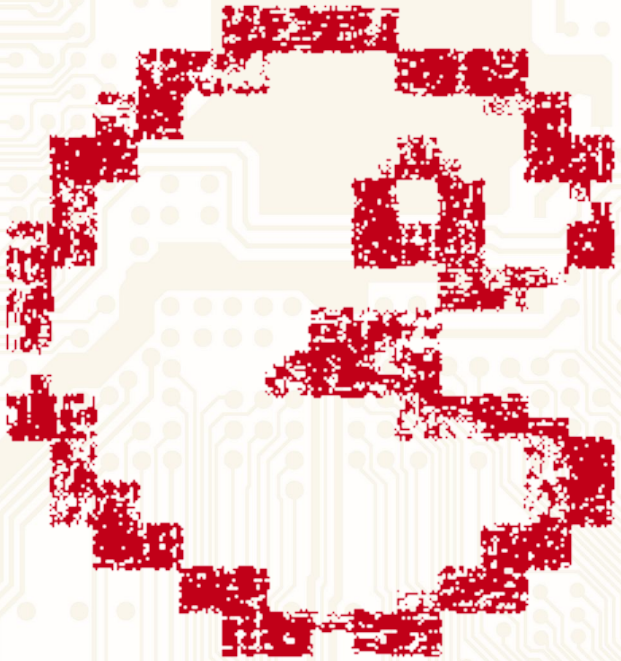
- nauka o skrývání dat v jiných datech, příklad:
- obrázek: mřížka pixelů, každý má barevnou hodnotu složek R,G,B
- využívají se LSB: Least Significant Bity:

- do LSB se zakóduje (nejlépe zašifrovaná) zpráva
- tato steganografická technika jde odhalit

Původní

Se skrytými daty

Rozdíl mezi obrázky



Popiratelné šifrování

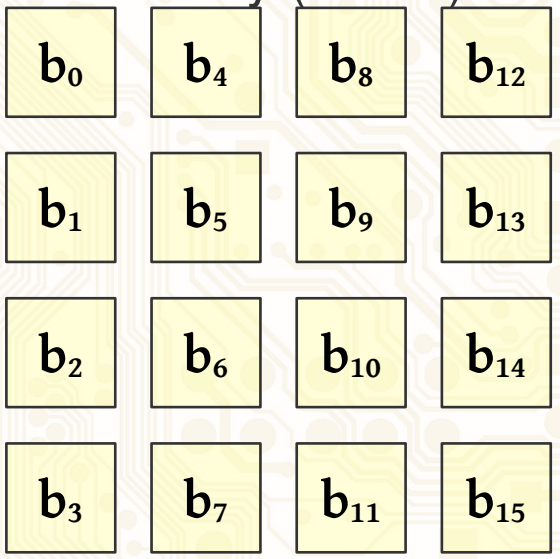
- *deniable encryption*
- kombinace **kryptografie** a **steganografie**
- využití steganografie ke skrytí zašifrovaných dat tak, že je možné **hodnověrně popřít** existenci odpovídajícího plain textu
- např. VeraCrypt, apod.

Bezpečnost šifrování

- **fyzická bezpečnost PC** – keyloggery, kamery, bezdrátové klávesnice, postranní kanály, zlá pokojská (evil maid), zlý opravář,…
- **bezpečnost OS** – bezpečnost služeb i nastavení, a také absence malwaru nebo v OS vestavěných bezpečnost/soukromí narušujících funkcí
- **bezpečnost komunikačních partnerů** – když je partner kompromitován, jste kompromitování i vy (resp. vaše komunikace s daným partnerem)
- **bezpečnost mezičlánků** – mezičlánky v podobě používaných externích služeb pro přenos komunikace mohou narušovat soukromí i provádět MITM
- **bezpečnost užití šifrování** (OPSEC) – špatné užití šifrování může vést k prolomení bezpečnosti
- **právní hledisko** – někde může být trestné mít zašifrovaná data či nevydat klíč v případě oficiální výzvy orgánů činných v trestním řízení
- **hadicová kryptoanalýza** – použití násilí pro získání hesla

Rijndael (AES): princip

- Rijndael je založený na principu substitučně–permutační sítě
- pracuje v matici 4x4 bytů (16 bytů je velikost bloku)
- z klíče se odvodí podklíče pro jednotlivé iterace/rundy (round)
- **inicializační část**: **`AddRoundKey`**: každý byte aktuálního stavu je bitově XORován se všemi byty podklíče
- **iterační část** (9-13 rund/iterací dle délky klíče)
  - **`SubBytes`**: záměna každého bytu za jiný dle vyhledávací tabulky (S-box)
  - **`ShiftRows`**: prohození řádků
  - **`MixColumns`**: promíchání sloupců
  - **`AddRoundKey`**: přidání podklíče
- **závěrečná část**
  - **`SubBytes`**, **`ShiftRows`**, **`AddRoundKey`**



Substituce pomocí S-BOXu

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Kryptografická sůl (salt)

- náhodná data, která se přidávají jako extra vstup k jednosměrným funkcím (jako hash)
- hash(*salt*+*heslo*) = bezpečněji uložené heslo
- **Rainbow tables** = předspočítané hodnoty hashí (mnoho TB dat), v nich se pak dá rychle vyhledávat (existují i online služby!)
- sůl neutralizuje použití Rainbow tables

Funkce pro odvození klíče

- komplikují lámání hesel tím, že jej činí výpočetně složitější (Argon2, PBKDF2, atd.)
- spočítá se hash z hesla, pak hash z hashe, což se opakuje do určitého počtu iterací
- brute force útoku pak každý pokus o prolomení hesla trvá déle
- o něco zesilují slabší hesla
- nespoléhejte na to a používejte silná hesla!

Ukládání hesel a webové aplikace

- hesla by webové aplikace nikdy neměly ukládat v textové podobě, měla by být individuálně solena (1 heslo, 1 sůl) a hashována
- webové frameworky a programovací jazyky určené pro programování webových aplikací dnes už mívají vestavěné funkce
- PHP: `password_hash()`, `password_verify()`

Cloud

- poskytovatelé cloudu mají zpravidla větší množství serverů, na kterých spravují a poskytují cloudové služby:
  - **SaaS**: **S**oftware **a**s **a** **S**ervice – provider poskytuje aplikaci
  - **PaaS**: **P**latform **a**s **a** **S**ervice – provider poskytuje platformu, na které si provozujete nebo stavíte vlastní aplikaci
  - **IaaS**: **I**nfrastructure **a**s **a** **S**ervice – provider poskytuje infrastrukturu, na které si konfigurujete a provozujete, co chcete (např. VM)
- okamžité zřízení, velká škálovatelnost, zpravidla levnější než provozovat ekvivalent vlastními silami, nemusíte se starat o hardware, robustnost (odolnost vůči výpadkům), automatizace,…

Cloud

- poskytovatelé cloudu mají zpravidla větší množství serverů, na kterých spravují a poskytují cloudové služby:
  - **SaaS**: **S**oftware **a**s **a** **S**ervice – provider poskytuje aplikaci
  - **PaaS**: **P**latform **a**s **a** **S**ervice – provider poskytuje platformu, na které si provozujete nebo stavíte vlastní aplikaci
  - **IaaS**: **I**nfrastructure **a**s **a** **S**ervice – provider poskytuje infrastrukturu, na které si konfigurujete a provozujete, co chcete (např. VM)
- okamžité zřízení, velká škálovatelnost, zpravidla levnější než provozovat ekvivalent vlastními silami, nemusíte se starat o hardware, robustnost (odolnost vůči výpadkům), automatizace,…
- *počítače někoho jiného, nad kterými nemáte kontrolu*