Elliptikus görbék

PPKE, ITK
Csapodi Márton

Használatuk

Az elliptikus görbék kriptográfiai alkalmazását 1985-ben két kutató javasolta:

- Neal Koblitz (University of Washington)
- Victor Miller (IBM)

A jelenlegi nyilvános kulcsú eljárások elsősorban:

- egészek faktorizációja
- diszkrét logaritmus

problémákon alapulnak.

A diszkrét logaritmus számítható "elliptikus görbe felett".

Használatuk

Az alábbi táblázatban három eljárás általánosan vagy szabvány szerint használt kulcsméreteit láthatjuk (ECC: elliptikus görbe feletti diszkrét logaritmus alapú).

NIST·javaslata·az·egyes·kriptorendszerek·kulcshosszának·összehasonlítására:¤			
ECC·modulus¤	AES¤	RSA↵modulus¤	RSA:ECC¤
112¤	56¤	512¤	5:1¤
161¤	80¤	1024¤	6:1¤
256¤	128¤	3072¤	12:1¤
384¤	192¤	7680¤	20:1¤
512¤	256¤	15630¤	30:1¤

Az egy sorban lévő kulcsméretek közel azonos biztonságot nyújtanak (a NIST ajánlása szerint).

Használatuk

Az ECC sokkal kisebb kulcsmérettel is biztonságos:

- gyorsabb algoritmusok
- kevesebb továbbítandó és kezelendő adat
- kisebb tárigény
- kisebb tanúsítványok.

Használatuk a nyilvános kulcsú eljárásokon kívül:

- prímtényezőkre bontás
- prímteszt

Definíció

Elliptikus görbe: $y^2 = f(x)$ alakú egyenlettel definiált síkbeli görbe, ahol

- f(x) egy $x^3+ax+ b$ alakú kifejezés
- x, y valós változók (egyelőre)
- a, b egész számok (egyelőre)
- a görbe nemszinguláris

Általános képlet: $y^2+a_1xy+a_2y=x^3+a_3x^2+a_4x+a_5$

Izomorfizmus erejéig visszavezethető az előbbire.

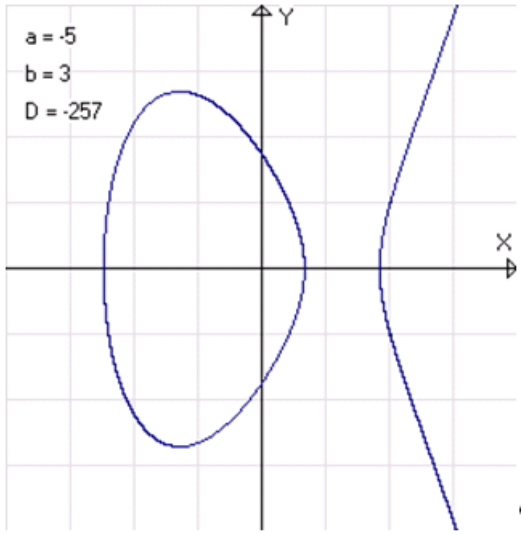
(a, b) változtatása

1. a=0, b=0
2. a=0, b=1
3. a=0, b=-1
4. a=1, b=0
5. a=-1, b=0
6. a=1, b=1
7. a=1, b=-1
8. a=-1, b=1
9. a=-1, b=-1

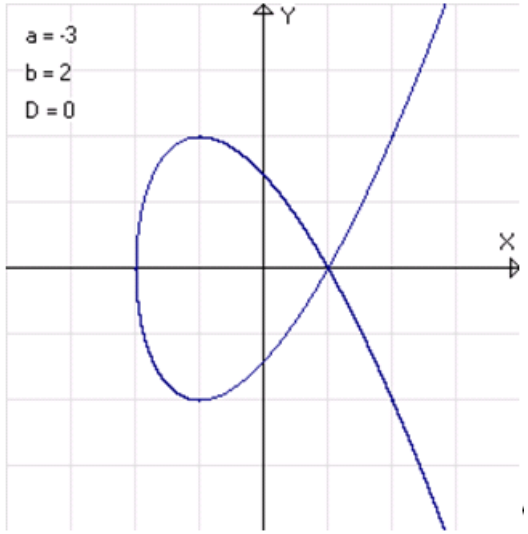
diszkrimináns

diszkrimináns: $D = 4*a^3 + 27*b^2$

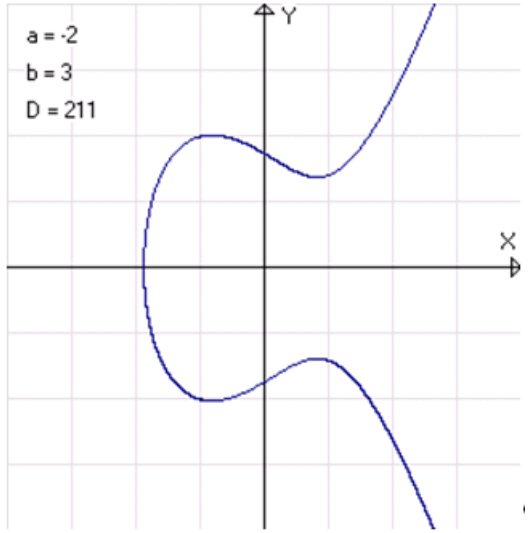
a=-5, b=3



a=-3, b=2

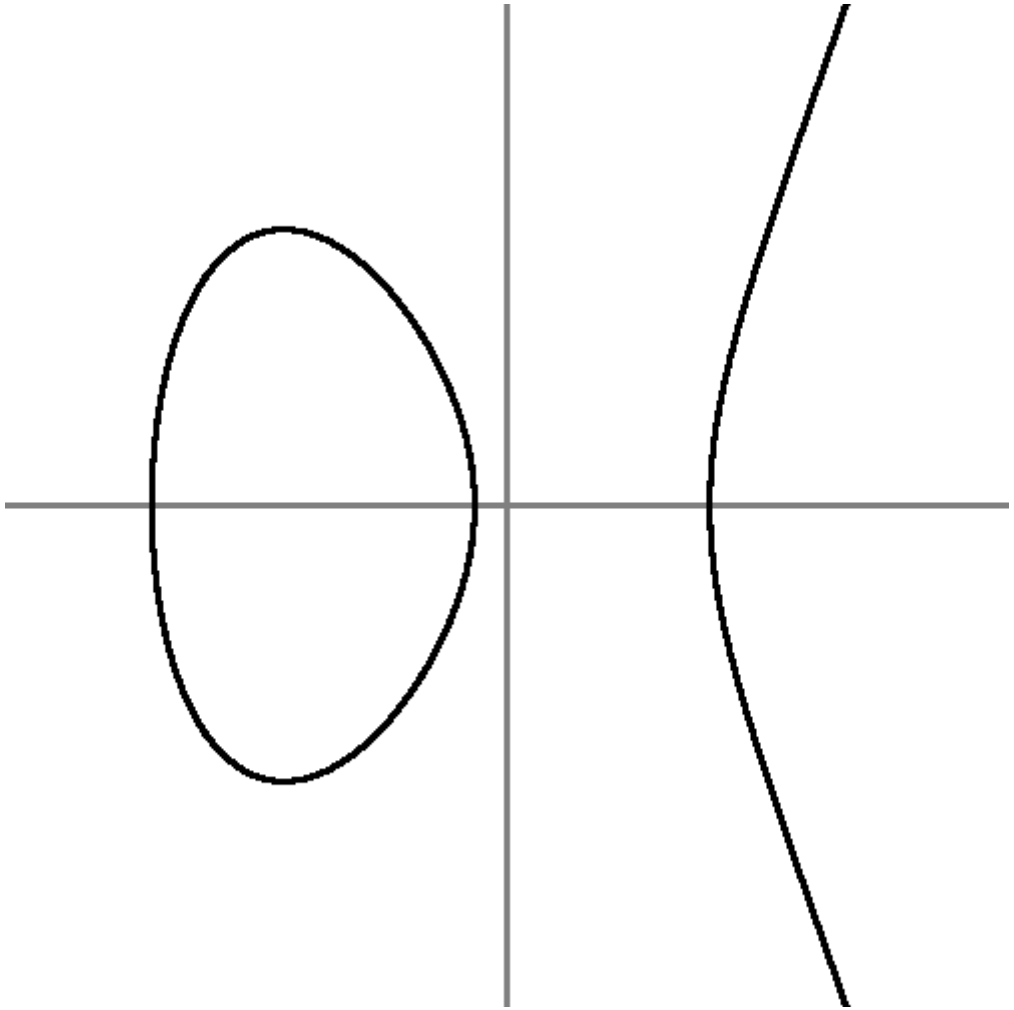


a=-2, b=3



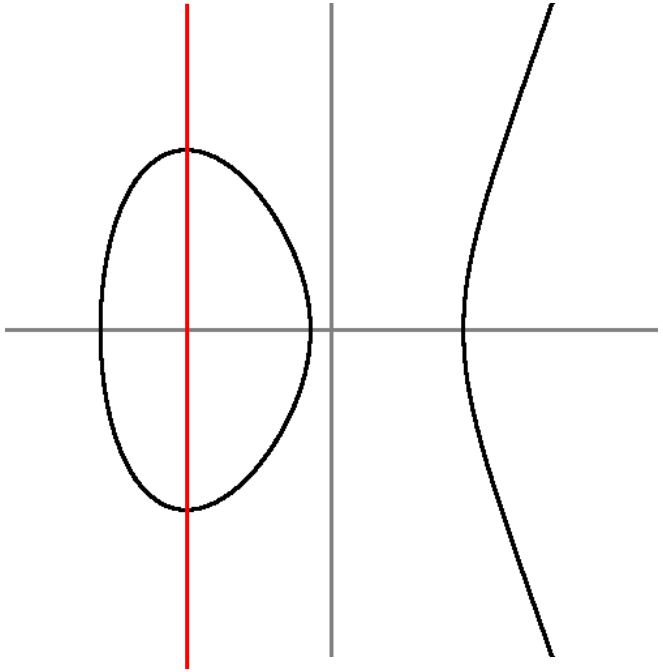
D = 0 esetén szinguláris

(a, b) változtatása



Húrok, érintők, ideális pont

Az érintő és metszéspontok számát tekintve a függőleges egyenes másképpen viselkedik



Legyen a görbe része a függőleges egyenesek ideális (végtelen távoli) pontja is

Ekkor már egyformán viselkednek (az érintő két metszéspontnak számít).

Műveletek

A görbe pontjain műveletet értelmezünk.

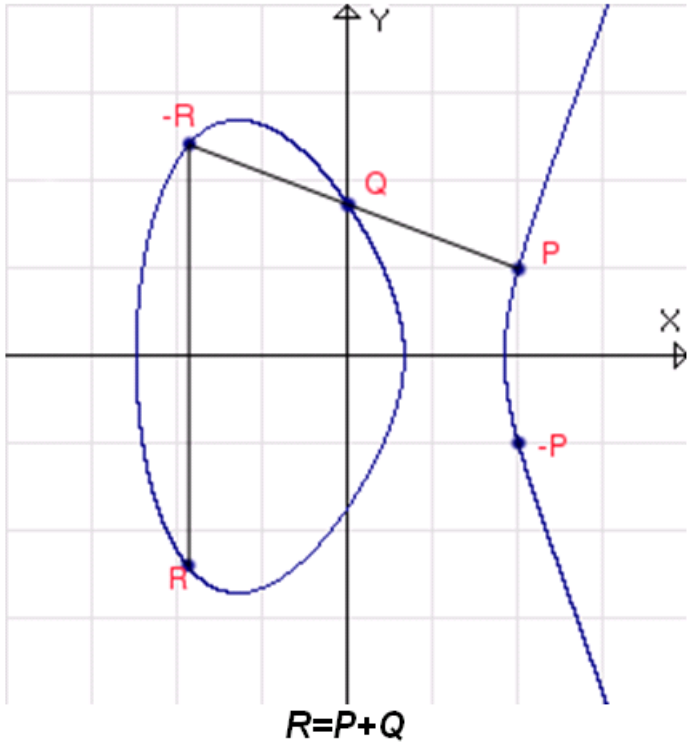
Az ideális pontról feltételezzük, hogy rajta van minden függőleges (az y-tengellyel párhuzamos) egyenesen és hogy az x-tengelyre vonatkozó tükörképe önmaga.

A görbe P és Q pontjainak P + Q összege:

a P, Q pontokon átmenő egyenes és a görbe harmadik metszéspontja -R; ennek az x-tengelyre való R tükörképe (ami szintén pontja a görbének) a P + Q összeg.

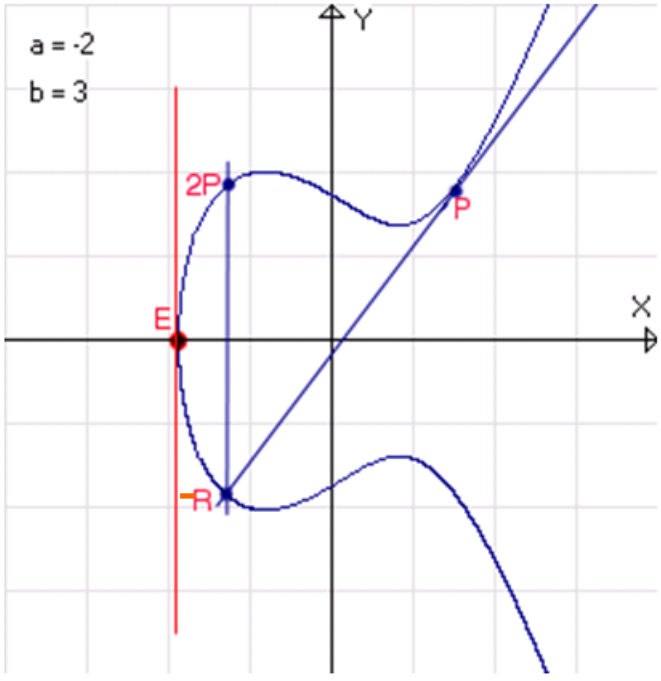
R + -R =?

az ideális pont (O)



Műveletek

Ha P = Q, akkor az összekötő egyenesükön a görbe P-beli érintőjét kell érteni.



Ha -R megegyezik a P, Q pontok valamelyikével, ekkor az egyenes -R-ben érinti a görbét.

Műveletek

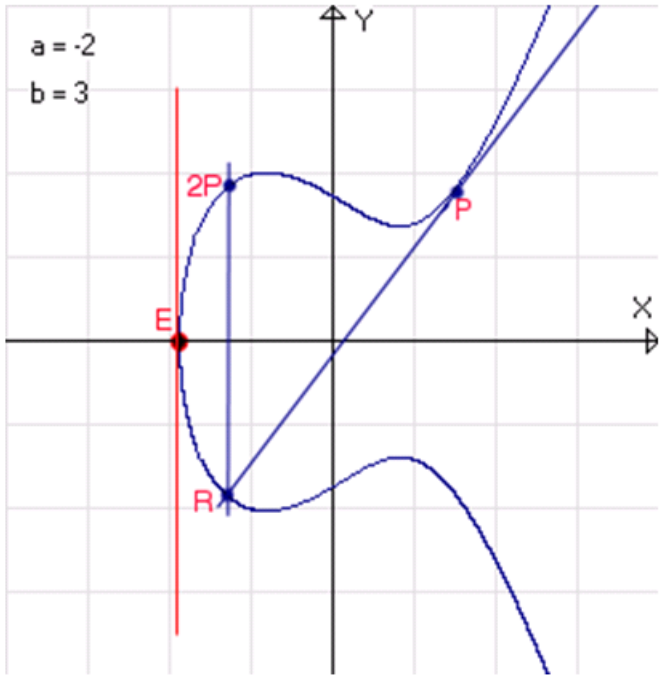
O + O = O

P + O = P

2*E = E + E = O

3*E = E + E + E = E

4*E = E + E + E + E = O



Csoport tulajdonság

Van egységelem: O

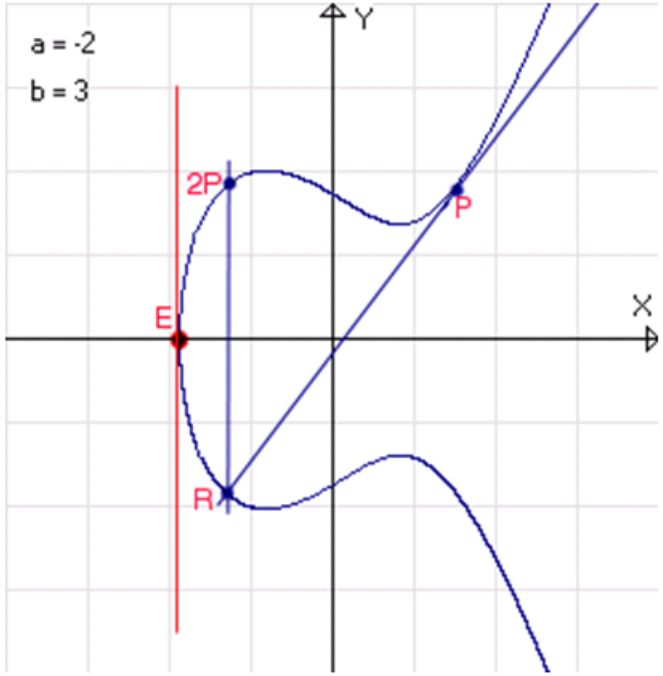
Van inverz: tükörkép

Asszociatív: (A + B) + C = A + (B + C) (nem bizonyítjuk)

Miért fontos?

gyorsan számítható az összeadás:

29*A = 2*2*2*2*A + 2*2*2*A + 2*2*A + A



Algebrai formula az összegre

Ha

P: (x_P, y_P)

Q: (x_Q, y_Q)

R: (x_R, y_R)

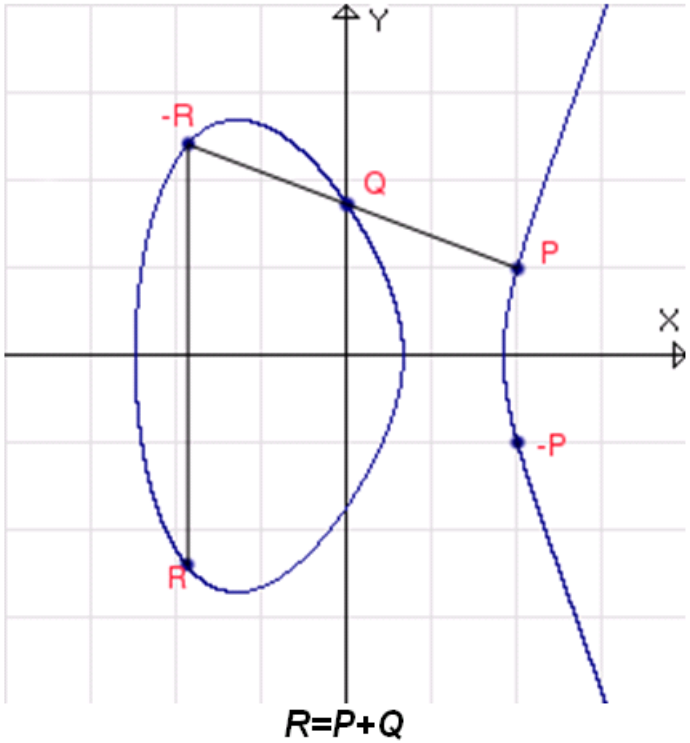
R = P + Q esetén:

$x_R = m^2 - x_P - x_Q$

$y_R = m*(x_P - x_R) - y_P$

ahol: $m = (y_P - y_Q) / (x_P - x_Q)$

az átmenő egyenes meredeksége



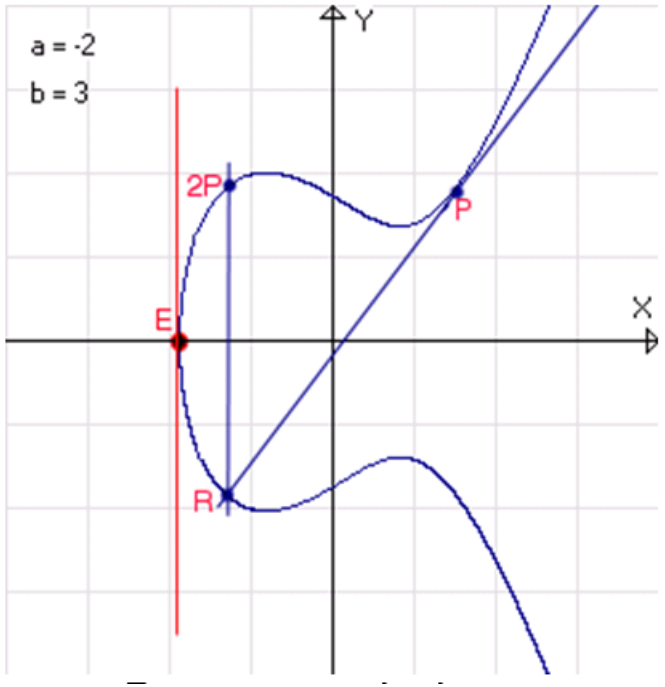
Algebrai formula az összegre

$R = 2*P$

$x_R = m^2 - 2*x_P$

$y_R = m*(x_P - x_R) - y_P$

ahol: $m = (3*x_P^2 + a) / (2*y_P)$



Elliptikus görbék mod p

Eddig:

x, y valós változók

a, b egész számok

Mostantól:

x, y mod p egészek

a, b mod p egészek

ahol p: prím

Valós számok / mod p egészek:

Mindkettő test, tehát ugyanúgy lehet számolni a görbe pontjaival

A diszkrimináns: $D = 4*a^3 + 27*b^2$ itt se lehet 0

Elliptikus görbék mod p

Miért ezzel számolunk:

- A valós számokkal való számolás lassú és pontatlan. A moduláris aritmetika gyors és pontos, csak egész számokkal dolgozik.
- A „valós” görbének végtelen sok pontja van, a modulárisnak véges.
- A moduláris aritmetikában behatárolható a számok értelmezési tartománya, mert a műveletek operandusa(i) és eredménye mindig $0 \leq x \leq p-1$ közé esik.

A "görbe" alakja

legyen

p = 11

a = 1

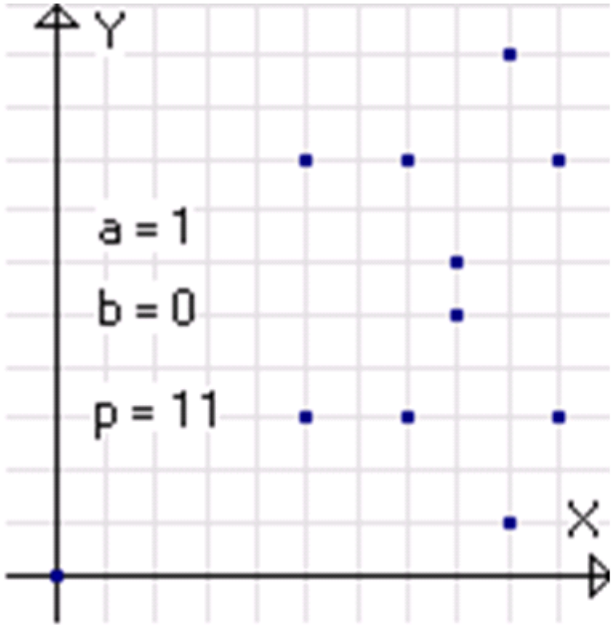
b = 0

$y^2 \equiv x^3 + x \bmod 11$

11 pont (nem mindig p-vel egyenlő!)

az x = y = 0 kivételével szimmetrikus

(minden x-hez 2 y érték)



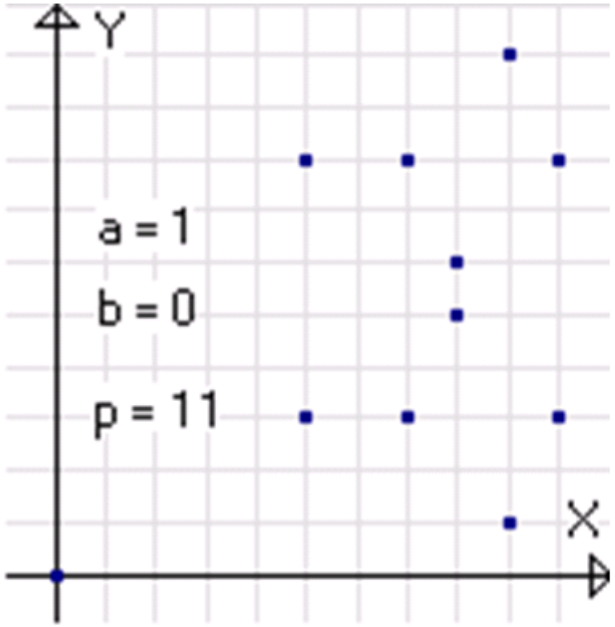
Műveletek

P: (x_P, y_P)
Q: (x_Q, y_Q)
R: (x_R, y_R)

R = P + Q esetén:

$x_R \equiv m^2 - x_P - x_Q \mod p$
$y_R \equiv m*(x_P - x_R) - y_P \mod p$

ahol: $m \equiv (y_P - y_Q) * (x_P - x_Q)^{-1} \mod p$



Műveletek

P: (x_P, y_P)

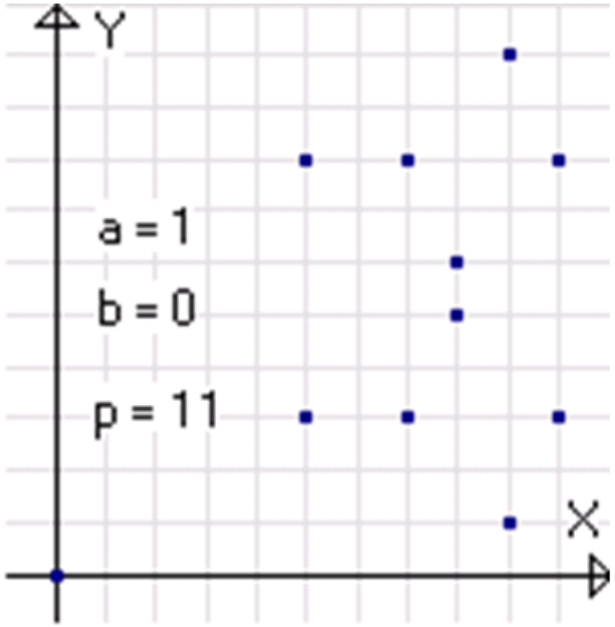
-P: $(x_P, -y_P \bmod p)$

$R = 2*P : (x_R, y_R)$

$x_R \equiv m^2 - 2*x_P \bmod p$

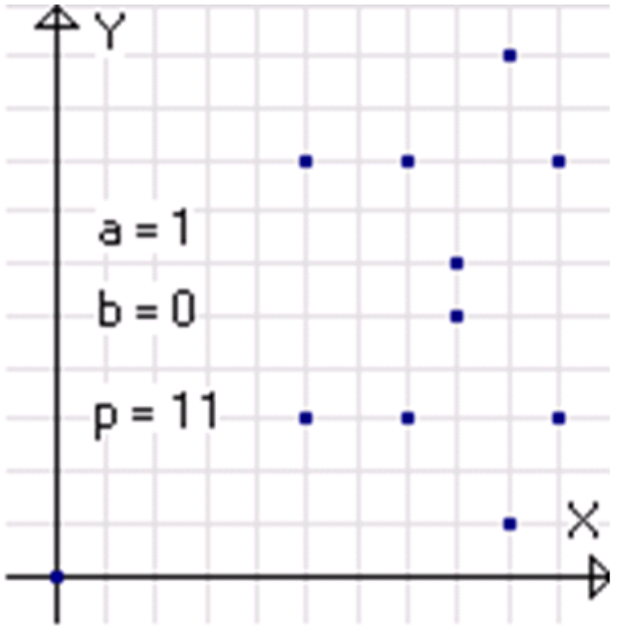
$y_R \equiv m*(x_P - x_R) - y_P \bmod p$

ahol: $m \equiv (3*x_P^2 + a) * (2*y_P)^{-1} \bmod p$



Generátor pont

p-szer összeadva előállít minden pontot



Példa

Legyen egy elliptikus görbe egyenlete:

$y^2 = x^3 + 9x + 17 \bmod 23$

Mennyi az R=(16,5) alapú diszkrét logaritmusa Q = (4,5) pontnak? (Q=nR)

Első megközelítésben n kiszámolásához addig adogassuk össze a R pontot önmagával, amíg meg nem kapjuk Q-t:

1R=(16,5) 2R=(20,20) 3R=(14,14) 4R=(19,20)

5R=(13,10) 6R=(7,3) 7R=(8,7) 8R=(12,17)

9R=(4,5)

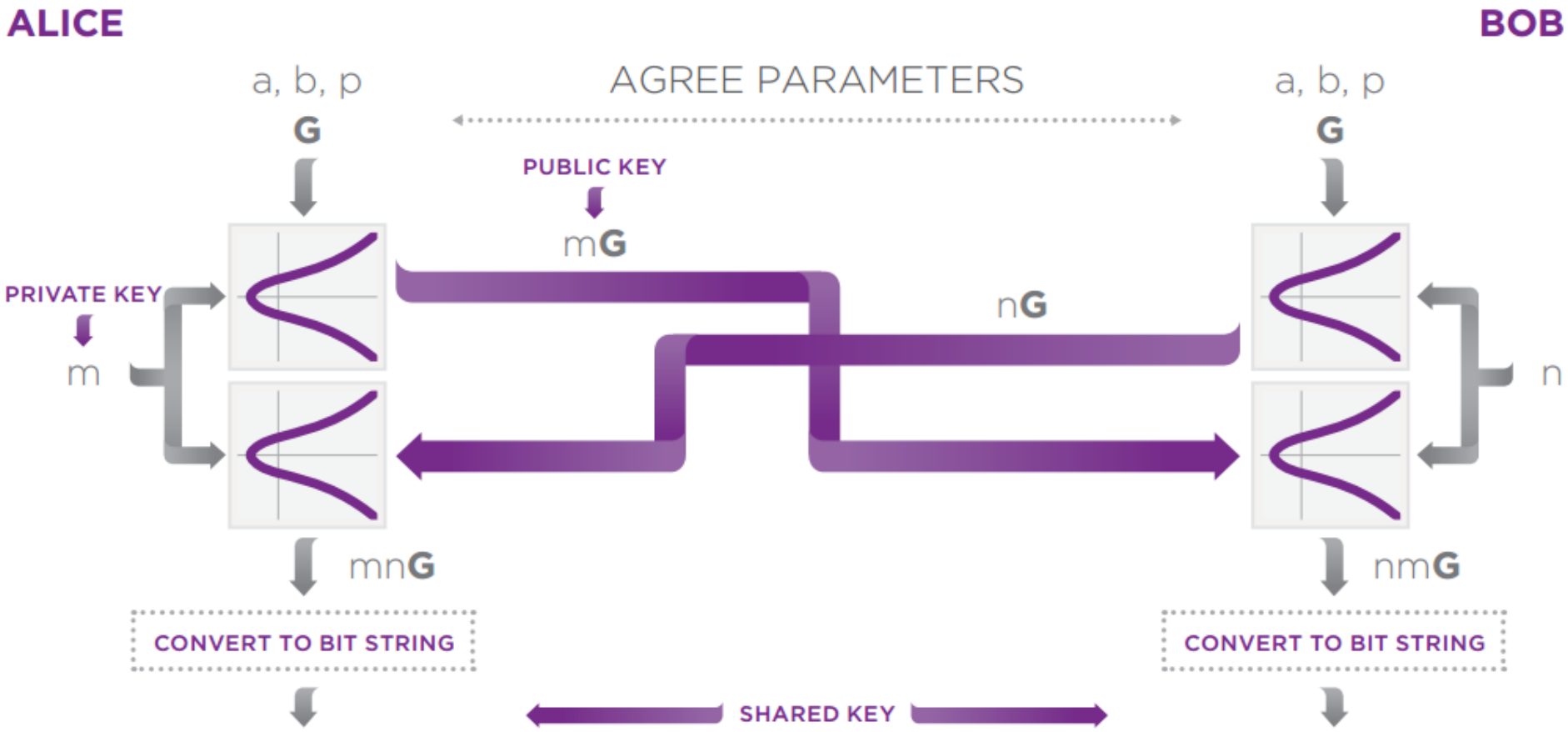
Mivel Q(4,5)=9R, ezért a Q pont R alapú diszkrét logaritmusa mod 23 felett: 9.

Miért hívjuk ezt logaritmusnak?

Lehetne osztásnak is hívni, mert a művelet összeadások sorozata.

A mod p egészek feletti diszkrét logaritmus problémához való hasonlóság miatt hívjuk itt is így.

ECDH
Elliptic-curve Diffie-Hellman



ECDSA: Elliptic-curve Digital Signature Algorithm

Alice wants to sign a message with her private key (d_A), and Bob wants to validate the signature using Alice's public key ($H_A = d_A G$).

The algorithm performed by Alice to sign the message works as follows:

1. Take a random integer k chosen from {1,…,n−1} (where n is the subgroup order).
2. Calculate the point P=kG (where G is the base point of the subgroup).
3. Calculate the number $r=x_P \bmod n$ (where x_P is the x coordinate of P).
4. If r=0, then choose another k and try again.
5. Calculate $s=k^{-1}(z+rd_A) \bmod n$ (where k^{-1} is the multiplicative inverse of k modulo n and z is the hash of the message to be signed).
6. If s=0, then choose another k and try again.

The pair (r,s) is the signature.

ECDSA: Elliptic-curve Digital Signature Algorithm

In order to verify the signature we'll need Alice's public key H_A, the hash of the message z and, obviously, the signature (r,s).

1. Calculate the integer $u_1=s^{-1}z \bmod n$
2. Calculate the integer $u_2=s^{-1}r \bmod n$
3. Calculate the point $P=u_1G+u_2H_A$

The signature is valid only if $r=x_P \bmod n$

$P=u_1G+u_2H_A=u_1G+u_2d_AG=(u_1+u_2d_A)G=(s^{-1}z+s^{-1}rd_A)G=s^{-1}(z+rd_A)G=kG$

mivel: $s=k^{-1}(z+rd_A) \bmod n$, vagyis $k=s^{-1}(z+rd_A) \bmod n$

GSM Security

PPKE, ITK

Csapodi Márton

GSM Evolution

2G (1991-)

- GSM

2.5G (1999-)

- GPRS: MMS, WAP

3G

- EDGE (2003-)
- UMTS (WCDMA, HSDPA , HSUPA, HSPA+)

4G

- LTE

GSM Security Goals

- Operators
  - bills right people
  - avoid fraud
  - protect services
- Customers
  - privacy
  - anonymity
- Make a system at least as secure as PSTN

GSM Security Goals

- Confidentiality and anonymity on the radio path
- Strong client authentication to protect the operator against the billing fraud
- Prevention of operators from compromising of each others' security
  - inadvertently
  - competition pressure

GSM Security Design Requirements

The security mechanism

- MUST NOT
  - add significant overhead on call set up
  - increase bandwidth of the channel
  - increase error rate
  - add expensive complexity to the system
- Define security procedures
  - generation and distribution of keys
  - exchange information between operators
  - confidentiality of algorithms

GSM Security Features

- Key management is independent of equipment
  - subscribers can change handsets without compromising security
- Subscriber identity protection
  - not easy to identify the user of the system intercepting a user data
- Detection of compromised equipment
  - detection mechanism whether a mobile device was compromised or not
- Subscriber authentication
  - the operator knows for billing purposes who is using the system
- Signaling and user data protection
  - signaling and data channels are protected over the radio path

GSM Mobile Station

- Mobile Station
  - Mobile Equipment (ME)
    - Physical mobile device
    - Identifiers
      - IMEI – International Mobile Equipment Identity
  - Subscriber Identity Module (SIM)
    - Smart Card containing keys, identifiers and algorithms
    - Identifiers
      - K_i – Subscriber Authentication Key
      - IMSI – International Mobile Subscriber Identity
      - TMSI – Temporary Mobile Subscriber Identity
      - MSISDN – Mobile Subscriber Integrated Services Digital Network Number (the telephone number)
      - PIN – Personal Identity Number protecting a SIM
      - LAI – location area identity

GSM Architecture

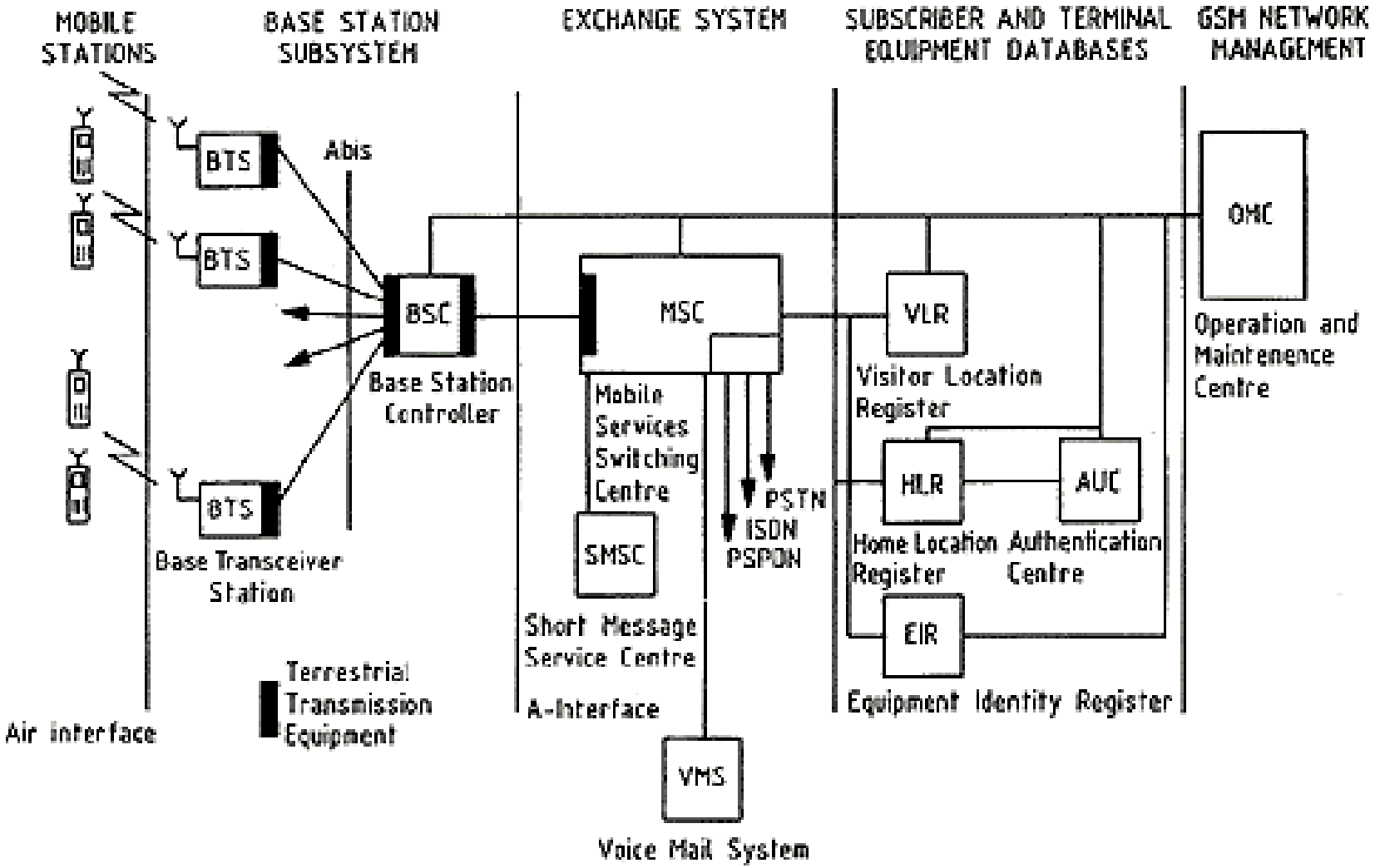


Fig. 1. GSM network architecture

Into the architecture

- Mobile phone is identified by SIM card.
- Key feature of the GSM
- Has the “secret” for authentication

Into the architecture(2)

- BTS – houses the radiotransceivers of the cell and handles the radio-link protocols with the mobile
- BSC – manages radio resources (channel setup, handover) for one or more BTSs

Into the architecture(3)

- MSC – Mobile Switching Center
- The central component of the network
- Like a telephony switch plus everything for a mobile subscriber: registration, authentication, handovers, call routing, connection to fixed networks.
- Each switch handles dozens of cells

Into the architecture(4)

- HLR – database of all users + current location. One per network
- VLR – database of users + roamers in some geographic area. Caches the HLR
- EIR – database of valid equipment
- AUC – Database of users' secret keys

Subscriber Identity Protection

- TMSI – Temporary Mobile Subscriber Identity
  - Goals
    - TMSI is used instead of IMSI as a temporary subscriber identifier
    - TMSI prevents an eavesdropper from identifying of subscriber
  - Usage
    - TMSI is assigned when IMSI is transmitted to AUC on the first phone switch on
    - Every time a location update (new MSC) occurs the network assigns a new TMSI
    - TMSI is used by the MS to report to the network or during a call initialization
    - Network uses TMSI to communicate with MS
    - On MS switch off TMSI is stored on SIM card to be reused next time
  - The Visitor Location Register (VLR) performs assignment, administration and update of the TMSI

Key Management Scheme

- $\mathbf{K_i}$ – Subscriber Authentication Key
  - Shared 128 bit key used for authentication of subscriber by the operator
  - Key Storage
    - Subscriber's SIM (owned by operator, i.e. trusted)
    - Operator's Authentication Centre (AUC) of the subscriber's home network
- SIM can be used with different equipment

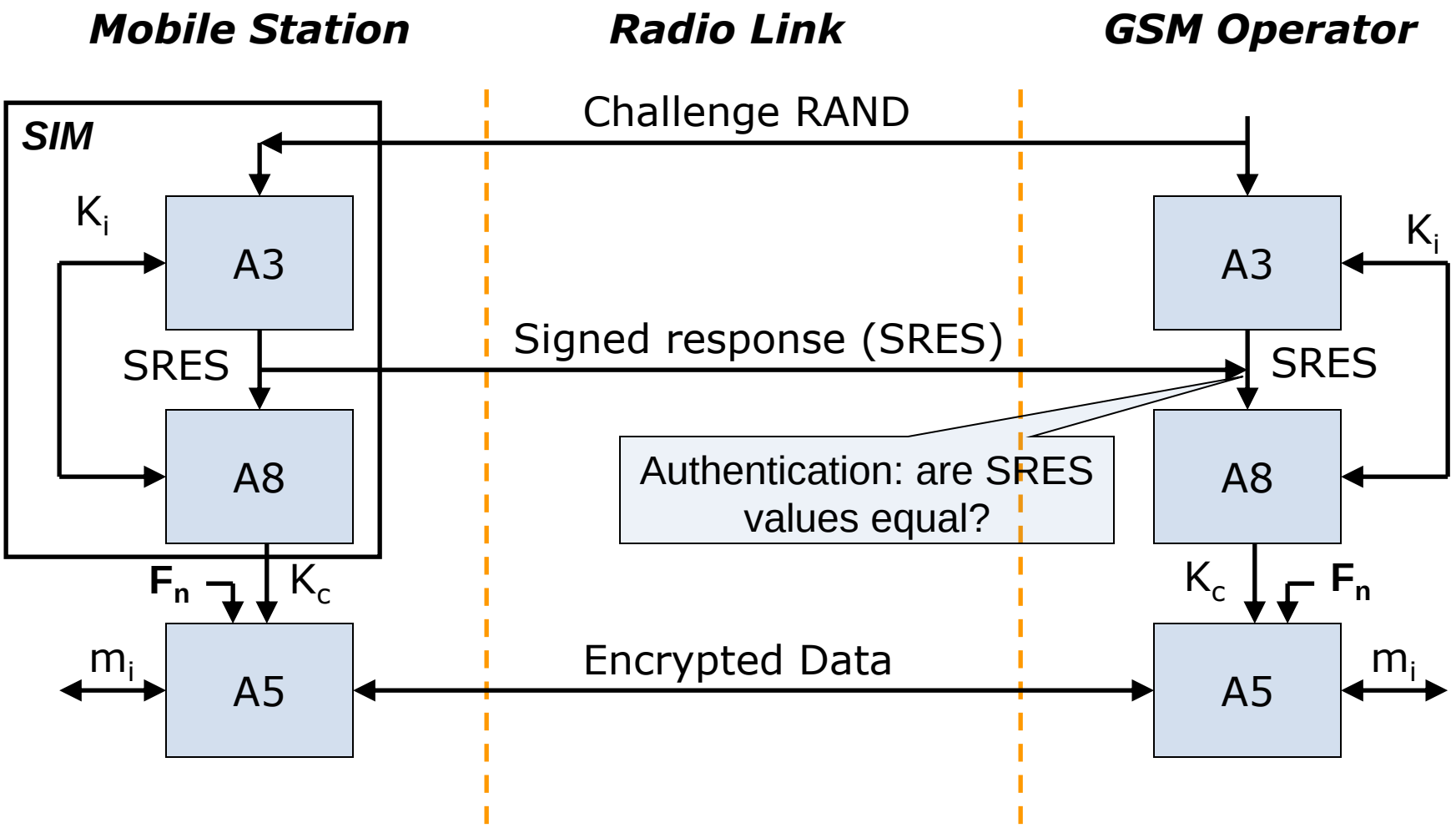
Detection of Compromised Equipment

- International Mobile Equipment Identifier (IMEI)
  - Identifier allowing to identify mobiles
  - IMEI is independent of SIM
  - Used to identify stolen or compromised equipment (*#06#)
- Equipment Identity Register (EIR)
  - Black list – stolen or non-type mobiles
  - White list - valid mobiles
  - Gray list – local tracking mobiles
- Central Equipment Identity Register (CEIR)
  - Approved mobile type (type approval authorities)
  - Consolidated black list (posted by operators)

Authentication

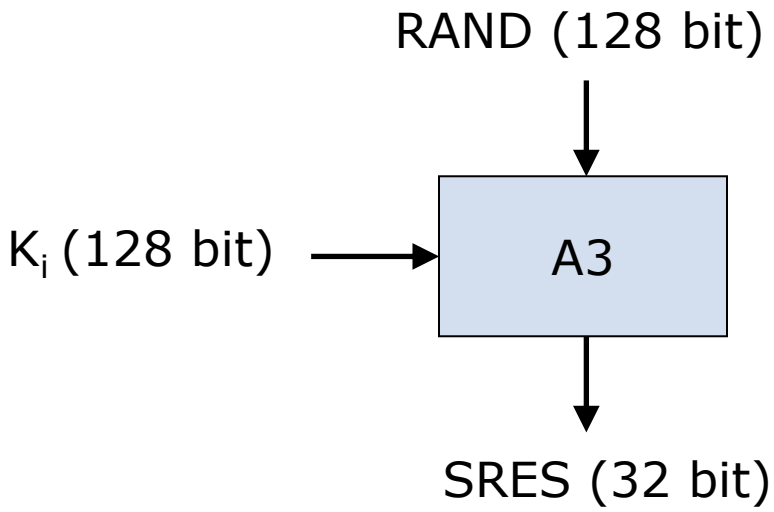
- Authentication Goals
  - Subscriber (SIM holder) authentication
  - Protection of the network against unauthorized use
  - Create a session key
- Authentication Scheme
  - Subscriber identification: IMSI or TMSI
  - Challenge-Response authentication of the subscriber by the operator

Authentication and Encryption Scheme



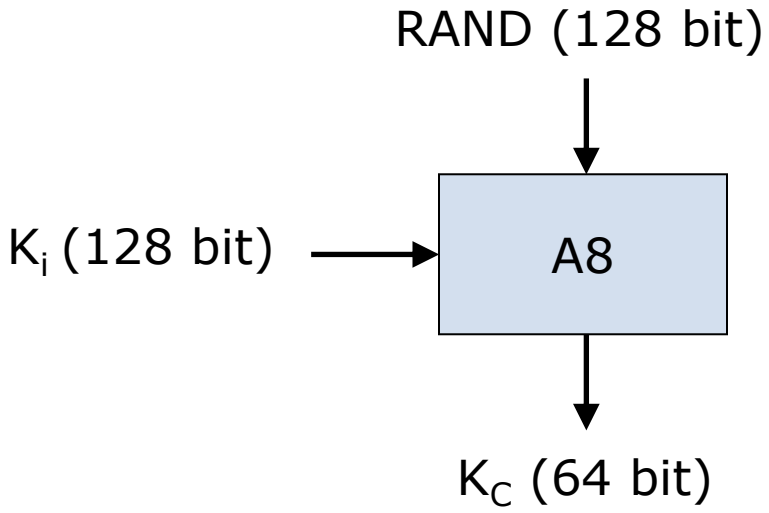
A3 – MS Authentication Algorithm

- Goal
  - Generation of SRES response to MSC's random challenge RAND



A8 – Voice Privacy Key Generation Algorithm

- Goal
  - Generation of session key K_s
    - A8 specification was never made public

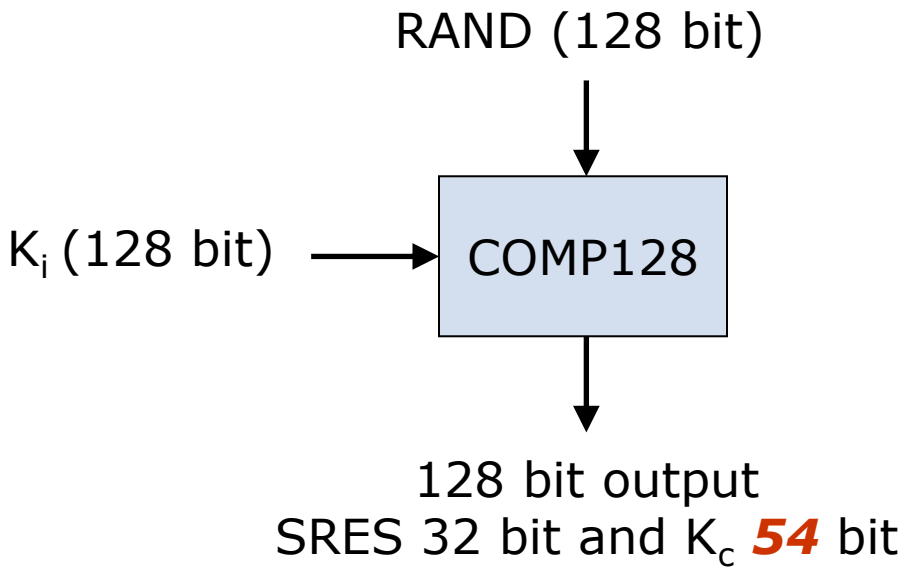


Logical Implementation of A3 and A8

- Both A3 and A8 algorithms are implemented on the SIM
  - Algorithm implementation is independent of hardware manufacturers and network operators.

Logical Implementation of A3 and A8

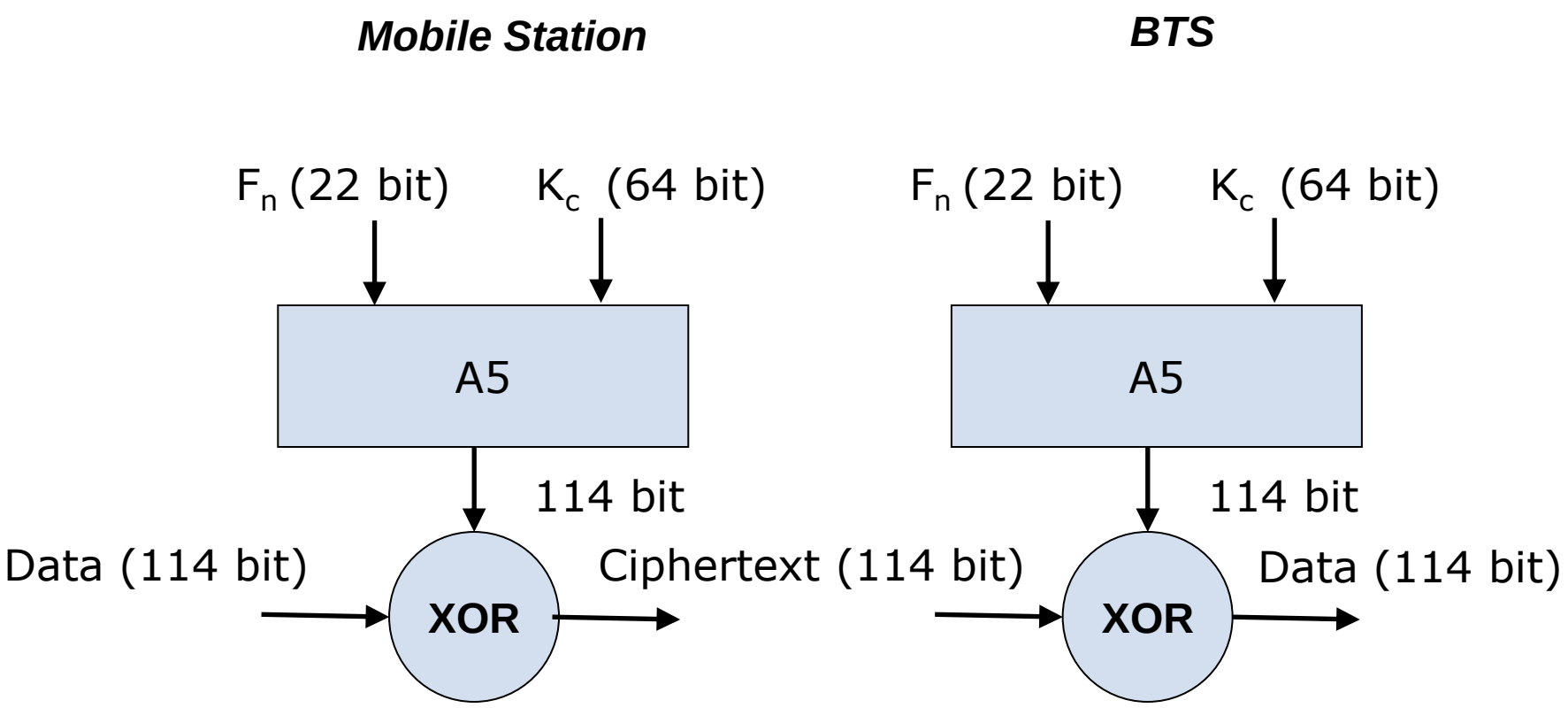
- COMP128 is used for both A3 and A8 in most GSM networks.
  - COMP128 is a keyed hash function



A5 – Encryption Algorithm

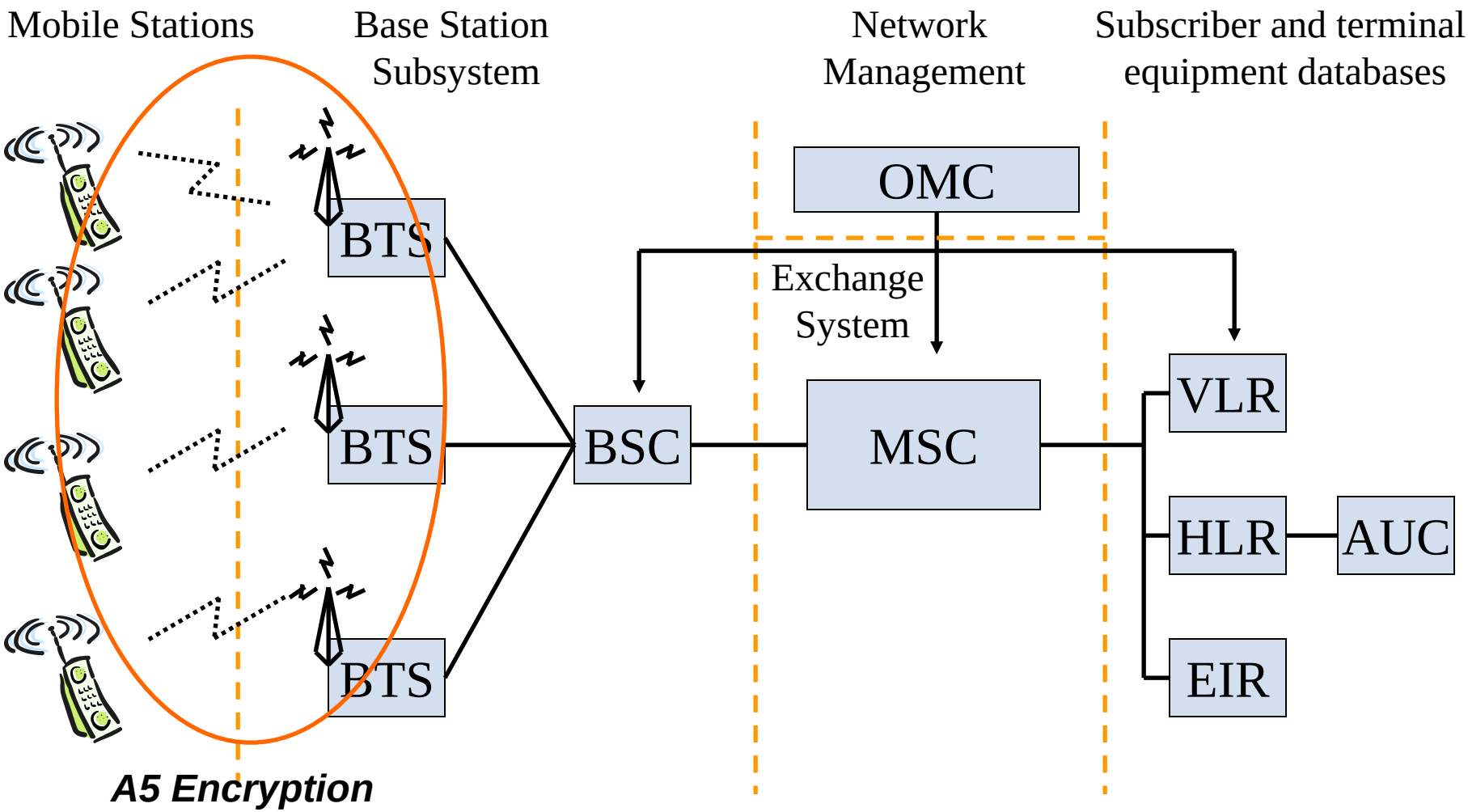
- A5 is a stream cipher
  - Implemented very efficiently on hardware
  - Design was never made public
  - Leaked to Ross Anderson and Bruce Schneier
- Variants
  - A5/1 – the strong version
  - A5/2 – the weak version
  - A5/3
    - GSM Association Security Group and 3GPP design
    - Based on Kasumi algorithm used in 3G mobile systems

Logical A5 Implementation



Real A5 output is 228 bit for both directions

A5 Encryption



A5/1: Operation

- A5/1 is a stream cipher, which is initialized all over again for every frame sent.
- Consists of 3 LFSRs of 19,22,23 bits length.
- The 3 registers are clocked in a stop/go fashion using the majority rule.

A5/1: Operation

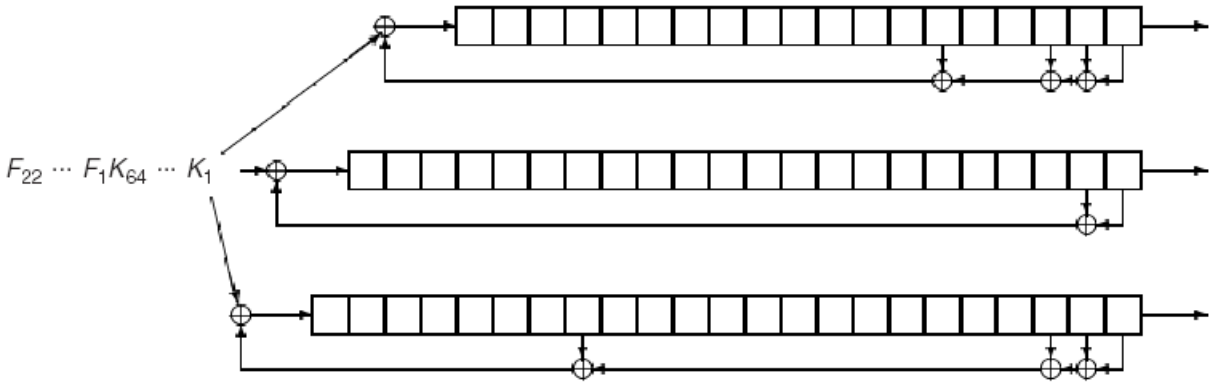


Fig. 1. Initialization of the A5/1 running-key generator

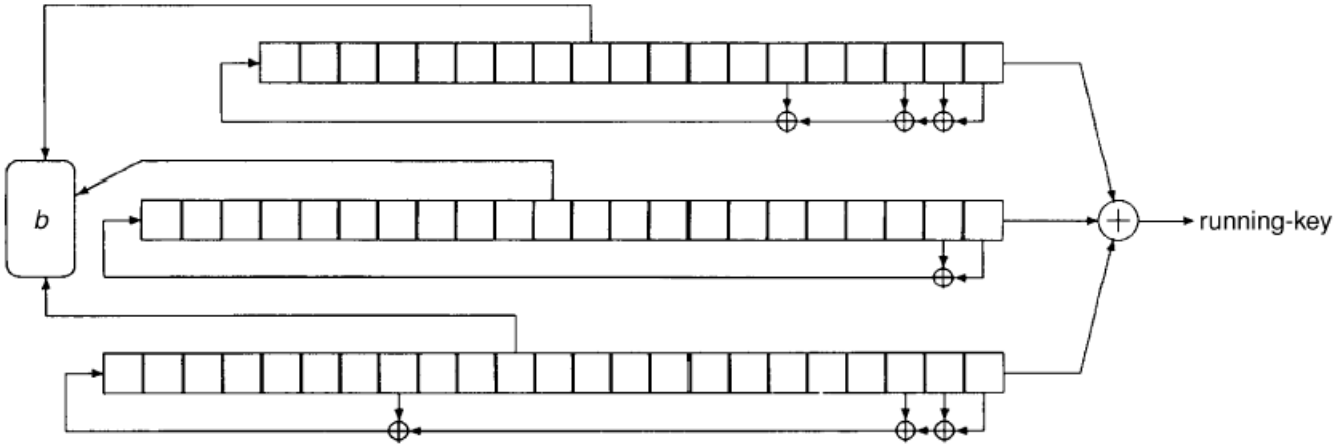
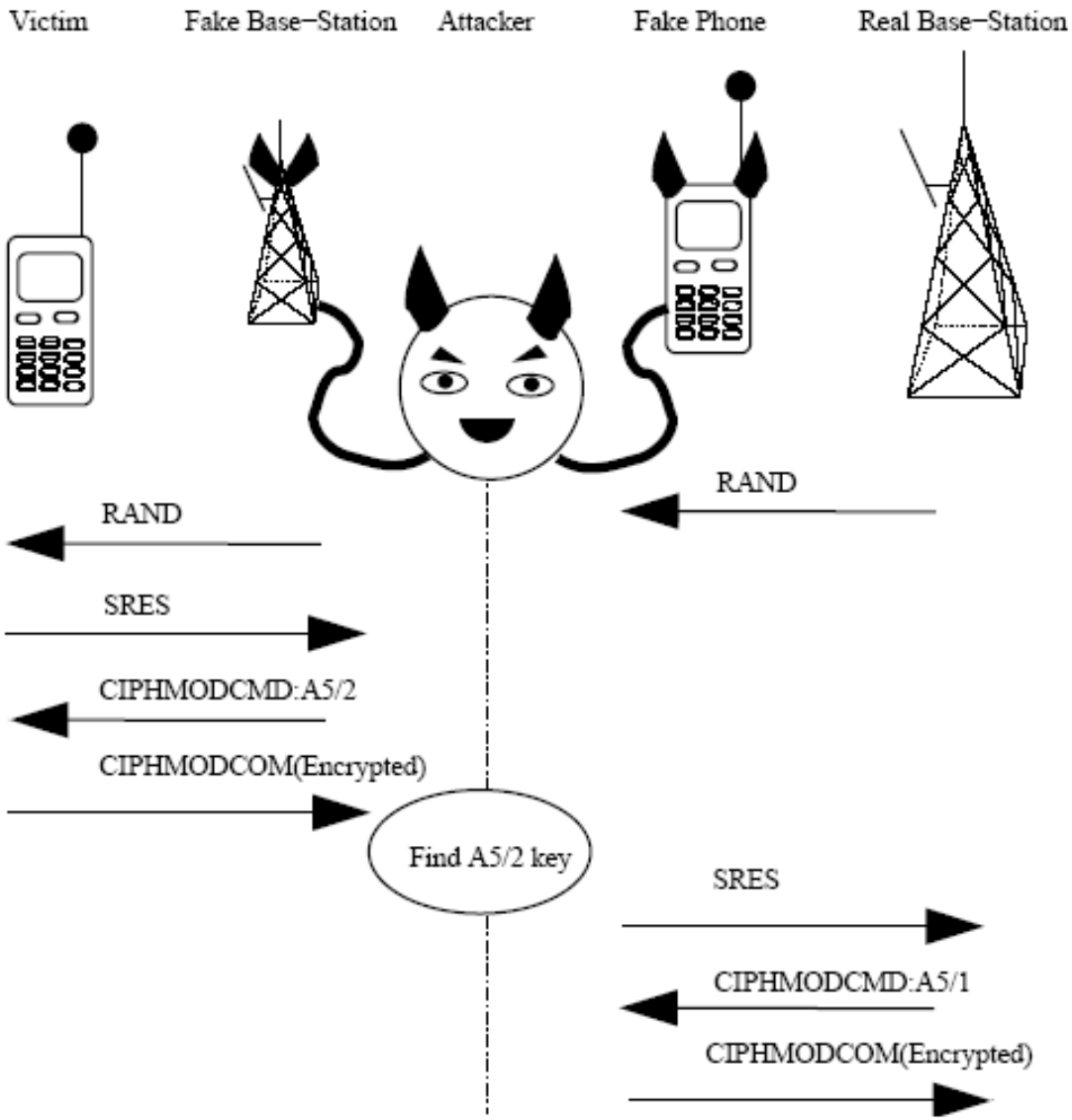


Fig. 2. A5/1 running-key generator

A5/1: Operation

- All 3 registers are zeroed
- 64 cycles (without the stop/go clock) :
  - Each bit of K (lsb to msb) is XOR'ed in parallel into the lsb's of the registers
- 22 cycles (without the stop/go clock) :
  - Each bit of F_n (lsb to msb) is XOR'ed in parallel into the lsb's of the registers
- 100 cycles with the stop/go clock control, discarding the output
- 228 cycles with the stop/go clock control which produce the output bit sequence.

Man-in-the-Middle Attack



Kapcsolódó termékek

http://www.toplinkpac.com/ibis.html

http://www.toplinkpac.com/gtres.html

http://www.toplinkpac.com/3g-cat.html

https://phantom-technologies.com/cpi1890-gsm-interception-system/

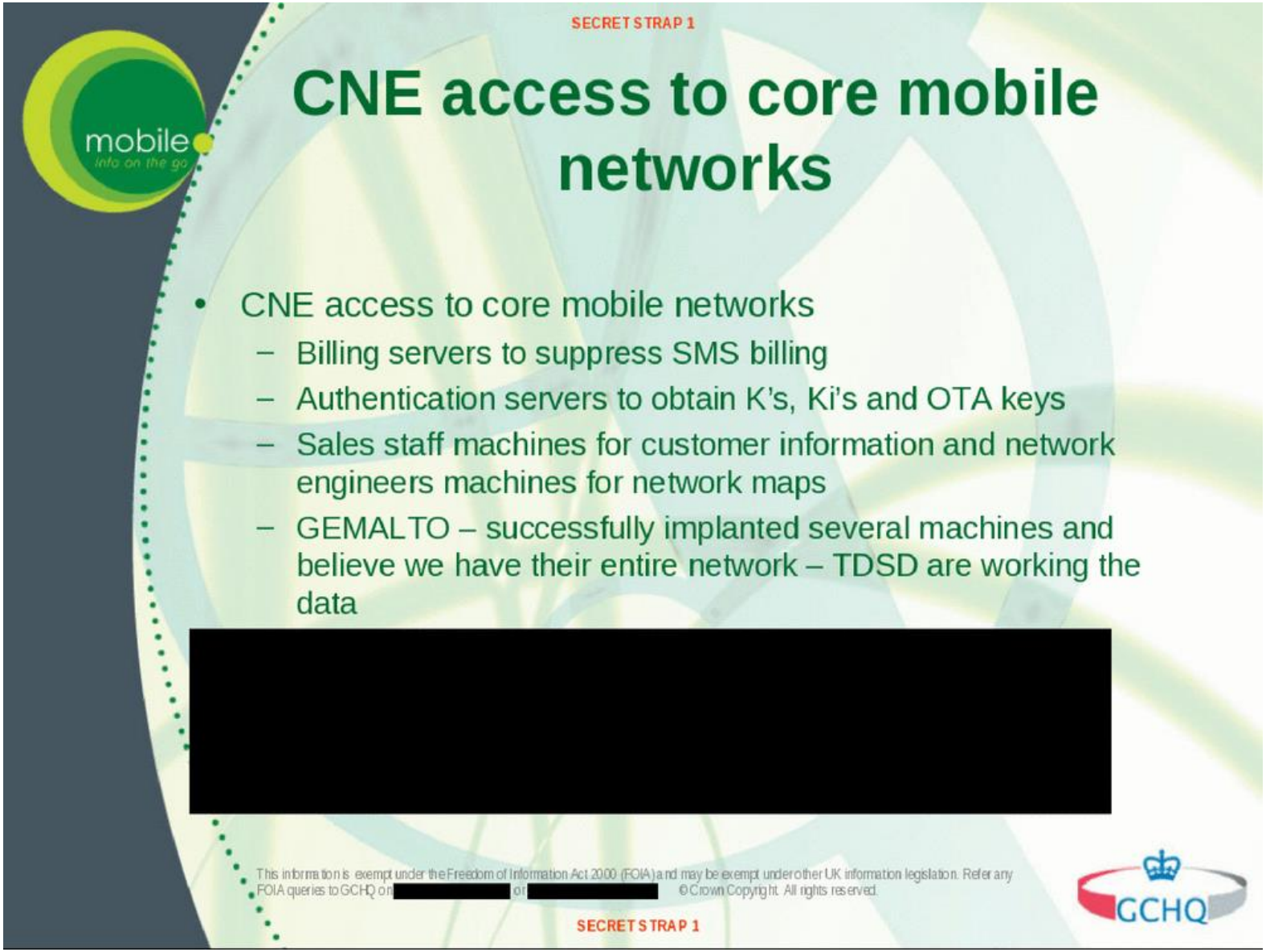
https://phantom-technologies.com/imsi400-imsi-catcher/

SIM klónozás

https://www.mobiledit.com/sim-cloning

http://www.tech2hack.com/how-to-clone-sim-card-easily/

Snowden


SECRET STRAP 1
mobile
info on the go
CNE access to core mobile networks
• CNE access to core mobile networks
– Billing servers to suppress SMS billing
– Authentication servers to obtain K's, Ki's and OTA keys
– Sales staff machines for customer information and network engineers machines for network maps
– GEMALTO – successfully implanted several machines and believe we have their entire network – TDSD are working the data
[redacted]
This information is exempt under the Freedom of Information Act 2000 (FOIA) and may be exempt under other UK information legislation. Refer any FOIA queries to GCHQ on [redacted] or [redacted]
© Crown Copyright. All rights reserved.
GCHQ
SECRET STRAP 1

Nyilvános kulcsú rejtjelezés

PPKE, ITK
Csapodi Márton

New directions in Cryptography

Whitfield Diffie & Martin E. Hellman, New directions in Cryptography, IEEE Transactions on Information Theory (1976)

I. Introduction

WE STAND TODAY on the brink of a revolution in cryptography. The development of cheap digital hardware has freed it from the design limitations of mechanical computing and brought the cost of high grade cryptographic devices down to where they can be used in such commercial applications as remote cash dispensers and computer terminals. In turn, such applications create a need for new types of cryptographic systems which minimize the necessity of secure key distribution channels and supply the equivalent of a written signature. At the same time, theoretical developments in information theory and computer science show promise of providing provably secure cryptosystems, changing this ancient art into a science.

New directions in Cryptography

The best known cryptographic problem is that of privacy: preventing the unauthorized extraction of information from communications over an insecure channel. In order to use cryptography to insure privacy, however, it is currently necessary for the communicating parties to share a key which is known to no one else. This is done by sending the key in advance over some secure channel such as private courier or registered mail. A private conversation between two people with no prior acquaintance is a common occurrence in business, however, and it is unrealistic to expect initial business contacts to be postponed long enough for keys to be transmitted by some physical means. The cost and delay imposed by this key distribution problem is a major barrier to the transfer of business communications to large teleprocessing networks.

Section III proposes two approaches to transmitting keying information over public (i.e., insecure) channels

Public key distribution systems *public key cryptosystem*

New directions in Cryptography

A second problem, amenable to cryptographic solution, which stands in the way of replacing contemporary business communications by teleprocessing systems is authentication. In current business, the validity of contracts is guaranteed by signatures. A signed contract serves as legal evidence of an agreement which the holder can present in court if necessary. The use of signatures, however, requires the transmission and storage of written contracts. In order to have a purely digital replacement for this paper instrument, each user must be able to produce a message whose authenticity can be checked by anyone, but which could not have been produced by anyone else, even the recipient. Since only one person can originate messages but many people can receive messages, this can be viewed as a broadcast cipher. Current electronic authentication techniques cannot meet this need.

New directions in Cryptography

III. Public Key Cryptography

We propose that it is possible to develop systems of the type shown in Fig. 2, in which two parties communicating solely over a public channel and using only publicly known techniques can create a secure connection. We examine two approaches to this problem, called public key cryptosystems and public key distribution systems, respectively. The first are more powerful, lending themselves to the solution of the authentication problems treated in the next section, while the second are much closer to realization.

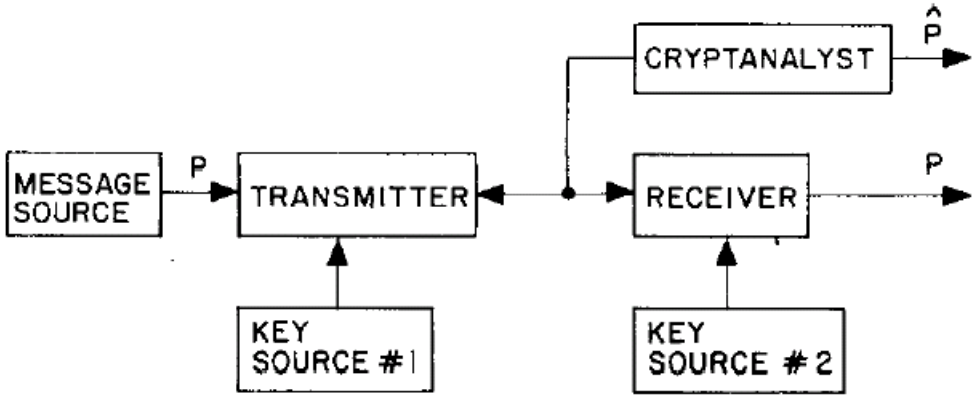


Fig. 2. Flow of information in public key system.

New directions in Cryptography

A *public key cryptosystem* is a pair of families $\{E_K\}_{K \in \{K\}}$ and $\{D_K\}_{K \in \{K\}}$ of algorithms representing invertible transformations,

$$E_K : \{M\} \rightarrow \{M\} \tag{2}$$

$$D_K : \{M\} \rightarrow \{M\} \tag{3}$$

on a finite message space $\{M\}$, such that

1) for every $K \in \{K\}$, E_K is the inverse of D_K,
2) for every $K \in \{K\}$ and $M \in \{M\}$, the algorithms E_K and D_K are easy to compute,
3) for almost every $K \in \{K\}$, each easily computed algorithm equivalent to D_K is computationally infeasible to derive from E_K,
4) for every $K \in \{K\}$, it is feasible to compute inverse pairs E_K and D_K from K.

New directions in Cryptography

A suggestive, although unfortunately useless, example of a public key cryptosystem is to encipher the plaintext, represented as a binary n-vector $\boldsymbol{m}$, by multiplying it by an invertible binary $n \times n$ matrix E. The cryptogram thus equals $E\boldsymbol{m}$. Letting $D = E^{-1}$ we have $\boldsymbol{m} = D\boldsymbol{c}$. Thus, both enciphering and deciphering require about n^2 operations. Calculation of D from E, however, involves a matrix inversion which is a harder problem. And it is at least conceptually simpler to obtain an arbitrary pair of inverse matrices than it is to invert a given matrix. Start with the identity matrix I and do elementary row and column operations to obtain an arbitrary invertible matrix E. Then starting with I do the inverses of these same elementary operations in reverse order to obtain $D = E^{-1}$. The sequence of elementary operations could be easily determined from a random bit string.

Unfortunately, matrix inversion takes only about n^3 operations. The ratio of "cryptanalytic" time (i.e., computing D from E) to enciphering or deciphering time is thus at most n, and enormous block sizes would be required to obtain ratios of 10^6 or greater. Also, it does not appear that knowledge of the elementary operations used to obtain E from I greatly reduces the time for computing D. And, since there is no round-off error in binary arithmetic, numerical stability is unimportant in the matrix inversion. In spite of its lack of practical utility, this matrix example is still useful for clarifying the relationships necessary in a public key cryptosystem.

A more practical approach to finding a pair of easily computed inverse algorithms E and D; such that D is hard to infer from E, makes use of the difficulty of analyzing programs in low level languages. Anyone who has tried to determine what operation is accomplished by someone else's machine language program knows that E itself (i.e., what E does) can be hard to infer from an algorithm for E. If the program were to be made purposefully confusing through addition of unneeded variables and statements, then determining an inverse algorithm could be made very difficult. Of course, E must be complicated enough to prevent its identification from input–output pairs.

Essentially what is required is a one-way compiler: one which takes an easily understood program written in a high level language and translates it into an incomprehensible program in some machine language. The compiler is one-way because it must be feasible to do the compilation, but infeasible to reverse the process. Since efficiency in size of program and run time are not crucial in this application, such compilers may be possible if the structure of the machine language can be optimized to assist in the confusion.

New directions in Cryptography

We now suggest a new public key distribution system

The new technique makes use of the apparent difficulty of computing logarithms over a finite field $GF(q)$ with a prime number q of elements. Let

$$Y = \alpha^X \bmod q, \qquad \text{for } 1 \leq X \leq q-1, \tag{4}$$

where α is a fixed primitive element of $GF(q)$, then X is referred to as the logarithm of Y to the base α, mod q:

$$X = \log_\alpha Y \bmod q, \qquad \text{for } 1 \leq Y \leq q-1. \tag{5}$$

Calculation of Y from X is easy, taking at most $2 \times \log_2 q$ multiplications [6, pp. 398–422]. For example, for $X = 18$,

$$Y = \alpha^{18} = (((\alpha^2)^2)^2)^2 \times \alpha^2. \tag{6}$$

Computing X from Y, on the other hand can be much more difficult and, for certain carefully chosen values of q, requires on the order of $q^{1/2}$ operations, using the best known algorithm [7, pp. 9, 575–576], [8].

New directions in Cryptography

Each user generates an independent random number X_i chosen uniformly from the set of integers $\{1,2,\cdots,q-1\}$. Each keeps X_i secret, but places

$$Y_i = \alpha^{X_i} \bmod q \tag{7}$$

in a public file with his name and address. When users i and j wish to communicate privately, they use

$$K_{ij} = \alpha^{X_i X_j} \bmod q \tag{8}$$

as their key. User i obtains K_{ij} by obtaining Y_j from the public file and letting

$$K_{ij} = Y_j^{X_i} \bmod q \tag{9}$$

$$= (\alpha^{X_j})^{X_i} \bmod q \tag{10}$$

$$= \alpha^{X_j X_i} = \alpha^{X_i X_j} \bmod q. \tag{11}$$

User j obtains K_{ij} in the similar fashion

$$K_{ij} = Y_i^{X_j} \bmod q. \tag{12}$$

Another user must compute K_{ij} from Y_i and Y_j, for example, by computing

$$K_{ij} = Y_i^{(\log_\alpha Y_j)} \bmod q. \tag{13}$$

We thus see that if logs mod q are easily computed the system can be broken. While we do not currently have a proof of the converse (i.e., that the system is secure if logs mod q are difficult to compute), neither do we see any way to compute K_{ij} from Y_i and Y_j without first obtaining either X_i or X_j.

New directions in Cryptography

IV. One-Way Authentication

The problem of authentication is perhaps an even more serious barrier to the universal adoption of telecommunications for business transactions than the problem of key distribution. Authentication is at the heart of any system involving contracts and billing. Without it, business cannot function. Current electronic authentication systems cannot meet the need for a purely digital, unforgeable, message dependent signature. They provide protection against third party forgeries, but do not protect against disputes between transmitter and receiver.

In order to develop a system capable of replacing the current written contract with some purely electronic form of communication, we must discover a digital phenomenon with the same properties as a written signature. It must be easy for anyone to recognize the signature as authentic, but impossible for anyone other than the legitimate signer to produce it. We will call any such technique *one-way authentication*. Since any digital signal can be copied precisely, a true digital signature must be recognizable without being known.

New directions in Cryptography

Consider the "login" problem in a multiuser computer system. When setting up his account, the user chooses a password which is entered into the system's password directory. Each time he logs in, the user is again asked to provide his password. By keeping this password secret from all other users, forged logins are prevented. This, however, makes it vital to preserve the security of the password directory since the information it contains would allow perfect impersonation of any user. The problem is further compounded if system operators have legitimate reasons for accessing the directory. Allowing such legitimate accesses, but preventing all others, is next to impossible.

New directions in Cryptography

This leads to the apparently impossible requirement for a new login procedure capable of judging the authenticity of passwords without actually knowing them. While appearing to be a logical impossibility, this proposal is easily satisfied. When the user first enters his password PW, the computer automatically and transparently computes a function $f(PW)$ and stores this, not PW, in the password directory. At each successive login, the computer calculates $f(X)$, where X is the proffered password, and compares $f(X)$ with the stored value $f(PW)$. If and only if they are equal, the user is accepted as being authentic. Since the function f must be calculated once per login, its computation time must be small. A million instructions (costing approximately \$0.10 at bicentennial prices) seems to be a reasonable limit on this computation. If we could ensure, however, that calculation of f^{-1} required 10^{30} or more instructions, someone who had subverted the system to obtain the password directory could not in practice obtain PW from $f(PW)$, and could thus not perform an unauthorized login. Note that $f(PW)$ is not accepted as a password by the login program since it will automatically compute $f(f(PW))$ which will not match the entry $f(PW)$ in the password directory.

New directions in Cryptography

We assume that the function f is public information, so that it is not ignorance of f which makes calculation of f^{-1} difficult. Such functions are called one-way functions and were first employed for use in login procedures by R. M. Needham [9, p. 91]. They are also discussed in two recent papers [10], [11] which suggest interesting approaches to the design of one-way functions.

More precisely, a function f is a *one-way function* if, for any argument x in the domain of f, it is easy to compute the corresponding value $f(x)$, yet, for almost all y in the range of f, it is computationally infeasible to solve the equation $y = f(x)$ for any suitable argument x.

A public key cryptosystem can be used to produce a true one-way authentication system as follows. If user A wishes to send a message M to user B, he "deciphers" it in his secret deciphering key and sends $D_A(M)$. When user B receives it, he can read it, and be assured of its authenticity by "enciphering" it with user A's public enciphering key E_A. B also saves $D_A(M)$ as proof that the message came from A. Anyone can check this claim by operating on $D_A(M)$ with the publicly known operation E_A to recover M. Since only A could have generated a message with this property, the solution to the one-way authentication problem would follow immediately from the development of public key cryptosystems.

A method for Obtaining Digital Signatures and Public-Key Cryptosystems

R. L. Rivest, A. Shamir & L. Adleman, Communications of the ACR (1977)

Abstract

An encryption method is presented with the novel property that publicly revealing an encryption key does not thereby reveal the corresponding decryption key. This has two important consequences:

1. Couriers or other secure means are not needed to transmit keys, since a message can be enciphered using an encryption key publicly revealed by the intended recipient. Only he can decipher the message, since only he knows the corresponding decryption key.
2. A message can be "signed" using a privately held decryption key. Anyone can verify this signature using the corresponding publicly revealed encryption key. Signatures cannot be forged, and a signer cannot later deny the validity of his signature. This has obvious applications in "electronic mail" and "electronic funds transfer" systems.

A message is encrypted by representing it as a number M, raising M to a publicly specified power e, and then taking the remainder when the result is divided by the publicly specified product, n, of two large secret prime numbers p and q. Decryption is similar; only a different, secret, power d is used, where $e \cdot d \equiv 1 \pmod{(p-1) \cdot (q-1)}$. The security of the system rests in part on the difficulty of factoring the published divisor, n.

A method for Obtaining Digital Signatures and Public-Key Cryptosystems

I Introduction

The era of "electronic mail" [10] may soon be upon us; we must ensure that two important properties of the current "paper mail" system are preserved: (a) messages are *private*, and (b) messages can be *signed*. We demonstrate in this paper how to build these capabilities into an electronic mail system.

At the heart of our proposal is a new encryption method. This method provides an implementation of a "public-key cryptosystem," an elegant concept invented by Diffie and Hellman [1]. Their article motivated our research, since they presented the concept but not any practical implementation of such a system. Readers familiar with [1] may wish to skip directly to Section V for a description of our method.

A method for Obtaining Digital Signatures and Public-Key Cryptosystems

V Our Encryption and Decryption Methods

To encrypt a message M with our method, using a public encryption key (e, n), proceed as follows. (Here e and n are a pair of positive integers.)

First, represent the message as an integer between 0 and $n-1$. (Break a long message into a series of blocks, and represent each block as such an integer.) Use any standard representation. The purpose here is not to encrypt the message but only to get it into the numeric form necessary for encryption.

Then, encrypt the message by raising it to the eth power modulo n. That is, the result (the ciphertext C) is the remainder when M^e is divided by n.

To decrypt the ciphertext, raise it to another power d, again modulo n. The encryption and decryption algorithms E and D are thus:

$$\begin{aligned} C \equiv E(M) &\equiv M^e \pmod{n}, \text{ for a message } M\ . \\ D(C) &\equiv C^d \pmod{n}, \text{ for a ciphertext } C\ . \end{aligned}$$

Note that encryption does not increase the size of a message; both the message and the ciphertext are integers in the range 0 to $n-1$.

The *encryption key* is thus the pair of positive integers (e, n). Similarly, the *decryption* key is the pair of positive integers (d, n). Each user makes his encryption key public, and keeps the corresponding decryption key private. (These integers should properly be subscripted as in n_A, e_A, and d_A, since each user has his own set. However, we will only consider a typical set, and will omit the subscripts.)

A method for Obtaining Digital Signatures and Public-Key Cryptosystems

How should you choose your encryption and decryption keys, if you want to use our method?

You first compute n as the product of two primes p and q:

$$n = p \cdot q \ .$$

These primes are very large, "random" primes. Although you will make n public, the factors p and q will be effectively hidden from everyone else due to the enormous difficulty of factoring n. This also hides the way d can be derived from e.

You then pick the integer d to be a large, random integer which is relatively prime to $(p-1) \cdot (q-1)$. That is, check that d satisfies:

$$\gcd(d, (p-1) \cdot (q-1)) = 1$$

The integer e is finally computed from p, q, and d to be the "multiplicative inverse" of d, modulo $(p-1) \cdot (q-1)$. Thus we have

$$e \cdot d \equiv 1 \pmod{(p-1) \cdot (q-1)}.$$

A method for Obtaining Digital Signatures and Public-Key Cryptosystems

We demonstrate the correctness of the deciphering algorithm using an identity due to Euler and Fermat [7]: for any integer (message) M which is relatively prime to n,

$$M^{\phi(n)} \equiv 1 \pmod{n} . \tag{3}$$

Here $\phi(n)$ is the Euler totient function giving number of positive integers less than n which are relatively prime to n. For prime numbers p,

$$\phi(p) = p - 1 .$$

In our case, we have by elementary properties of the totient function [7]:

$$\begin{aligned} \phi(n) &= \phi(p) \cdot \phi(q) \\ &= (p-1) \cdot (q-1) \\ &= n - (p+q) + 1 . \end{aligned} \tag{4}$$

Since d is relatively prime to $\phi(n)$, it has a multiplicative inverse e in the ring of integers modulo $\phi(n)$:

$$e \cdot d \equiv 1 \pmod{\phi(n)}. \tag{5}$$

$$M^{e \cdot d} \equiv M^{k \cdot \phi(n)+1} \equiv M \pmod{n}.$$

A method for Obtaining Digital Signatures and Public-Key Cryptosystems

B How to Find Large Prime Numbers

Each user must (privately) choose two large random numbers p and q to create his own encryption and decryption keys. These numbers must be large so that it is not computationally feasible for anyone to factor $n = p \cdot q$. (Remember that n, but not p or q, will be in the public file.) We recommend using 100-digit (decimal) prime numbers p and q, so that n has 200 digits.

To find a 100-digit "random" prime number, generate (odd) 100-digit random numbers until a prime number is found. By the prime number theorem [7], about $(\ln 10^{100})/2 = 115$ numbers will be tested before a prime is found.

To gain additional protection against sophisticated factoring algorithms, p and q should differ in length by a few digits, both $(p-1)$ and $(q-1)$ should contain large prime factors, and $\gcd(p-1, q-1)$ should be small. The latter condition is easily checked.

To find a prime number p such that $(p-1)$ has a large prime factor, generate a large random prime number u, then let p be the first prime in the sequence $i \cdot u + 1$, for $i = 2, 4, 6, \ldots$. (This shouldn't take too long.) Additional security is provided by ensuring that $(u-1)$ also has a large prime factor.

A method for Obtaining Digital Signatures and Public-Key Cryptosystems

C How to Choose d

It is very easy to choose a number d which is relatively prime to $\phi(n)$. For example, any prime number greater than $\max(p, q)$ will do. It is important that d should be chosen from a large enough set so that a cryptanalyst cannot find it by direct search.

D How to Compute e from d and $\phi(n)$

To compute e, use the following variation of Euclid's algorithm for computing the greatest common divisor of $\phi(n)$ and d. (See exercise 4.5.2.15 in [3].) Calculate $\gcd(\phi(n), d)$ by computing a series $x_0, x_1, x_2, \ldots$, where $x_0 \equiv \phi(n), x_1 = d$, and $x_{i+1} \equiv x_{i-1} \pmod{x_i}$, until an x_k equal to 0 is found. Then $\gcd(x_0, x_1) = x_{k-1}$. Compute for each x_i numbers a_i and b_i such that $x_i = a_i \cdot x_0 + b_i \cdot x_1$. If $x_{k-1} = 1$ then b_{k-1} is the multiplicative inverse of $x_1 \pmod{x_0}$. Since k will be less than $2\log_2(n)$, this computation is very rapid.

If e turns out to be less than $\log_2(n)$, start over by choosing another value of d. This guarantees that every encrypted message (except $M = 0$ or $M = 1$) undergoes some "wrap-around" (reduction modulo n) .

A method for Obtaining Digital Signatures and Public-Key Cryptosystems

VIII A Small Example

Consider the case $p = 47, q = 59, n = p \cdot q = 47 \cdot 59 = 2773$, and $d = 157$. Then $\phi(2773) = 46 \cdot 58 = 2668$, and e can be computed as follows:

$x_0 = 2668, \quad a_0 = 1, \quad b_0 = 0,$
$x_1 = 157, \quad a_1 = 0, \quad b_1 = 1,$
$x_2 = 156, \quad a_2 = 1, \quad b_2 = -16$ (since $2668 = 157 \cdot 16 + 156$) ,
$x_3 = 1, \quad a_3 = -1, \quad b_3 = 17$ (since $157 = 1 \cdot 156 + 1$) .

Therefore $e = 17$, the multiplicative inverse $\pmod{2668}$ of $d = 157$.

With $n = 2773$ we can encode two letters per block, substituting a two-digit number for each letter: blank = 00, A = 01, B = 02, ..., Z = 26. Thus the message

```
ITS ALL GREEK TO ME
```

(Julius Caesar, I, ii, 288, paraphrased) is encoded:

```
0920 1900 0112 1200 0718 0505 1100 2015 0013 0500
```

Since $e = 10001$ in binary, the first block ($M = 920$) is enciphered:

$$M^{17} = (((((1)^2 \cdot M)^2)^2)^2)^2 \cdot M = 948 \pmod{2773} .$$

The whole message is enciphered as:

```
0948 2342 1084 1444 2663 2390 0778 0774 0219 1655 .
```

The reader can check that deciphering works: $948^{157} \equiv 920 \pmod{2773}$, etc.

A method for Obtaining Digital Signatures and Public-Key Cryptosystems

We show in the next sections that all the obvious approaches for breaking our system are at least as difficult as factoring n. While factoring large numbers is not provably difficult, it is a well-known problem that has been worked on for the last three hundred years by many famous mathematicians. Fermat (1601?-1665) and Legendre (1752-1833) developed factoring algorithms; some of today's more efficient algorithms are based on the work of Legendre. As we shall see in the next section, however, no one has yet found an algorithm which can factor a 200-digit number in a reasonable amount of time. We conclude that our system has already been partially "certified" by these previous efforts to find efficient factoring algorithms.

In the following sections we consider ways a cryptanalyst might try to determine the secret decryption key from the publicly revealed encryption key. We do not consider ways of protecting the decryption key from theft; the usual physical security methods should suffice. (For example, the encryption device could be a separate device which could also be used to generate the encryption and decryption keys, such that the decryption key is never printed out (even for its owner) but only used to decrypt messages. The device could erase the decryption key if it was tampered with.)

A method for Obtaining Digital Signatures and Public-Key Cryptosystems

A Factoring n

Factoring n would enable an enemy cryptanalyst to "break" our method. The factors of n enable him to compute $\phi(n)$ and thus d. Fortunately, factoring a number seems to be much more difficult than determining whether it is prime or composite.

Table 1 gives the number of operations needed to factor n with Schroeppel's method, and the time required if each operation uses one microsecond, for various lengths of the number n (in decimal digits).

Table 1

Digits	*Number of operations*	*Time*
50	1.4×10^{10}	3.9 hours
75	9.0×10^{12}	104 days
100	2.3×10^{15}	74 years
200	1.2×10^{23}	3.8×10^{9} years
300	1.5×10^{29}	4.9×10^{15} years
500	1.3×10^{39}	4.2×10^{25} years

A method for Obtaining Digital Signatures and Public-Key Cryptosystems

B Computing $\phi(n)$ Without Factoring n

If a cryptanalyst could compute $\phi(n)$ then he could break the system by computing d as the multiplicative inverse of e modulo $\phi(n)$ (using the procedure of Section VII D).

We argue that this approach is no easier than factoring n since it enables the cryptanalyst to easily factor n using $\phi(n)$. This approach to factoring n has not turned out to be practical.

How can n be factored using $\phi(n)$? First, $(p+q)$ is obtained from n and $\phi(n) = n-(p+q)+1$. Then $(p-q)$ is the square root of $(p+q)^2-4n$. Finally, q is half the difference of $(p+q)$ and $(p-q)$.

Therefore breaking our system by computing $\phi(n)$ is no easier than breaking our system by factoring n. (This is why n must be composite; $\phi(n)$ is trivial to compute if n is prime.)

A method for Obtaining Digital Signatures and Public-Key Cryptosystems

C Determining d Without Factoring n or Computing $\phi(n)$.

Of course, d should be chosen from a large enough set so that a direct search for it is unfeasible.

We argue that computing d is no easier for a cryptanalyst than factoring n, since once d is known n could be factored easily. This approach to factoring has also not turned out to be fruitful.

A knowledge of d enables n to be factored as follows. Once a cryptanalyst knows d he can calculate $e \cdot d - 1$, which is a multiple of $\phi(n)$. Miller [6] has shown that n can be factored using any multiple of $\phi(n)$. Therefore if n is large a cryptanalyst should not be able to determine d any easier than he can factor n.

A cryptanalyst may hope to find a d' which is equivalent to the d secretly held by a user of the public-key cryptosystem. If such values d' were common then a brute-force search could break the system. However, all such d' differ by the least common multiple of $(p-1)$ and $(q-1)$, and finding one enables n to be factored. (In (3) and (5), $\phi(n)$ can be replaced by $\text{lcm}(p-1, q-1)$.) Finding any such d' is therefore as difficult as factoring n.

A method for Obtaining Digital Signatures and Public-Key Cryptosystems

D Computing D in Some Other Way

Although this problem of "computing e-th roots modulo n without factoring n" is not a well-known difficult problem like factoring, we feel reasonably confident that it is computationally intractable. It may be possible to prove that any general method of breaking our scheme yields an efficient factoring algorithm. This would establish that any way of breaking our scheme must be as difficult as factoring. We have not been able to prove this conjecture, however.

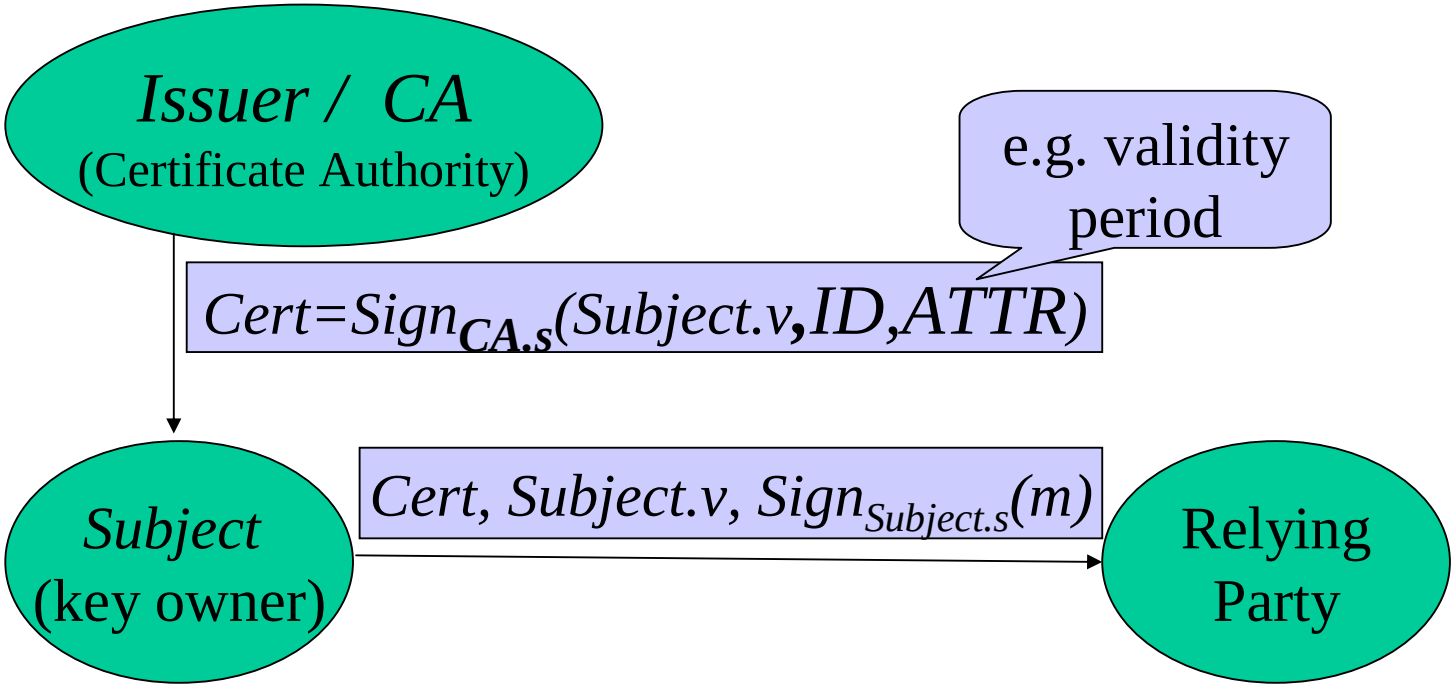
Our method should be certified by having the above conjecture of intractability withstand a concerted attempt to disprove it. The reader is challenged to find a way to "break" our method.

PKI

PPKE, ITK
Csapodi Márton

Public Key Certificates & Authorities

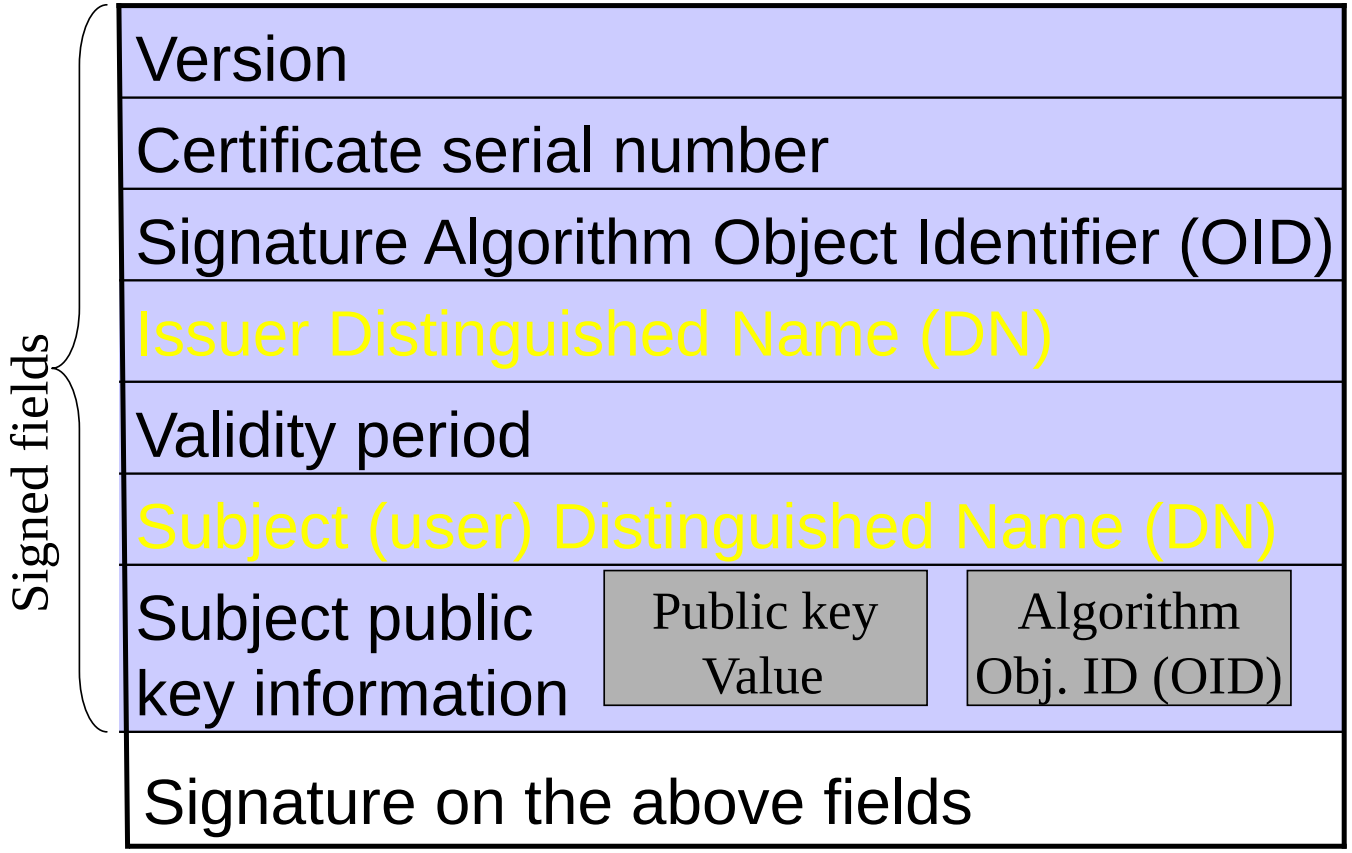
- *Certificate:* signature by Certificate Authority (CA) over subject's public key and attributes
- Attributes:
  - Validated by CA (liability?)
  - Used by **relying party** for decisions (e.g., use this website?)
  - **Questions:** Attributes? Identifiers? Format? …



X.509 public key certificates

- Public key **signed** by (trusted) **issuer (CA)**
  - Certificate: signed public key (and attributes)
  - CA: Certificate Authority (issuer)
- **X.509:** ITU's standard for certificates & usage
  - Widely adopted – in spite of complexity
- Main outcome of X.500 standard
  - ITU: International Telcos Union
  - Goal: trusted, centralized 'phone directory'
  - Global directory? No; but – X.509 widely used
    - Why global directory failed? Too complex, revealing
  - Identifiers: distinguished names
    - Goals: unique, meaningful, decentralized identifiers

Original (V1) X.509 Certs



Object Identifiers (OID):

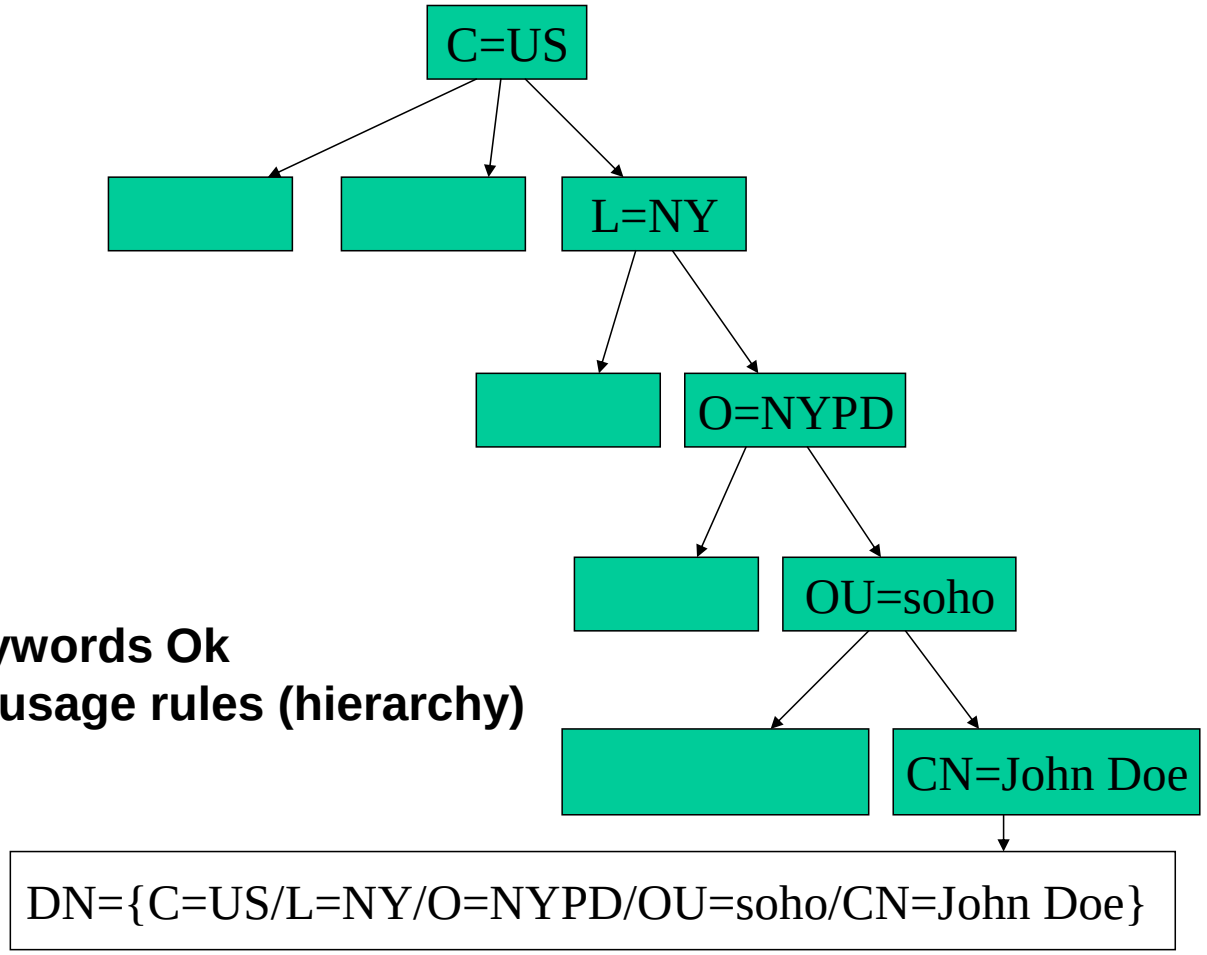
- Global, unique identifiers
- Sequence of numbers, e.g.: 1.16.840.1.45.33
  - Hierarchical

X.509 Distinguished Names (DN)

- Goal: meaningful, unique and decentralized identifiers
- Sequence of keywords, a string value for each of them
- Distributed directory, responsibility →*hierarchical DN*

Keyword	Meaning
C	Country
L	Locality name
O	Organization name
OU	Organization Unit name
CN	Common Name

Distinguished Name (DN) Hierarchy

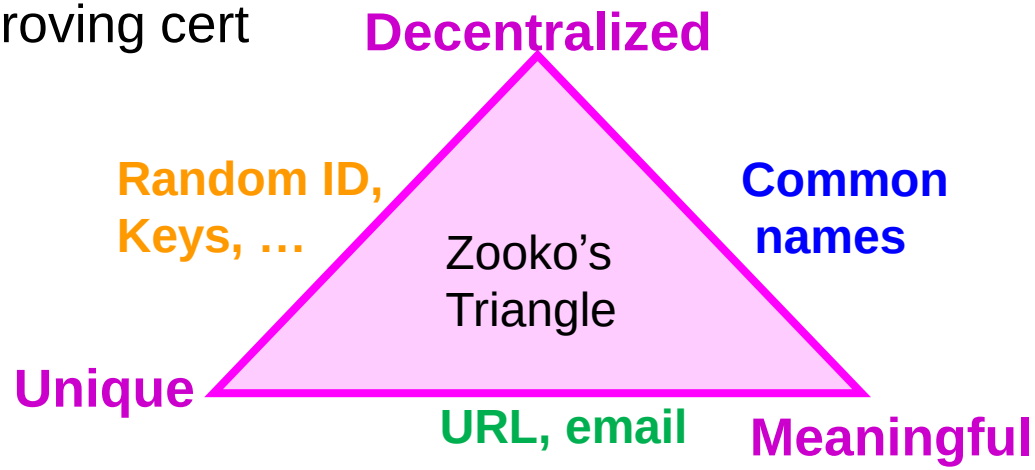


Comments:

1. **Other keywords Ok**
2. **No strict usage rules (hierarchy)**

Goals for Identifiers in Certificates

- Meaningful (to humans)
  - Memorable, reputation, off-net, legal
- Unique identification of entity (owner)
- Decentralized - with Accountability: assigned by any trusted certificate authority
  - Accountability: CA approving cert
- Pairs are easy:
  - Unique + Meaningful
  - Meaningful + Decentalized
  - Unique + Decentralized



- Zooko: can't have all three properties

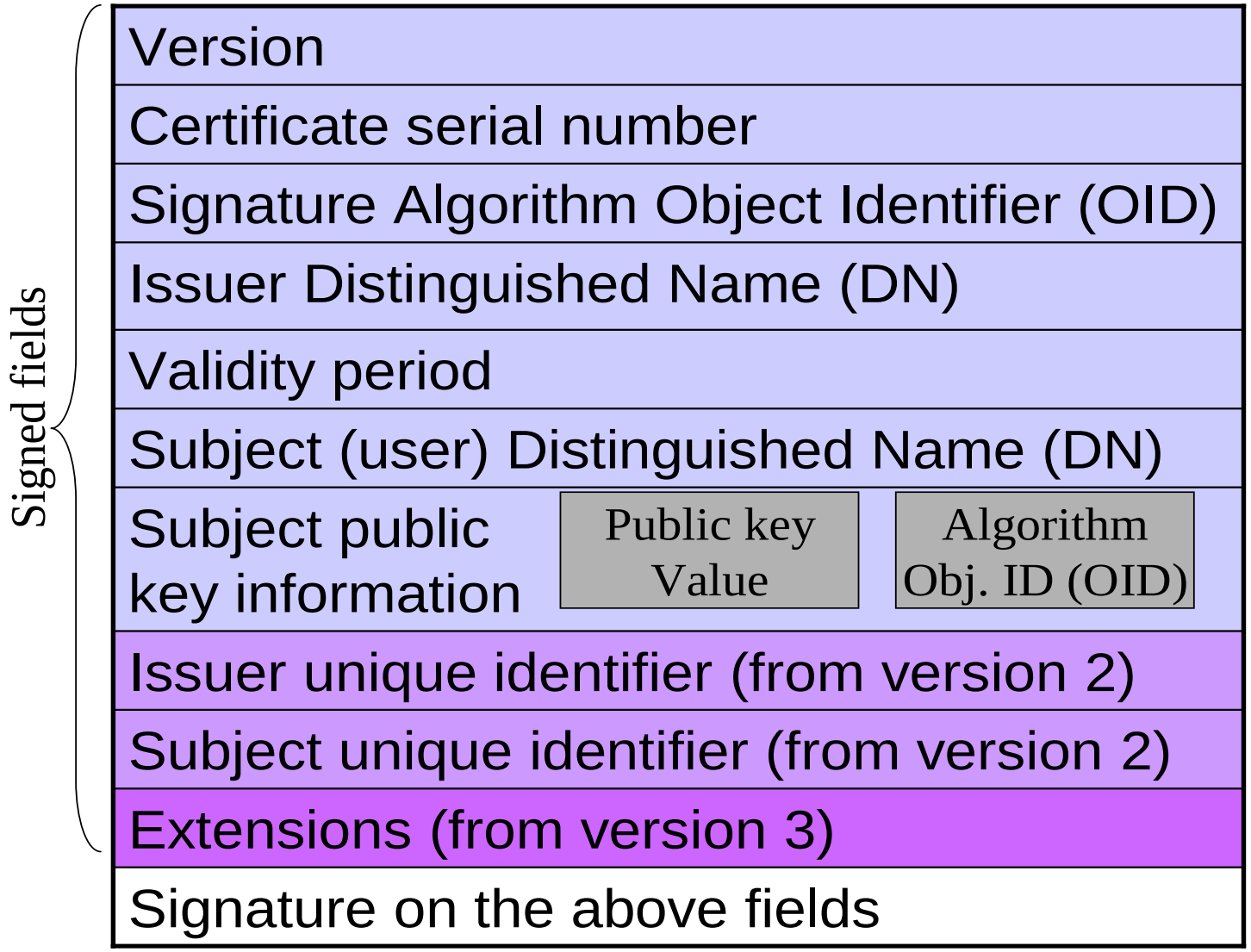
Distinguished Names - Evaluation

- **Decentralized?** 
  - Sure: any CA can select DN for its customers, sign cert
- **Unique ?** 
  - Could be, if each name space has one issuer
  - TLS reality: browsers trust 100s of CAs for **all** DN
- **Meaningful?** 
  - Usually: Julian Jones/UK/IBM
  - But not always: Julian Jones2/UK/IBM
    - Added ‘counter’ to distinguish → mistakes, loss of meaning
- X.509 response: v2: unique ID, v3: extensions

X.509 Certs & Subject Identifiers

- V1: Distinguished Name (for subject & issuer)
- V2: unique identifiers (for subject & issuer)
- V3: extensions
  - PKIX standard: SubjectAltName extension
    - Including DNSname
    - PKIX: Public Key Infrastructure working group of IETF
    - Widely adopted, including in SSL/TLS (& https)

X.509 Public Key Certificates

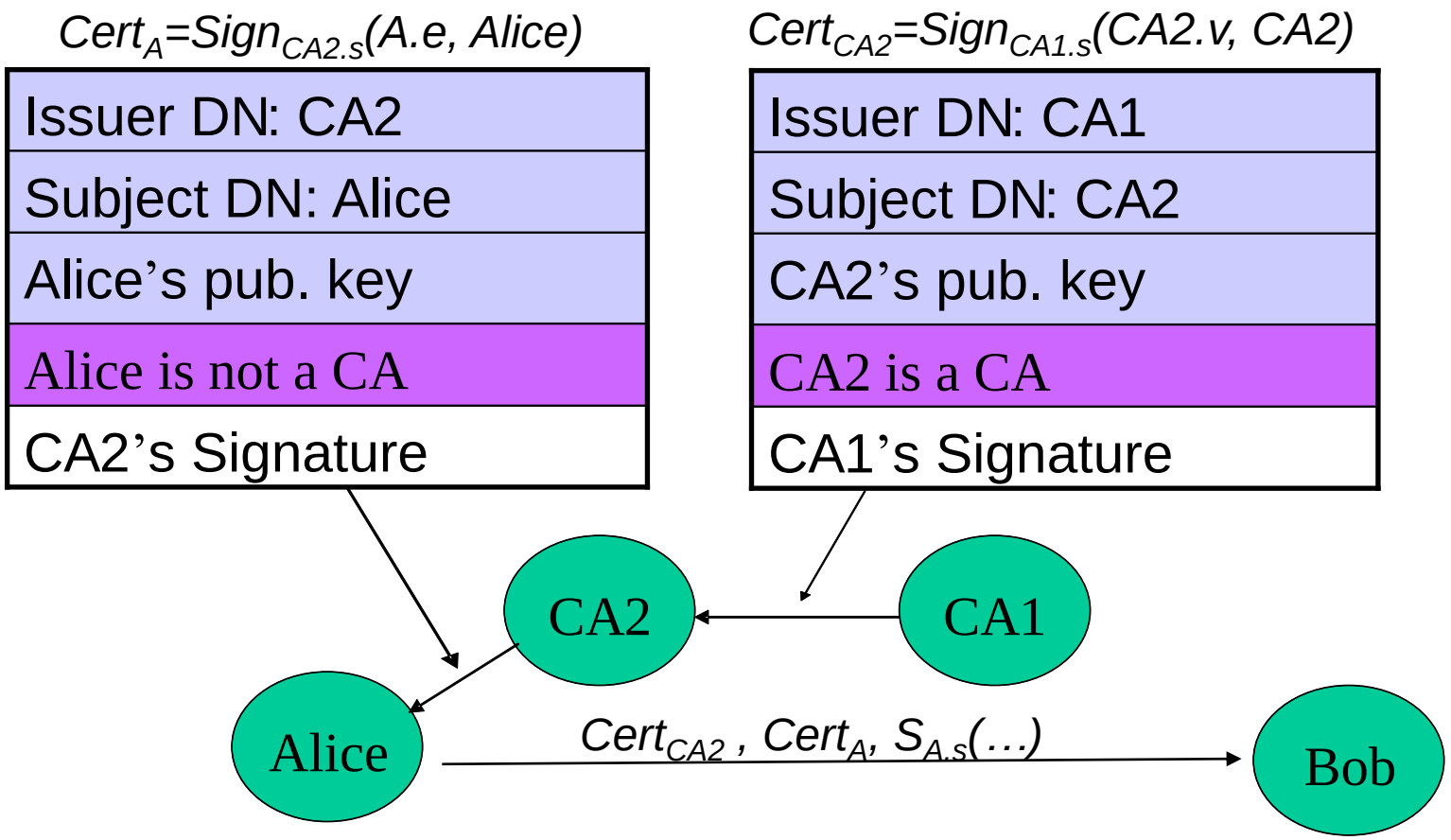


X.509 V3 Extensions Mechanism

- Each extension contains…
- Extension identifier
  - As an OID (Object Identifier)
  - E.g. `Naming constraints`
- Extension value
  - E.g. `Include C=IL`, `exclude dNSName=*.IBM.COM`
- Criticality indicator
  - If critical, relying parties MUST understand extension to use certificate
    - E.g. Naming constraints is `critical`
  - If non-critical, Ok to use certificate and ignore extension

Certificate Path

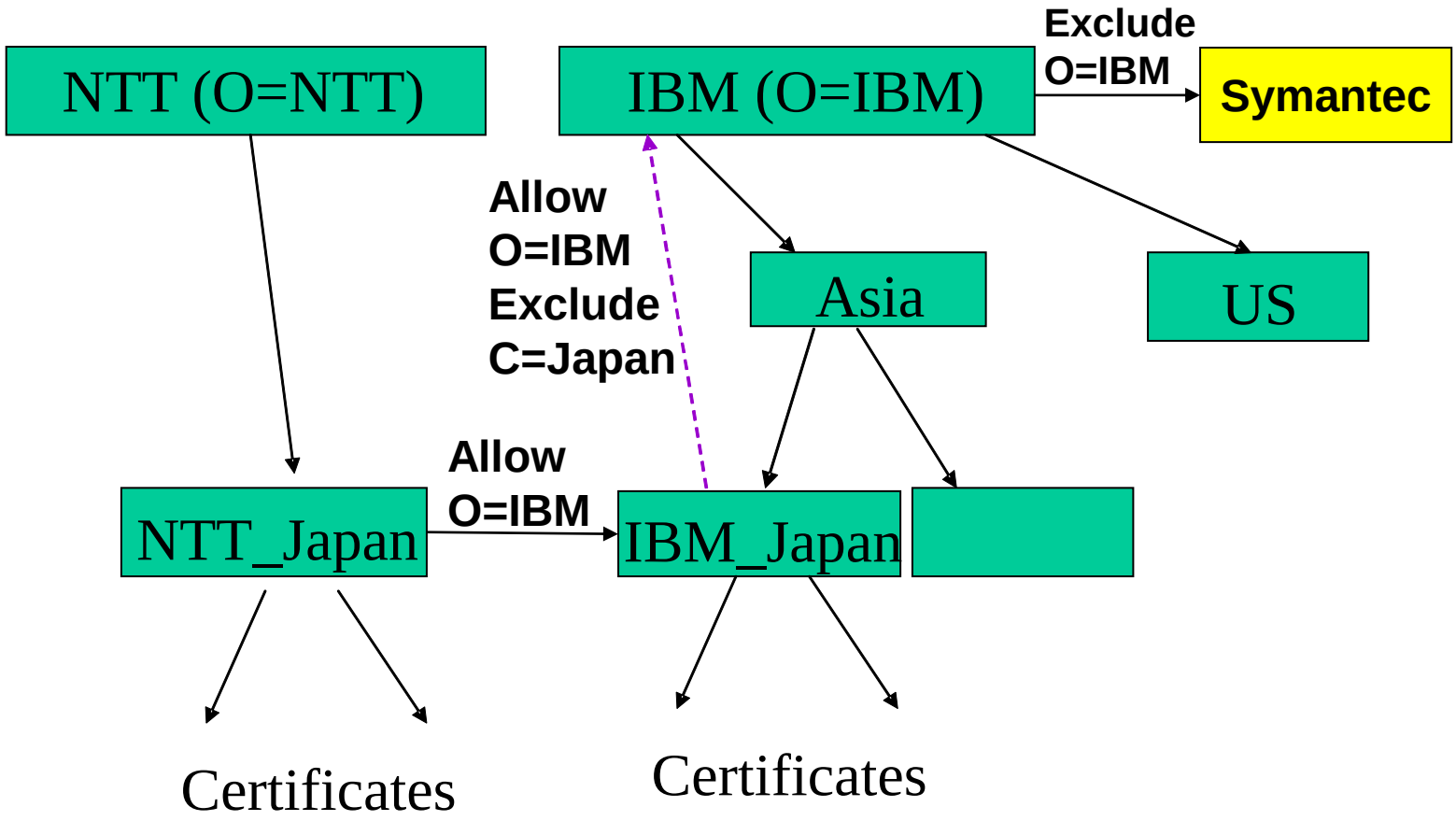
- Suppose relying party (browser) does not trust subject's CA…
- Solution: Certificate Path – a trusted CA certifies subject's CA



X.509v3/PKIX Standard Extensions

- Most important: Naming and Constraints extensions
- Certification path constraints extensions:
  - Basic constraints:
    - Goal: mark the (normal) case: subject isn't CA
    - `CA`: Subject is CA or end entity
    - `CertPathLength`
  - Naming constraints
    - Constraints on `DN` – in certs issued by subject
      - Only relevant when subject is a CA !
    - 'Allow' and 'Exclude'

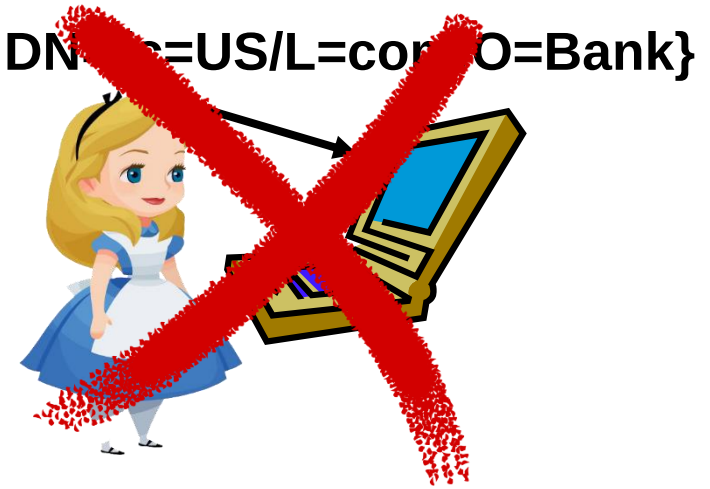
Applying naming constraints



- **NTT JP allows IBM JP to certify IBMers**
- **IBM JP allows IBM to certify all IBMers, except of IBM JP**
- **IBM trusts Symantec's certificates, except for O=IBM**

Reality: DNs aren’t usable identifiers

- Relying parties (users) don’t know the DN



- Hopefully, they know the domain (in URL)
- Naming extensions: alternative names
  - For TLS: `cert.SubjectAltName.dNSname`
  - Possible values: bank.com, *.bank.com (wildcard), …
  - May use also in naming constraints

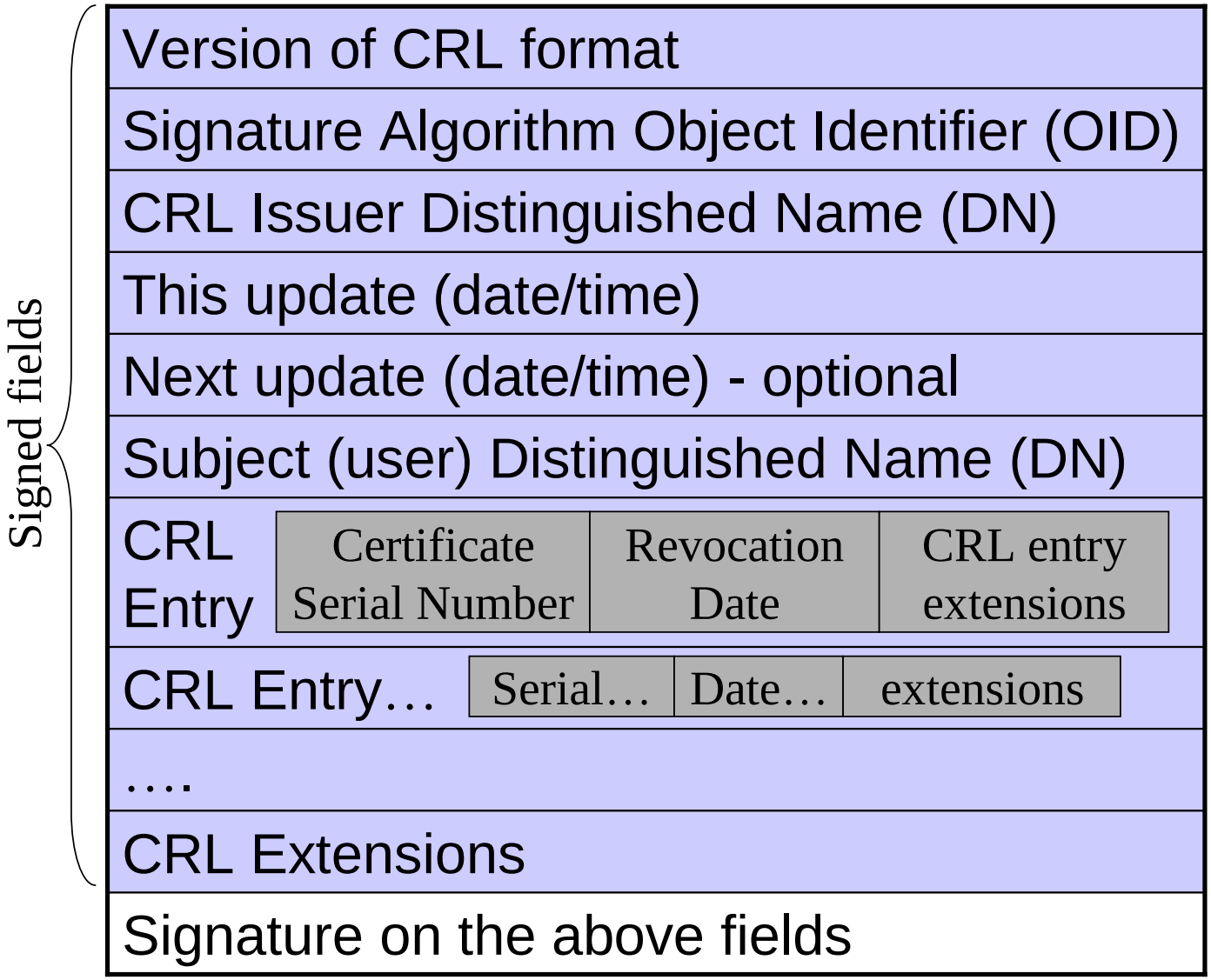
SSL / TLS PKI Challenges

- Many CAs `trusted' in browsers
- Every CA can certify any domain (name)
  - Since naming constraints NOT used
  - Two CAs can same name (equivocation)
  - To detect bad-CA: must find bad-certificate
    - No public, auditable log of certificates
- Several well-known failures
  - DigiNotar, Comodo, Stuxnet, …

Certificate Revocation

- Reasons for revoking certificate
  - Key compromise
  - CA compromise
  - Affiliation changed (changing DN or other attribute)
  - Superseded (replaced)
  - Cessation – not longer needed
- How to inform relying parties?
  - Do not inform – wait for end of (short?) validity period
  - Distribute *Certificate Revocation List (CRL)*
  - Ask - Online Certificate Status Protocol (OCSP)

X.509 CRL Format



Revocation is Difficult

- If CRLs contain all revoked certificates (which did not expire)… it may be huge!
- CRLs are (also) not immediate
  - Who is responsible until CRL is distributed?
  - What is the impact on non-repudiation?
- Solutions:
  - Online Certificate Status Protocol (OCSP)
  - More efficient CRL schemes (usually CRL extensions)
    - CRL distribution point – split certificates to several CRLs
    - Authorities Revocation List (ARL): list only revoked CAs
    - Delta CRL – only new revocations since last `base CRL`
    - Certificate Revocation Tree (more later)
  - Short validity for certificates

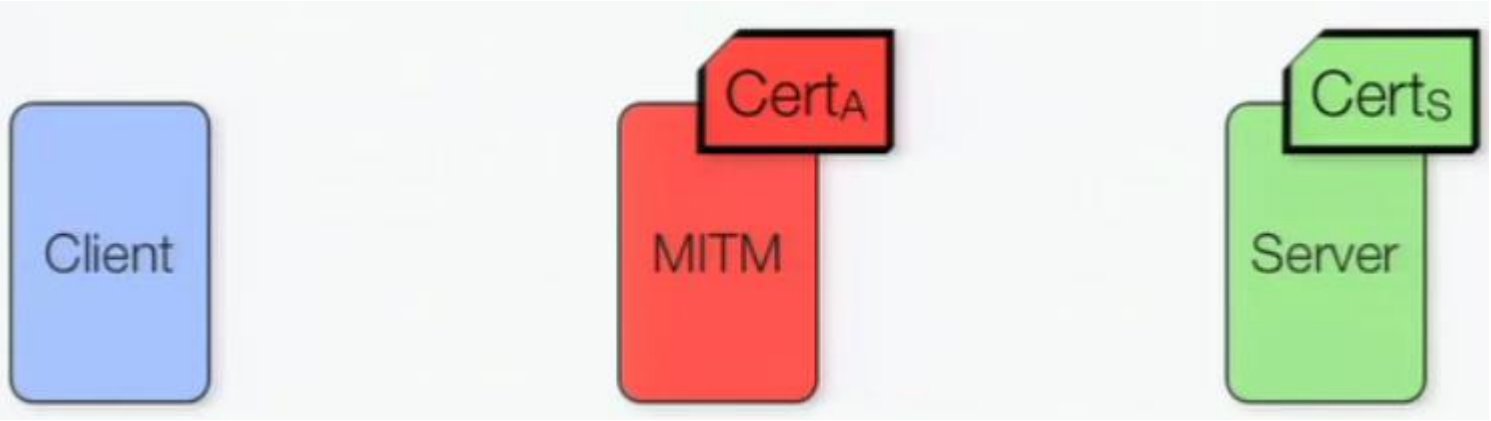
Short-Term Certificates

- Idea: short validity period of certificates, so no need to revoke them
- Concern: overhead of signing many certificates each (short) period
- Solutions:
  - Extend many certs with one signature: *hash tree*
    - $Sign_{CA.s}(date, valid{:}h(h(cert_A),h(cert_B),\ldots))$
    - Certificate revocation tree:
      $Sign_{CA.s}(date, all\ except{:}h(h(cert_A),h(cert_B),\ldots))$
  - Certificates includes a *hash chain*, e.g. for Jan 2005:
    $Cert_A=Sign_{CA.s}(A.s,"Alice",2005,h^{(11)}(x)))$
    - And for Feb 2005: $Cert_A$, $h^{(10)}(x)$
    - Validate incoming $Cert_A$, h_{10} by $h^{(11)}(x)=h(h_{10})$
    - Security based on random choice of x and h being one-way premutation
  - Often, requiring frequent CRL is more efficient

SSL / TLS PKI Challenges

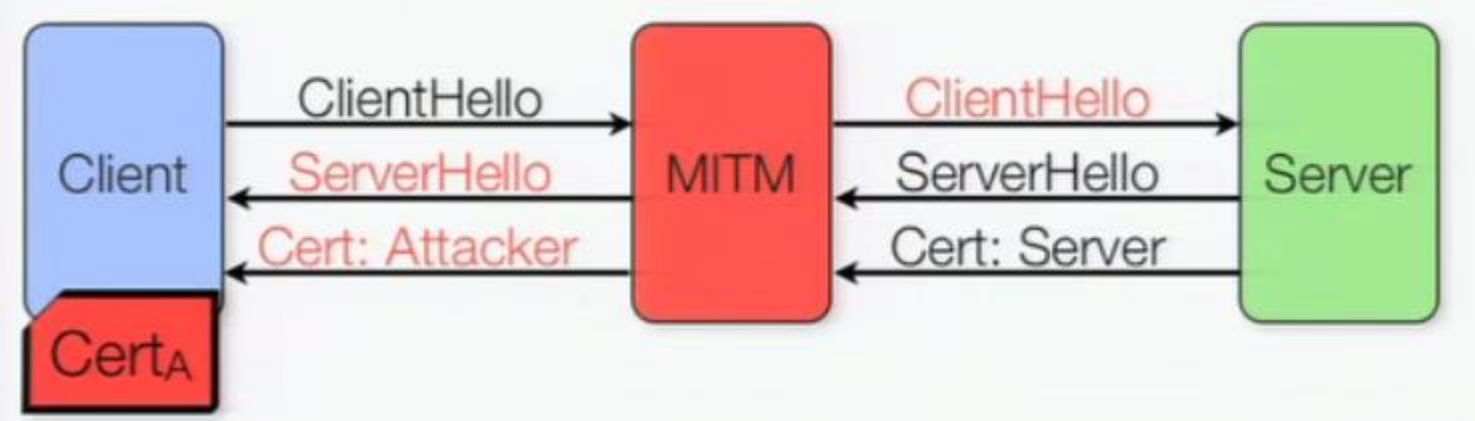
- Many CAs `trusted' in browsers
- Naming constraints NOT used
  - Every CA can certify any domain
- Several well-known failures
- DigiNotar, Comodo, Stuxnet, …

TLS Interception / MitM Attack



CertA is a fake-but-valid certificate for the identity of Server

TLS Interception / MitM Attack



Interception is used ethically, by 'locally' adding a CA, by many organizations, for filtering SSL/TLS traffic from malware, etc.

But also by attackers…

Defenses against Corrupt CAs

- **Use naming constraints to limit risk**
  - who can issue global TLDs (.com, etc.)?
- **‘Burned-in’ public keys (e.g., for Google)**
  - Detected MitM in Iran, using DigiNotar CA
- **Certificate / public-key pinning (HPKP)**
  - Server: I always use this PK / Cert / Chain
  - Client: remember and implement!
- Certificate Transparency (CT): Accountability
- Origin-bound certificates

Defenses against Corrupt CAs

- Use naming constraints to limit risk
- ‘Burned-in’ public keys (e.g., for Google)
- Certificate / public-key pinning (HPKP)
- **Certificate Transparency (CT): Accountability**
- Threshold schemes

Key Establishment & PKI : Conclusions

- Key Establishment: use PKs, cert for ‘handshake’
  - SSL/TLS: mature standard, widely used
    - Many vulnerabilities in older versions
    - Slow adoption of newer versions
- PKI & Trust: still challenging, active areas
  - SSL/TLS certs: too many legit CAs, no naming constraints
  - How to deal with rogue CAs?
    - Certificate Transparency (accountability) ?
  - Client certificates (authentication) ???

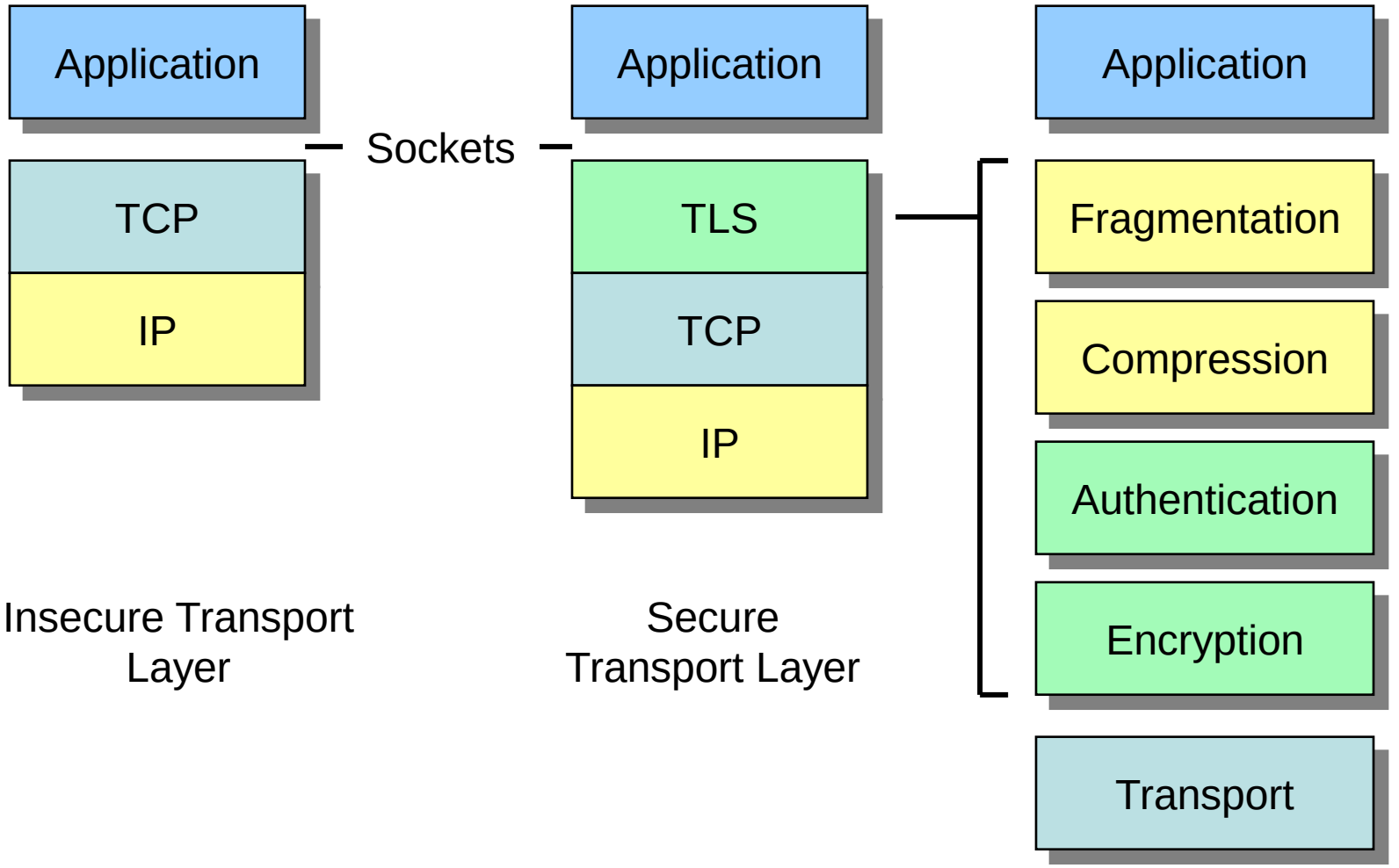
SSL / TLS

PPKE, ITK
Csapodi Márton

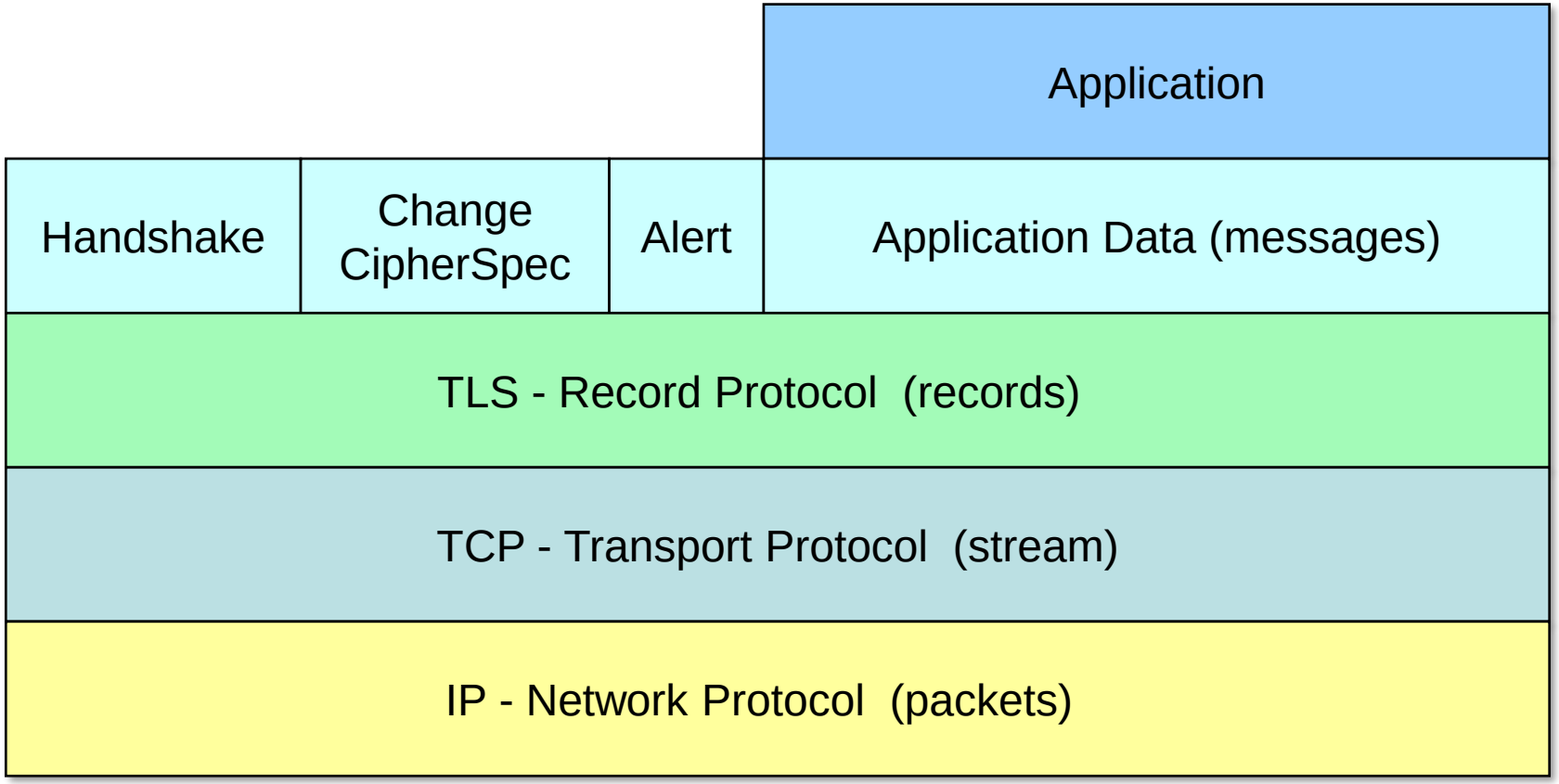
Secure Network Protocols for the OSI Stack

Communication layers	Security protocols
Application layer	ssh, S/MIME, PGP, Kerberos, WSS
Transport layer	TLS, [SSL]
Network layer	IPsec
Data Link layer	[PPTP, L2TP], IEEE 802.1X, IEEE 802.1AE, IEEE 802.11i
Physical layer	Quantum Cryptography

SSL/TLS Protocol Layers

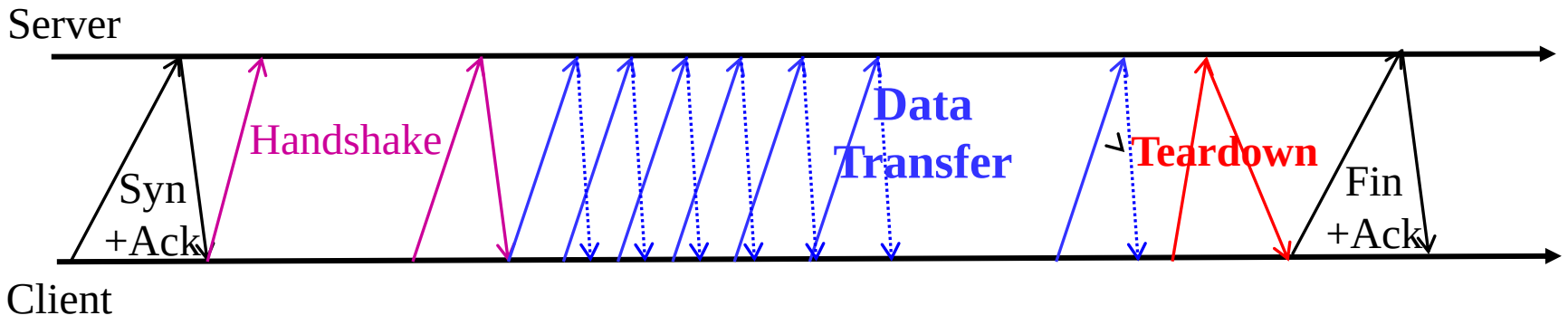


SSL/TLS Protocol Layers



SSL/TLS Operation Phases (high level)

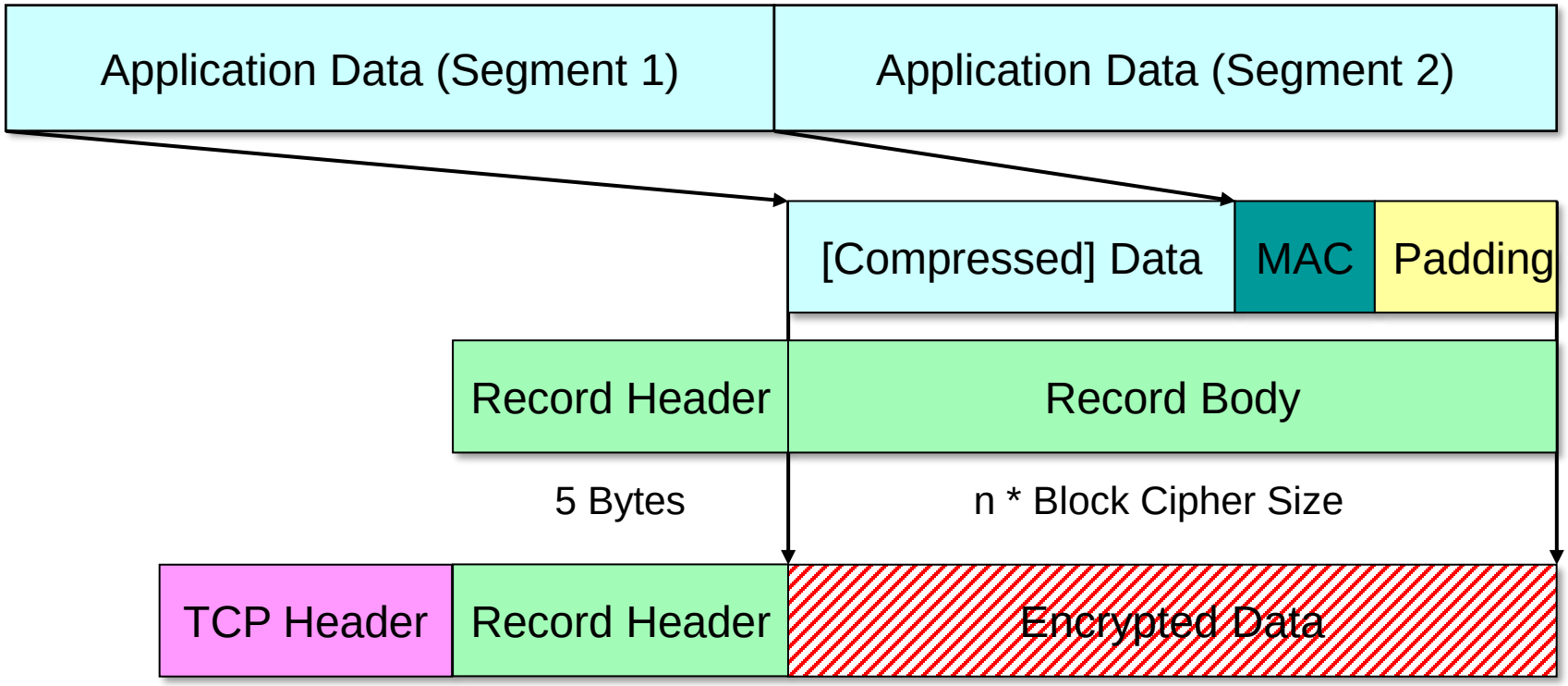
- TCP Connection setup (Syn+Ack)
- Handshake (key establishment)
  - Negotiate (agree on) algorithms, methods
  - Authenticate server and optionally client, establish keys
- Data transfer
- Secure Teardown
- TCP connection closure (Fin+Ack)



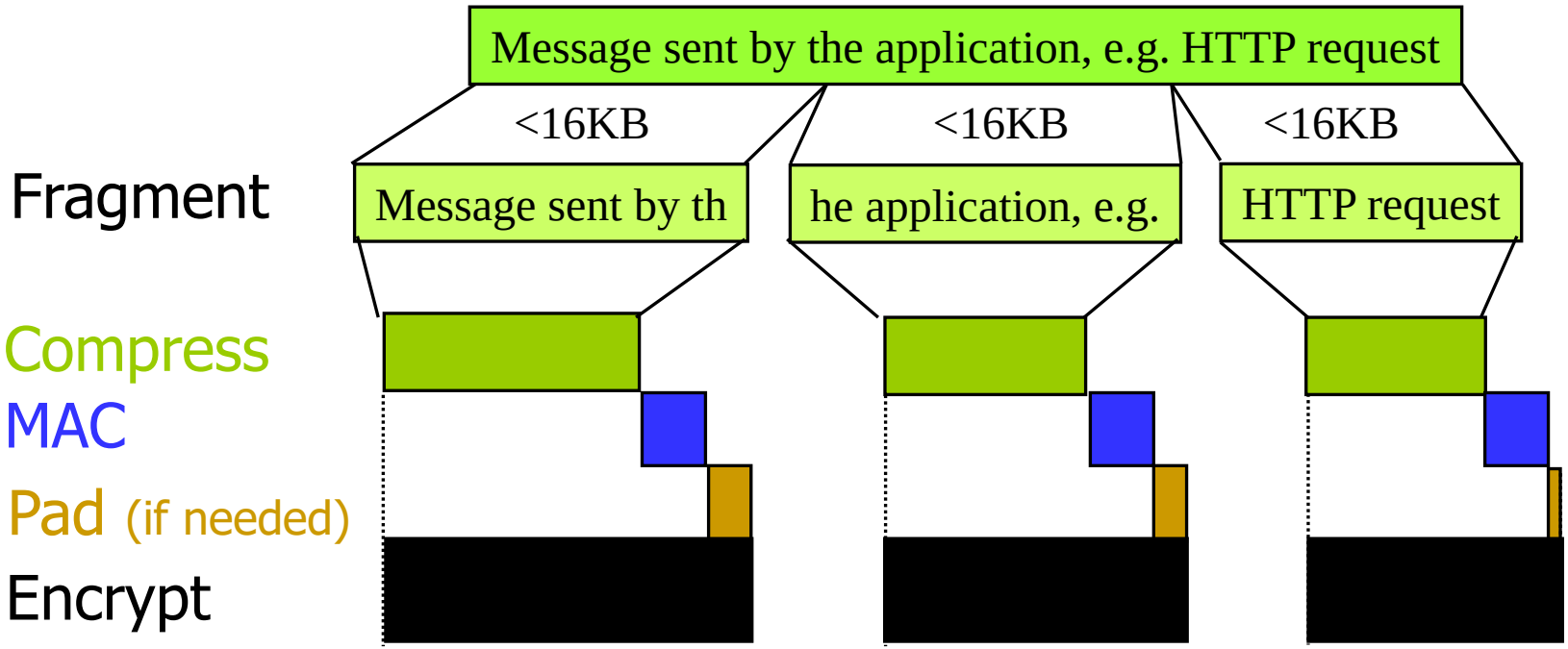
Data transfer: Record Protocol

- Assumes underlying reliable communication (TCP)
- Four services (in order):
  - Fragment: break TCP stream into fragments (<16KB)
    - Pipeline: send processed frag 1 while processing 2 and receiving 3
  - Compress (lossless) each fragment
    - Reduce processing, communication time
    - Ciphertext cannot be compressed – must compress before
    - Risk: exposure of amount of redundancy → *compression attacks*
  - Authenticate: [seq#||type||version||length||comp_fragment]
  - Encrypt
    - After padding (if necessary)
- Finally, add header: type (protocol), version & length

Fragmentation, compression, authentication, encryption



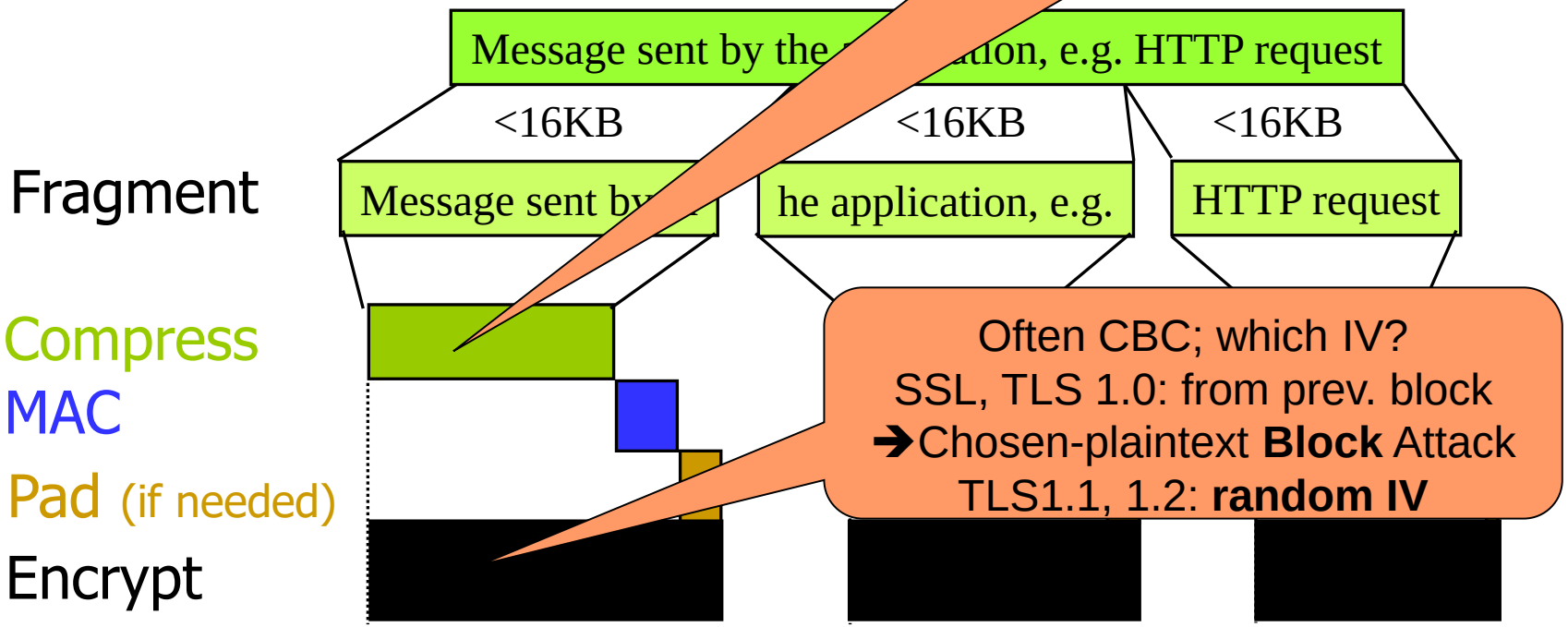
Fragmentation, compression, authentication, encryption



Send each block via TCP

Fragmentation, c… authentication, …

Fragment then Compress:
simpler - but revealing ?
TLS1.1,1.2: pad to fixed-lengths
to hide **exact** length
Exploited: CRIME, TIME attacks



Send each block via TCP

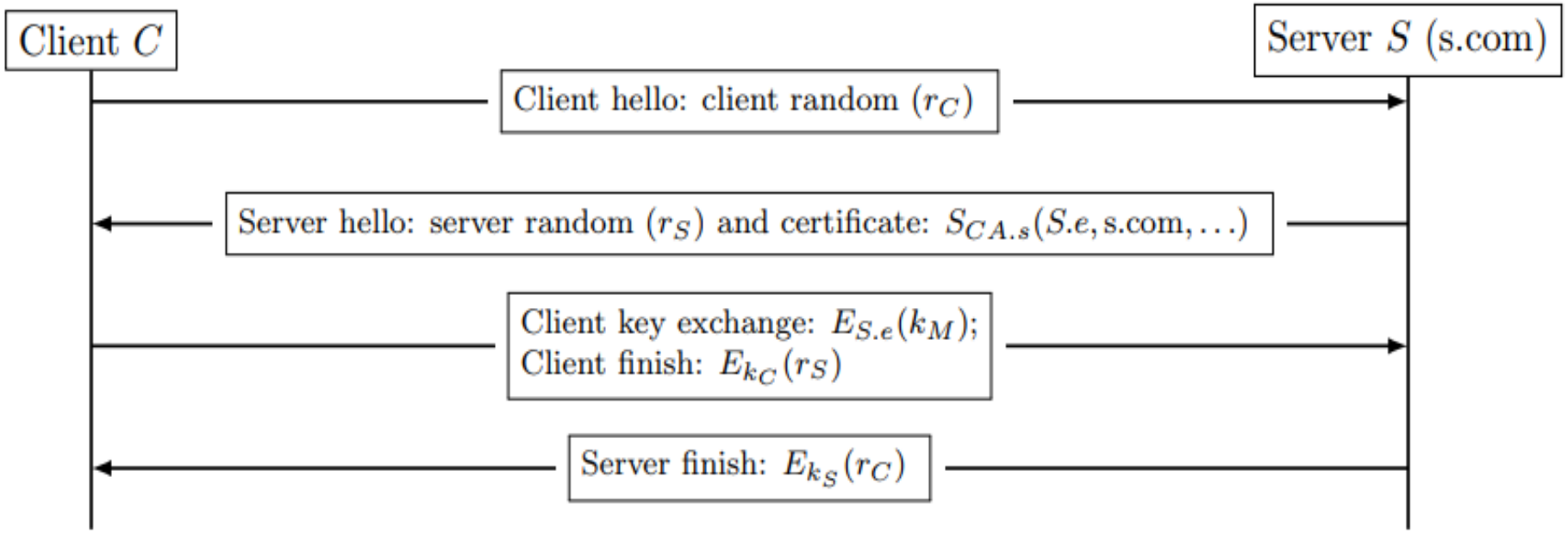
Vulnerabilities

- Surprisingly many found, exploited!
- ➔ SSL, TLS1.0: vulnerable record protocol:
  - Attacks on RC4 → to be avoided
  - CBC IV reuse in session (BEAST)
  - MAC-then-encrypt: padding attacks (Lucky13, POODLE)
  - Compress-then-encrypt: CRIME, TIME
  - downgrading to use vulnerable version
  - etc.

SSL/TLS Handshake Protocol

- The beginning: SSLv2
  - SSLv1 was never published, released
- The evolution: from SSLv3 to TLS 1.2
  - TLS: the IETF version of SSL
- State-of-Art: TLS 1.3
  - Significant changes
- Our focus is on the handshake protocol

Simplified SSLv2 Handshake



- Key derivation in SSLv2:
  - Client randomly selects k_M and sends to server
  - Client and server derive (directional) encryption keys:

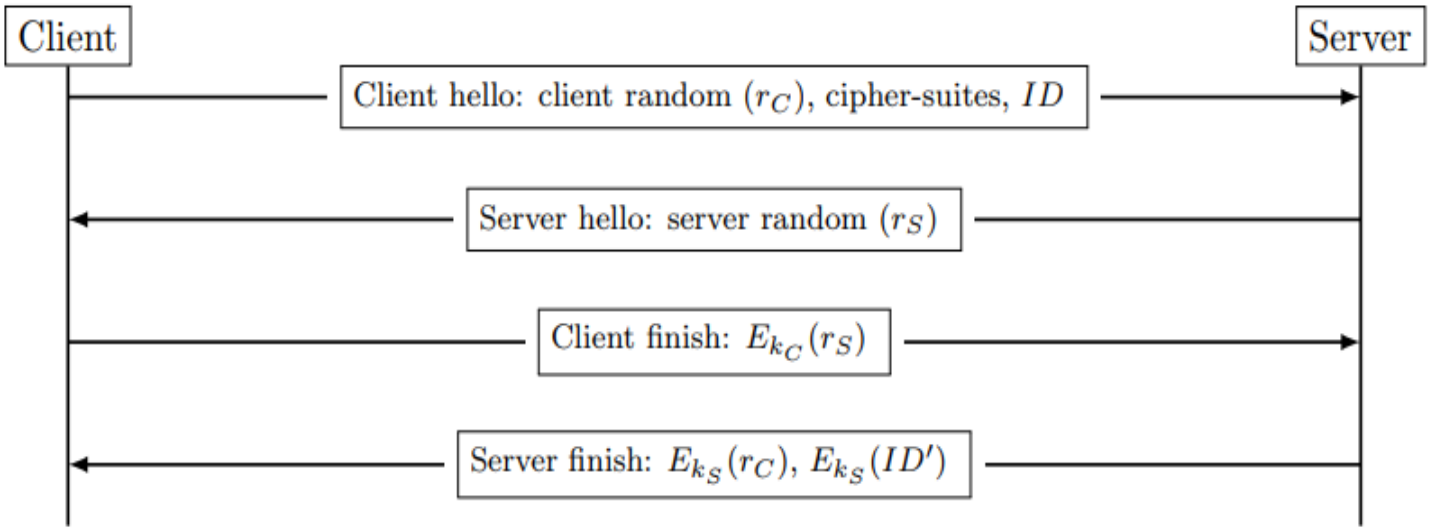
$$k_C = MD5(k_M || \text{“0”} || r_C || r_S) \qquad k_S = MD5(k_M || \text{“1”} || r_C || r_S)$$

SSLv2: important concepts

- Derive, from master key k_M, two separate keys:
  - k_C, for protecting traffic from client to server
  - k_S, for protecting traffic from server to client
  - Nonces r_C, r_S protect against replay
    - Even if client reuses same PK encryption of k_M
- Sessions: reusing public-key operations
- Cipher-agility
- Optional client authentication

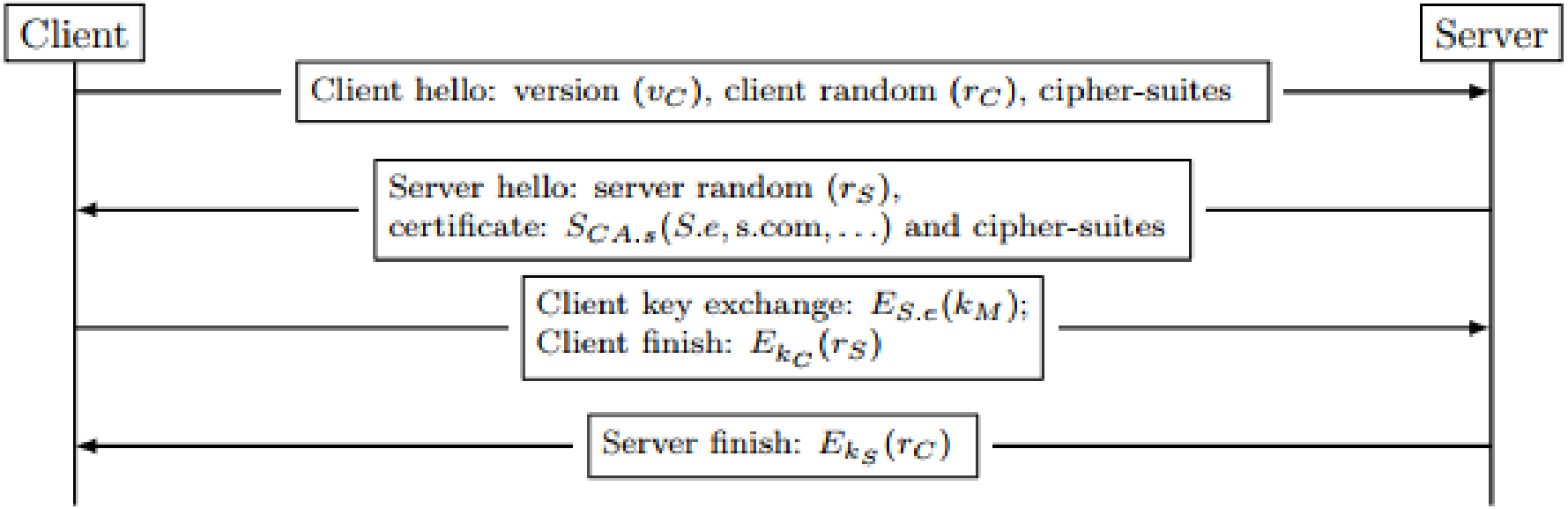
SSLv2 Session Resumption

- Goal: cache shared master key k_M (and *ID*)
  - By both client and server
  - Client identifies cached key by sending *ID* (if known)
  - If server knows *ID*, it sends only nonce (no cert)
  - Server sends (new) identifier *ID'* at end of handshake



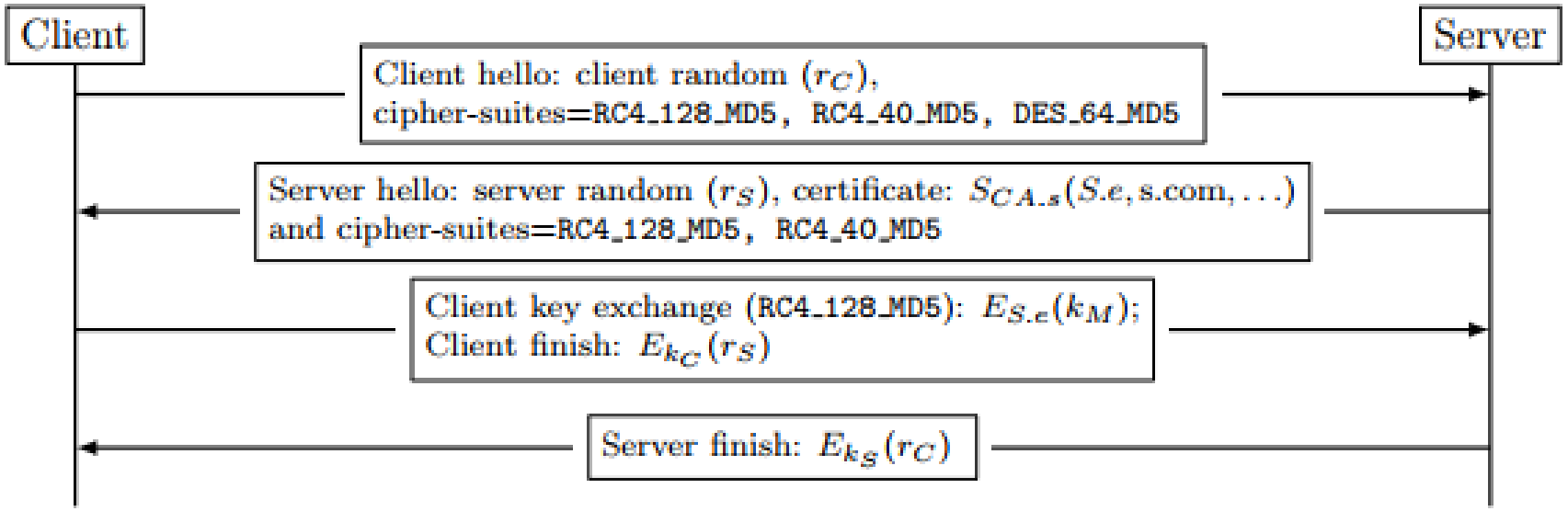
SSLv2 Ciphersuite Negotiation

- Client, server sends cipher-suites
- Client specifies choice in client-key-exchange



SSLv2 Ciphersuite Negotiation

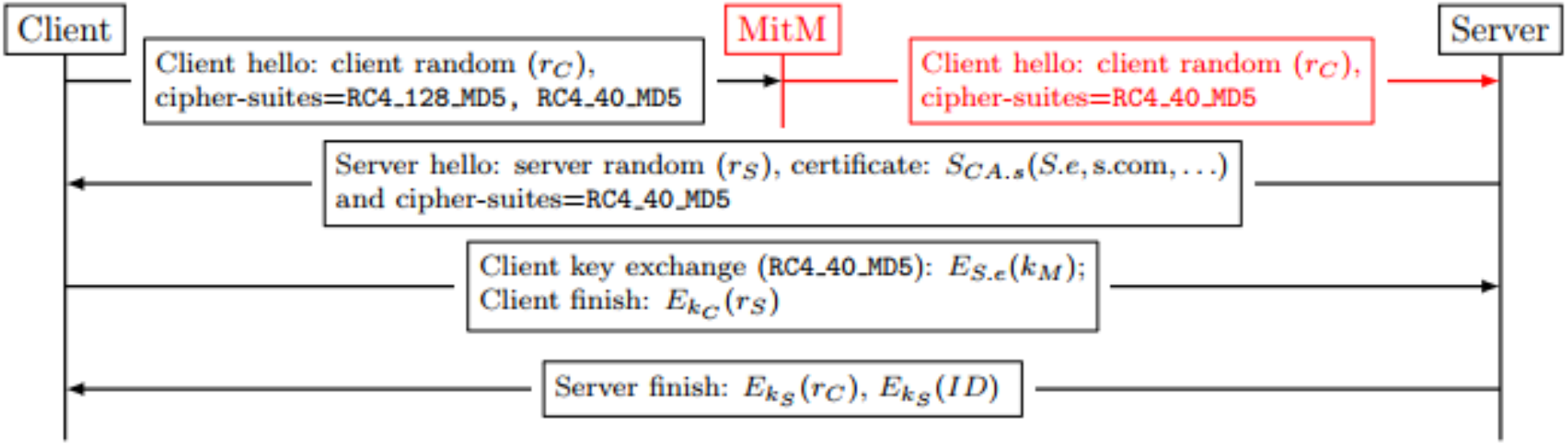
- Client, server sends cipher-suites
- Client specifies choice in client-key-exchange



- Example: RC4_128_MD5 chosen
- Vulnerable to downgrade attack!

SSLv2 Downgrade Attack

- Server and client tricked into using (insecure) 40-bit encryption (`export version')

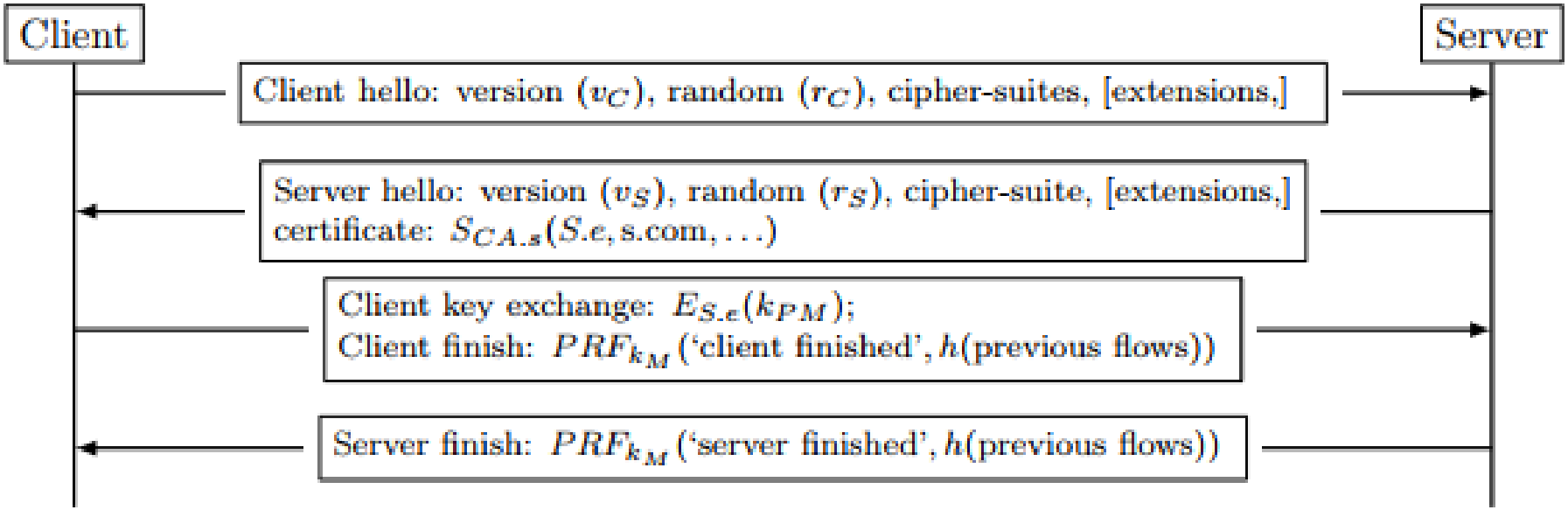


- Attacker may record connection and decrypt later – no need for real-time cryptanalysis!

The evolution: SSLv3, TLS1.0, 1.1, 1.2

- Main improvements:
  - Improved key derivation
    - Premaster key → master key → connection keys
  - Improved negotiation and handshake integrity
    - Prevents SSLv2 downgrade attack
    - Secure extensions, protocol-negotiation & more
  - DH key exchange and PFS (perfect forward secrecy)
    - SSLv2 allowed only RSA; TLS 1.3: only PFS
  - Session-ticket resumption

Basic RSA Handshake: SSL3-TLS1.2



$$k_M = PRF_{k_{PM}}(\text{“master secret”} \| r_C \| r_S)$$

$$key\text{-}block = PRF_{k_M}(\text{‘key expansion’} \| r_C \| r_S)$$

k_C^A	k_S^A	k_C^E	k_S^E	IV_C	IV_S

SSL3-TLS1.2: Key Derivation

- Handshake exchanges premaster key
- Derive master key (PRF: pseudo random function):

$$k_M = PRF_{k_{PM}}(\text{“master secret”}||r_C||r_S)$$

  - In case premaster key is not (fully) random
    - Weak randomness at a (weak) client
    - Weak client reuses same PK-encrypted key
    - DH-derived premaster key

SSL3-TLS1.2: Key Derivation

- Handshake exchanges premaster key
- Derive master key:

$$k_M = PRF_{k_{PM}}(\text{“master secret”}||r_C||r_S)$$

- Derive key block from master key:

$$key\text{-}block = PRF_{k_M}(\text{‘key expansion’}||r_C||r_S)$$

- Chop keys from key-block (A: authentication, E: encryption):

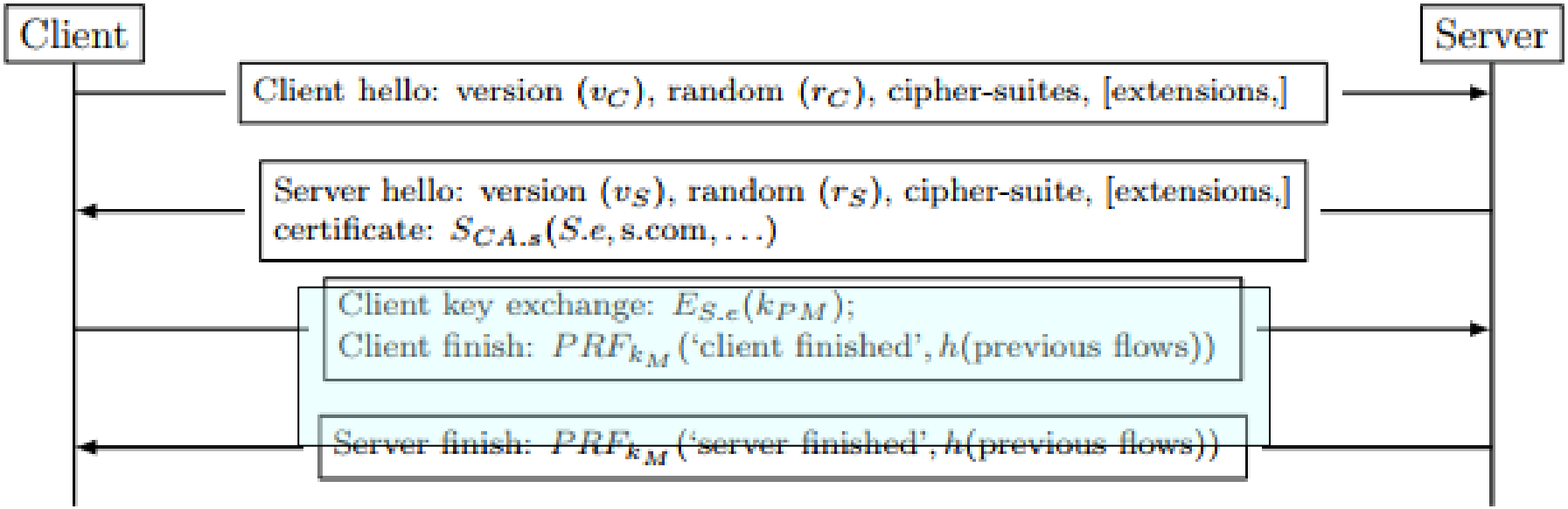
$key\text{-}block = PRF_{k_M}(\text{‘key expansion’}||r_C||r_S)$					
k_C^A	k_S^A	k_C^E	k_S^E	IV_C	IV_S

SSL3-TLS1.2: Agility and Integrity

- SSLv2: limited cipher-agility (ciphersuites)
  - And no integrity: vulnerable to downgrade attack
- SSLv3 to TLS1.2: integrity + improved agility:
  - Handshake integrity – foils downgrade attack!
  - Backwards compatibility
  - TLS extensions
  - Version-dependent key separation

SSL3-TLS1.2: Handshake integrity

- Foils the downgrade attack on SSLv2
- Extend the finish-message validation: authenticate entire previous handshake flows



SSL3-TLS1.2: Backward compatibility

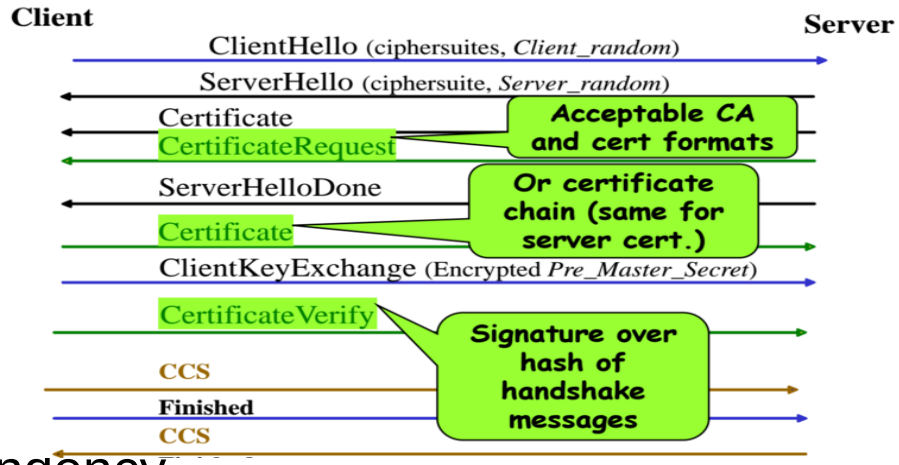
- Challenge: upgrading existing protocol
  - Unrealistic: all upgrade at same day
  - Backward compatibility: new (server, client) can still work with old (client, server)
    - Server selects version based on client's (in 'hello')
    - Downgrade prevented using 'finish' authentication
- Dilemmas for clients:
  - Some servers fail to respond to new handshake
  - 'Downgrade-dance' clients: try new versions, then older → vulnerable!

Advanced Handshake Features

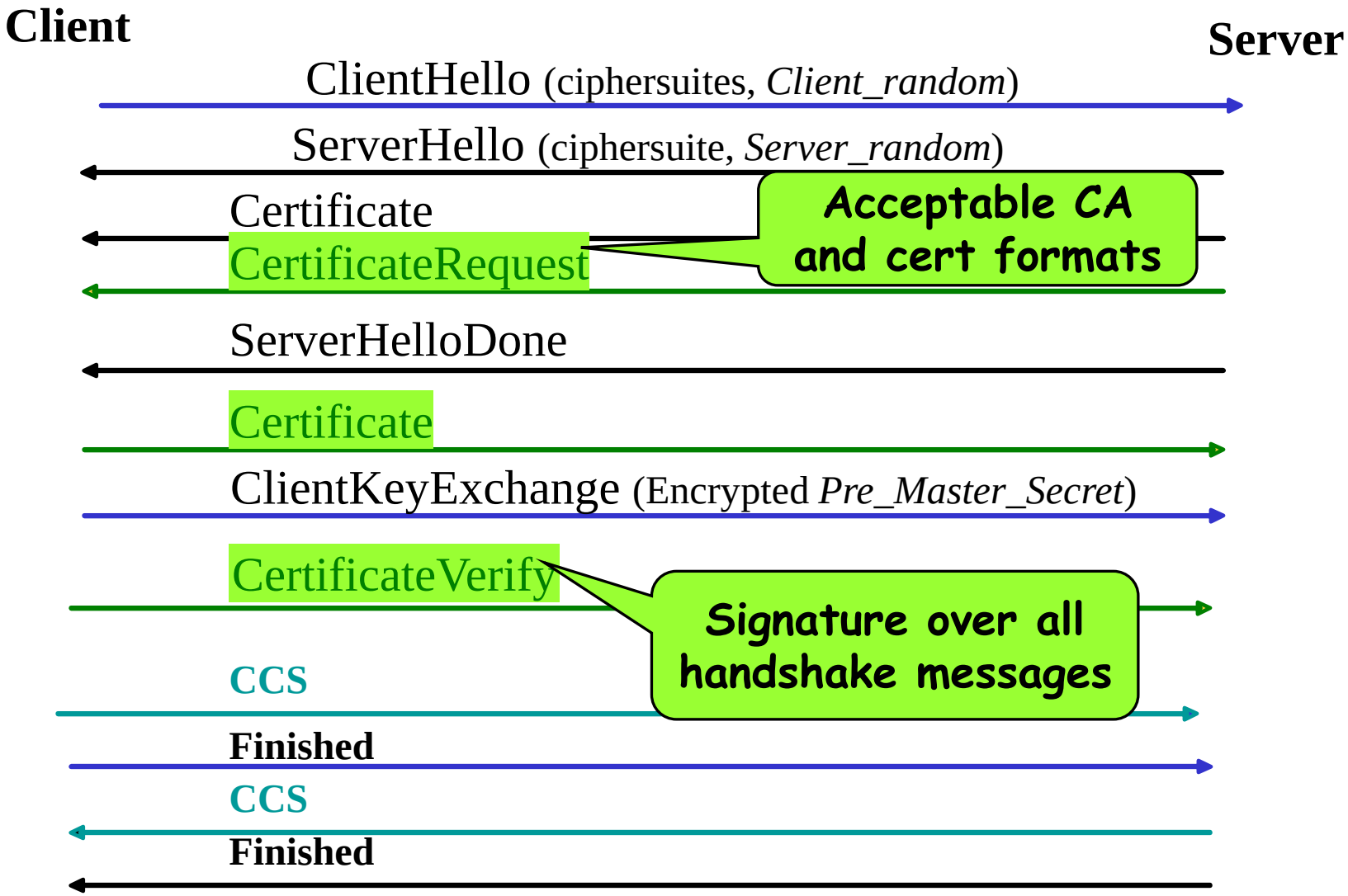
- Client authentication
- Perfect Forward Secrecy (PFS)
  - ephemeral Diffie-Hellman keys
- Session resumption (ID-based, ticket)
- TLS 1.3 handshakes

TLS/SSL Client Authentication

- Usually, TLS/SSL used only with server PK
  - Only allows client to authenticate server
  - Client authentication: encrypt secret (pw, cookie)
- But TLS/SSL also allows client certificates
- How?
  - Client authenticates by signing with certified PK
- Easy – no PW!
- But: PKI challenges, device dependency
- ➔ Limited use, mainly within organization/community



TLS/SSL Client Authentication



SSL Client Authentication: Issues

Which identifier?

- No global, unique namespace
- Result: each server use its own client names, certificates

Support for mobility of cert and key…

- Smartcard, USB `stick`?

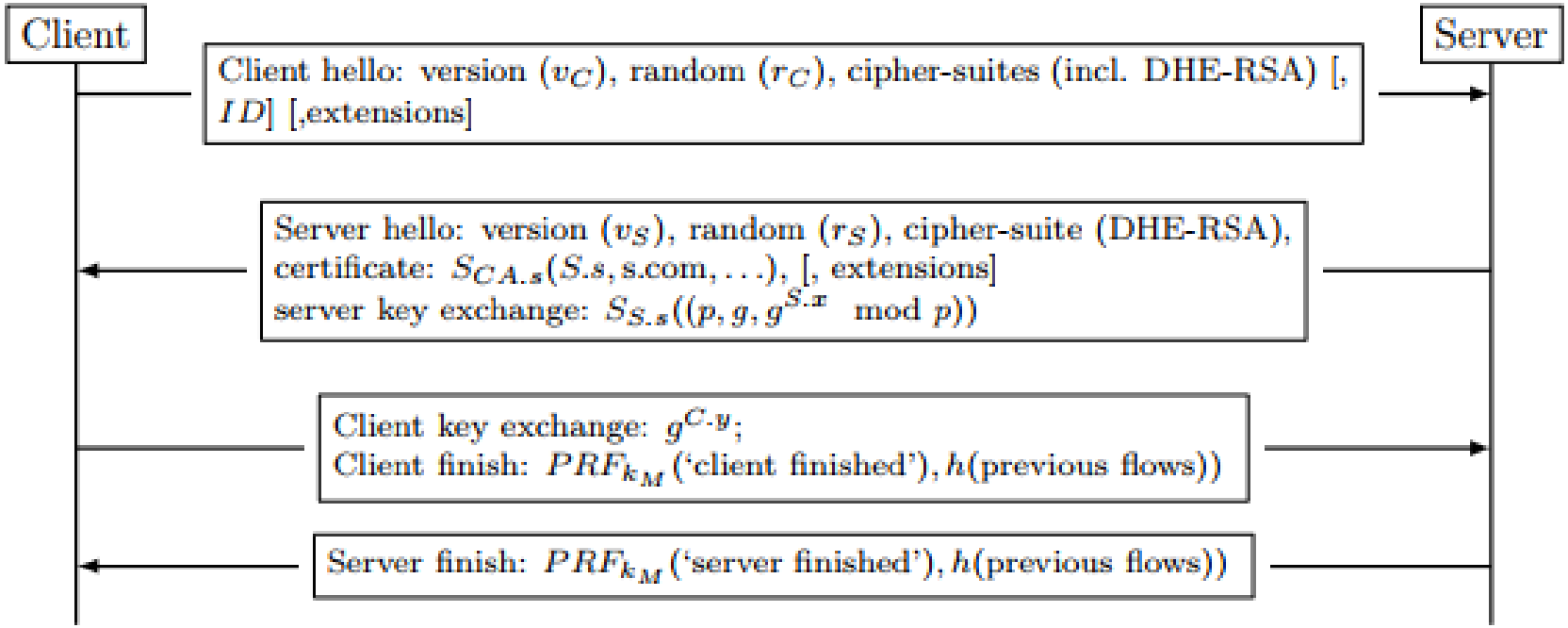
➔Rarely used

Ephemeral Diffie-Hellman keys

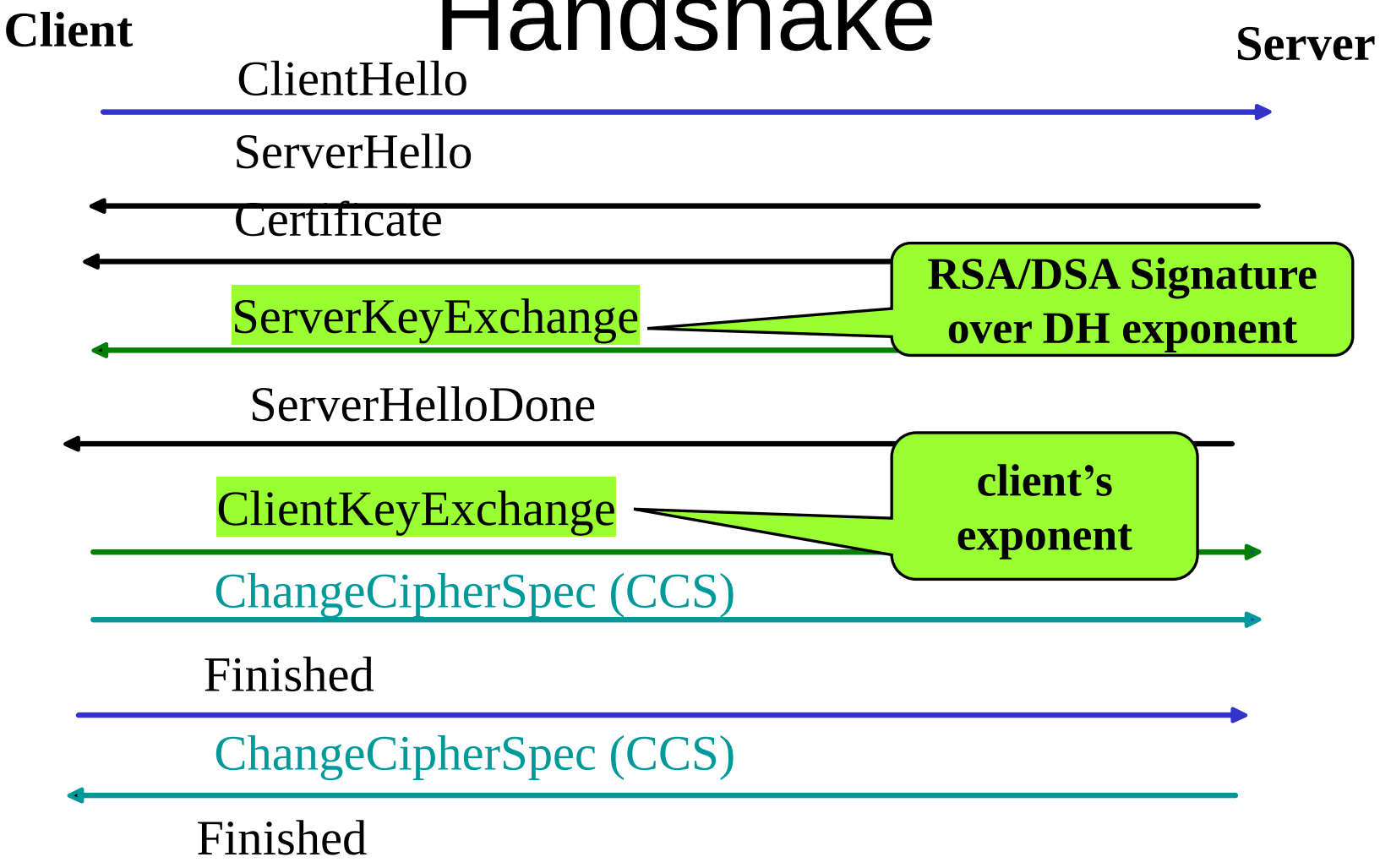
- Ephemeral keys: per-connection
  - Per-connection public keys ? Why?
- Motivations?
  - Perfect forward security: present traffic immune from future exposure – incl. of past keys
  - Historical: ‘export-grade’ (weak) keys (512 bit RSA)
- How?
  - Diffie-Hellman key exchange
  - Authenticated using long-term keys

TLS/SSL Handshake: Ephemeral DH

- Server signs a DH exponent $g^{S.x}$
  - E.g., using RSA signatures

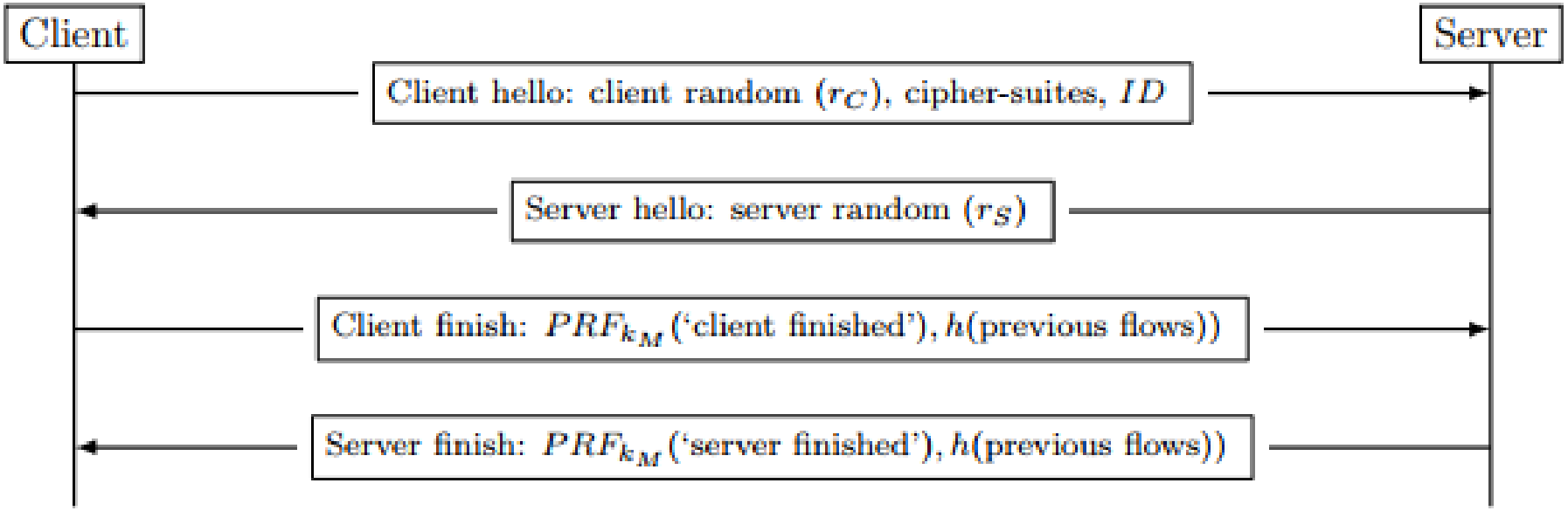


TLS/SSL Ephemeral PK Handshake

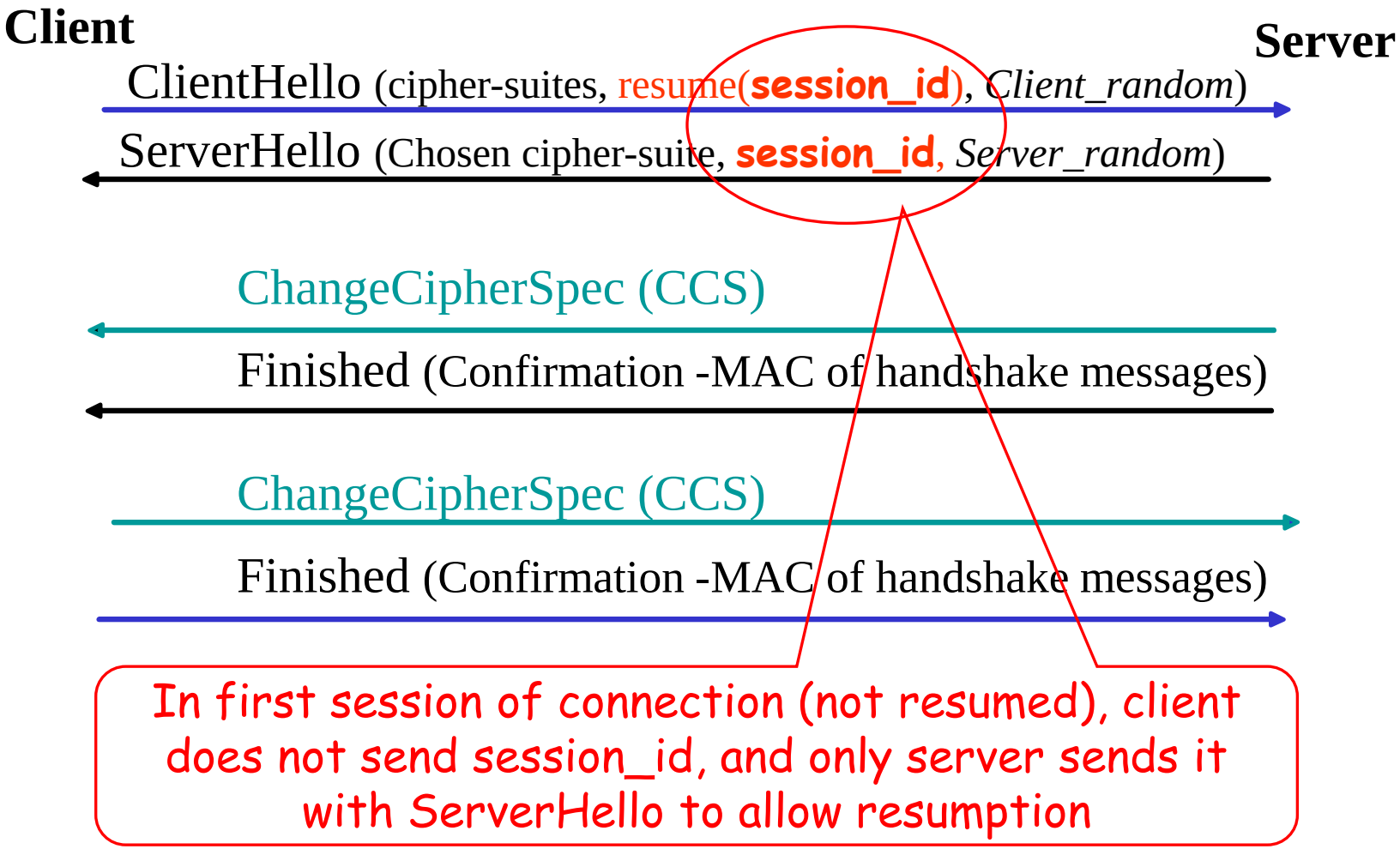


ID-based Session Resumption

- Idea: server, client store (ID, key) per peer
- Reuse in new connections btw same pair
- Saves PK operations (CPU, BW)



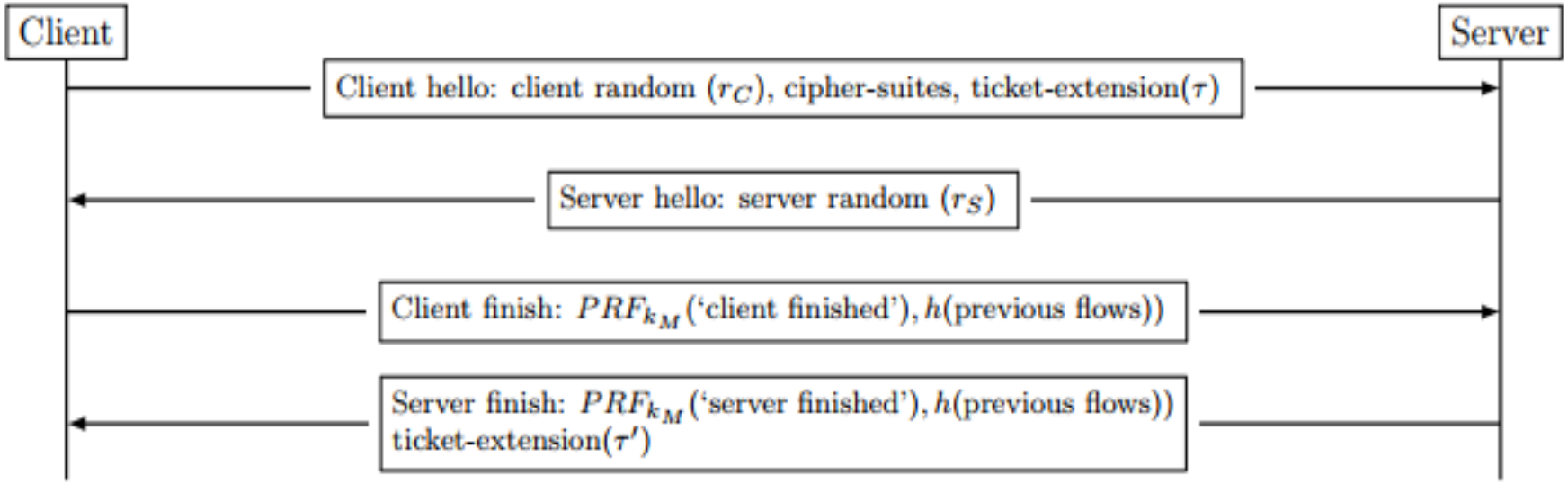
Session-ID Resumption Handshake



Session Resumption Issues

- Need to keep state, lookup ID…
  - Overhead (➔small cache: less effective)
  - Need to share among (many!) replicates of server
  - For PFS: ensure keys disappear after ‘period’
- Solution: Client-side caching (Session-Ticket Hello Extension)
  - Ticket contains master key, encrypted by a secret session ticket key, known (only) to server
    - Share with other servers of this site
    - Change periodically to enforce PFS
  - Uses TLS extension (not in SSL)

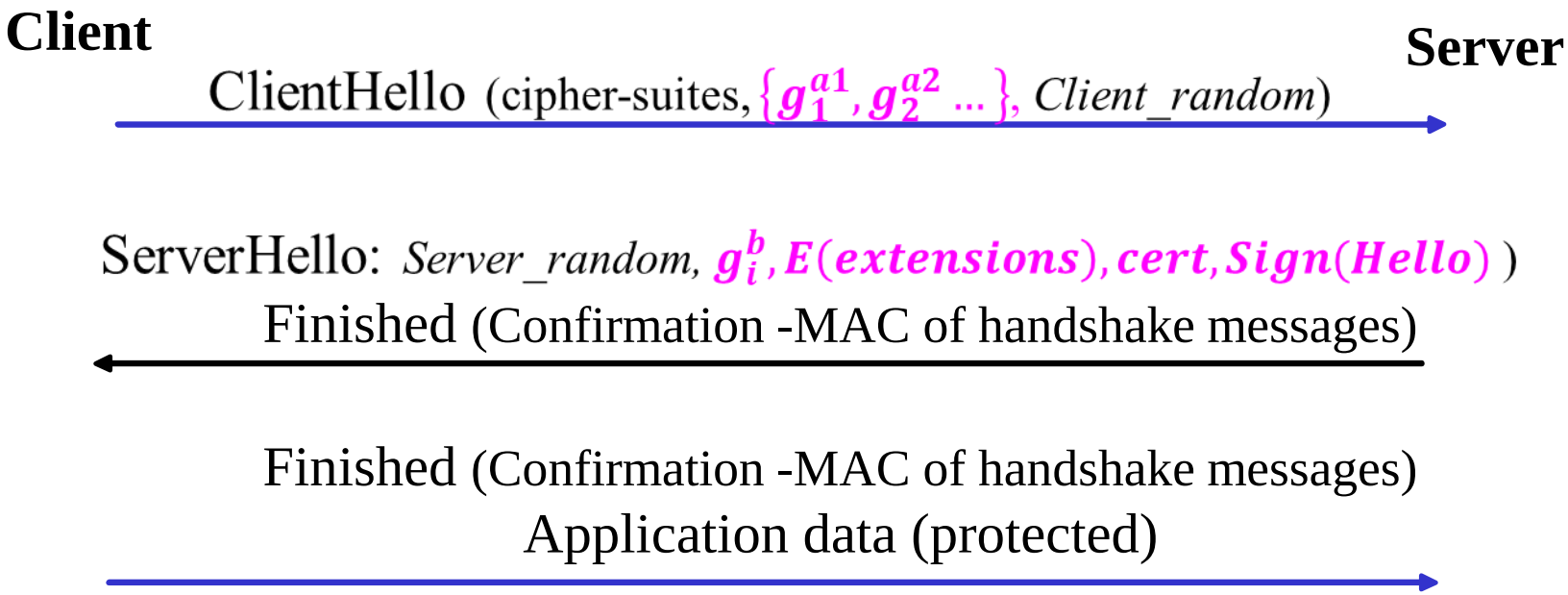
Session-Ticket Resumption



- To preserve PFS:
  - Tickets ‘expire’ after ‘time period’ (e.g., 24 hours)
  - Ticket-key changed rapidly (e.g., every hour or few)
  - Ticket-key erased after `time period’ ends (e.g., daily)
- Problem: many servers do not limit ticket-key lifetime

TLS 1.3 ‘Full handshake’: 1-RTT

- No RSA: only DH + signature by server
- 1-RTT: one round trip time



A Kerberos-tól az OpenID Connect-ig

PPKE, ITK

Csapodi Márton

A Kerberos célja

forrás:
http://technet.microsoft.com/en-us/library/cc772815(v=ws.10).aspx
http://www.kerberos.org/software/tutorial.html
http://www.rfc-editor.org/rfc/rfc4120.txt

„Kerberos is an authentication protocol for trusted hosts on untrusted networks”

célok:

- felhasználó (kliens gép) hitelesítése az alkalmazás szerverek felé
- a felhasználói jelszó nem utazik, nem tárolódik (csak titkosítva a szerveren)
- központi felhasználó menedzsment
- single-sign-on
- kölcsönös azonosítás (opció)
- titkosított kommunikáció (opció)

Kerberos fogalmak / 1

realm/domain: Tartomány, melyben a kliensek egyetlen autentikációs szerverhez kapcsolódnak. Az igénybe vett szolgáltatás más tartományban is lehet (cross-domain autentikáció).

principal: Felhasználó vagy szolgáltatás a tartományon belül, egyedi névvel.

ticket: Jegy, melyet a kliens mutat be az igénybe vett szolgáltatásnak az autentikáció során. Tartalma titkosítva van a szolgáltatás szimmetrikus kulcsával (melyet a ticket kibocsátója ismer, de a kliens nem). Tartalma: kliens principal, szolgáltatás principal egyedi neve, kliens IP cím (opcionális), érvényesség kezdete, érvényesség időtartama, session kulcs. A ticket bemutatásakor igazolni kell a session kulcs ismeretét (authenticator).

session key: A ticket kibocsátója hozza létre, és külön titkosítja a kliens és a szolgáltatás szimmetrikus titkosító kulcsával. Ennek ismerete a (kölcsönös) autentikáció alapja a kliens és a szolgáltatás között.

authenticator: Hitelesítő adat, melyet a kliens a ticket bemutatásával párhuzamosan ad át a szolgáltatásnak. A kliens principal egyedi nevét és az időpontot tartalmazza a session kulccsal titkosítva.

Kerberos fogalmak / 2

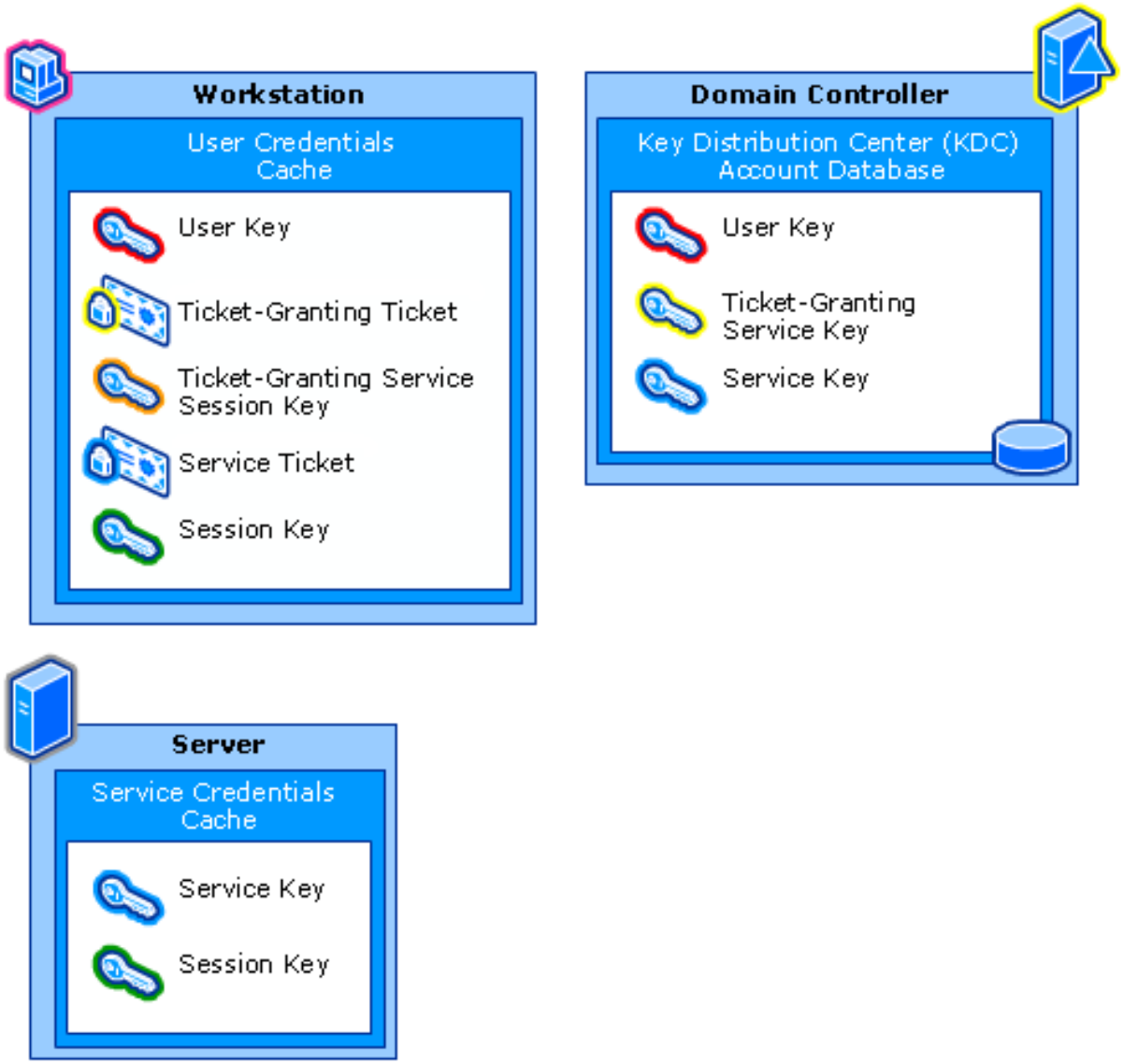
Key Distribution Center (KDC): Az adott tartomány autentikációs szolgáltatása, mely három részből áll: Database, Authentication Server és Ticket Granting Server.

Database: A tartomány prinicipal-jait, azok szimmetrikus kulcsát és a principal, valamint a ticket-ek érvényességi időit tartalmazza. (Titkosítva egy master kulccsal.)

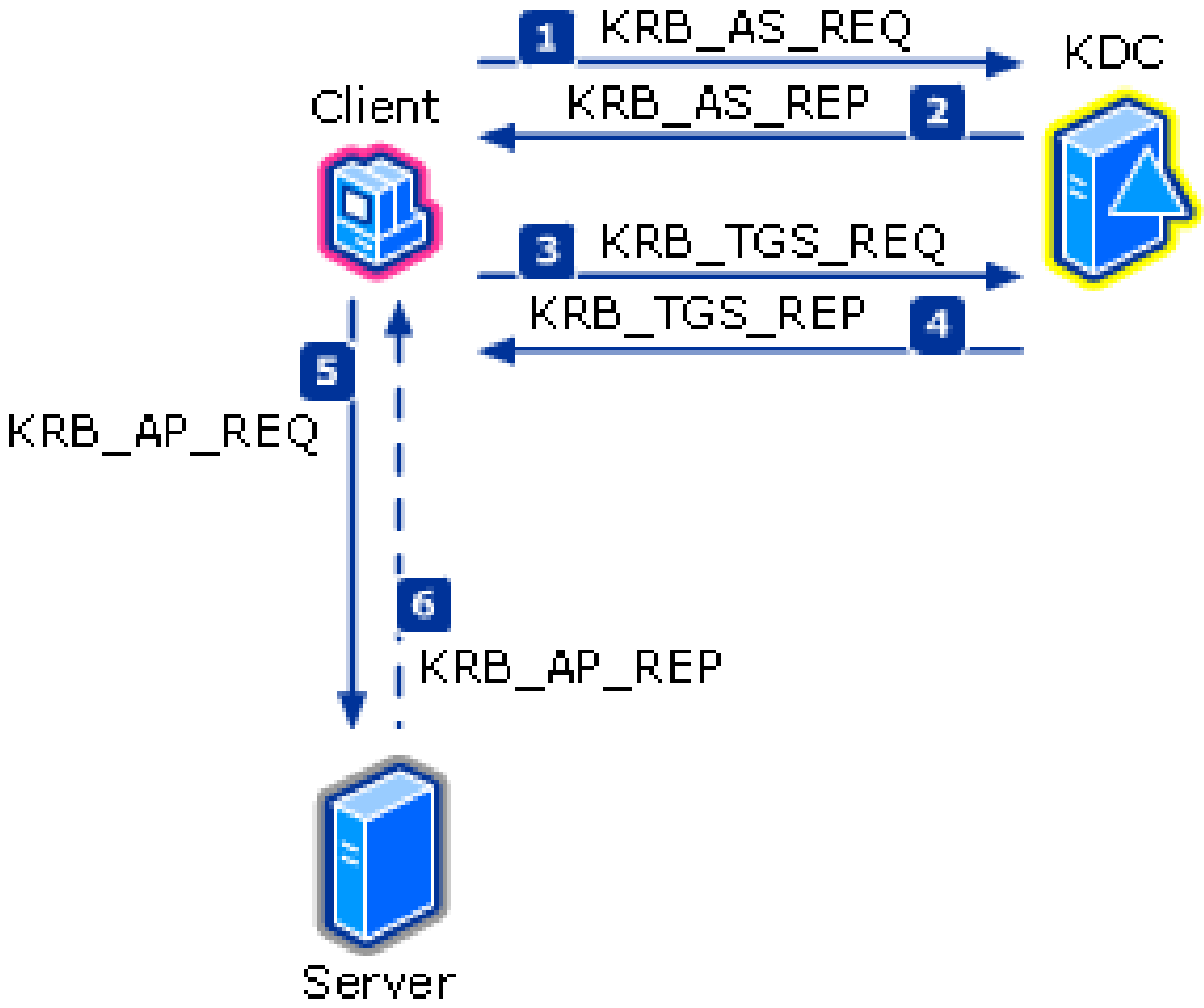
Authentication Server (AS): A még be nem jelentkezett felhasználó kezdeti autentikációját (jelszó ellenőrzését) végzi. A sikeres autentikáció eredménye egy Ticket Granting Ticket (TGT) átadása. A ticket-ek kiadása a TGT alapján történik, nem kell újra jelszót megadni.

Ticket Granting Server (TGS): A klienstől kapott érvényes TGT alapján átadja az adott szolgáltatáshoz kapcsolódó ticket-et.

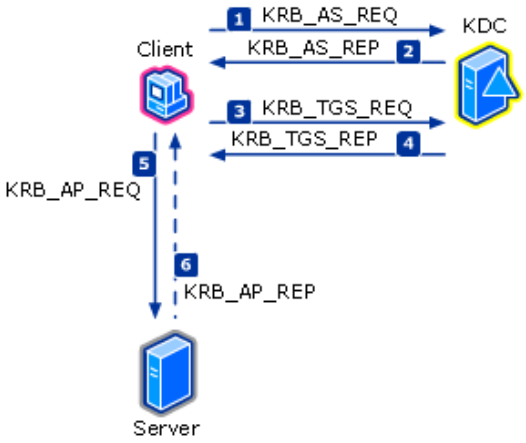
Kerberos adatok



Kerberos üzenetek

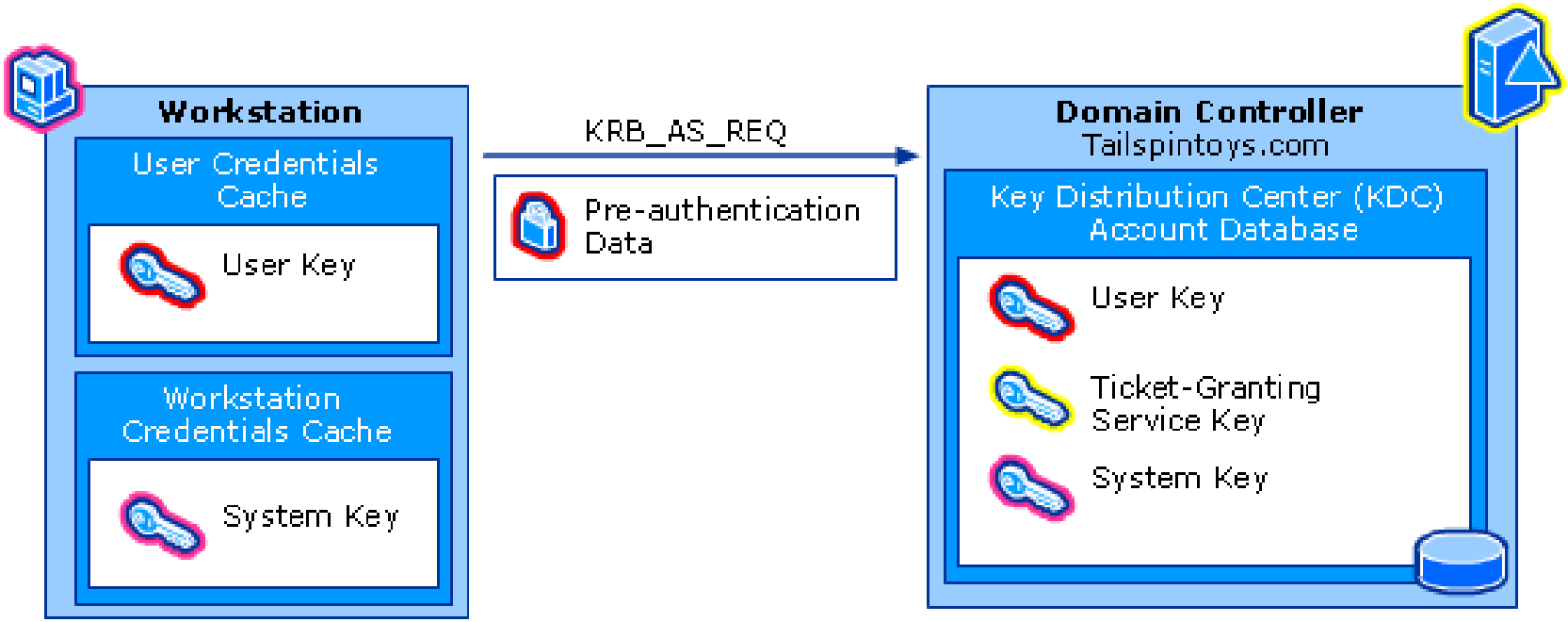


Kerberos autentikáció / 1

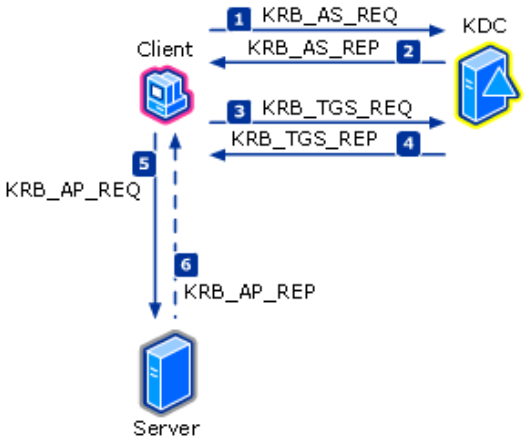


tartalma:

- kliens principal neve
- tartomány neve
- időpecsét titkosítva (User Key)

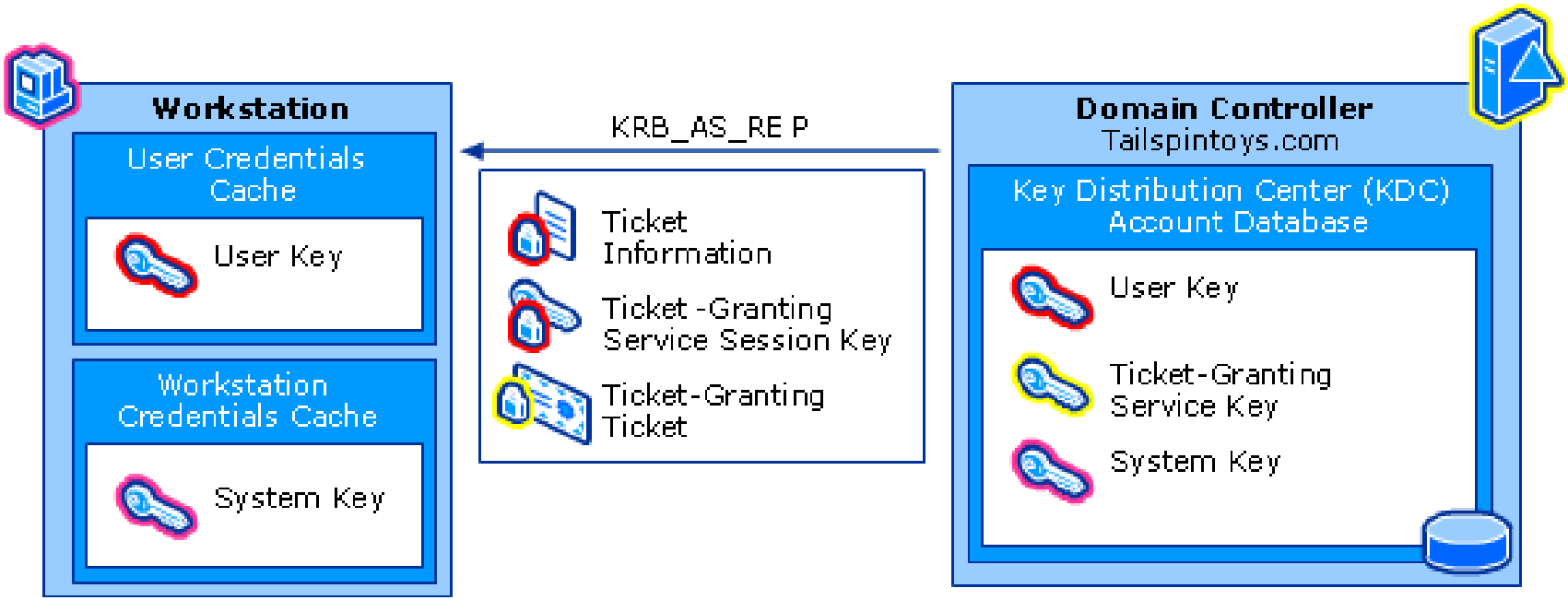


Kerberos autentikáció / 2

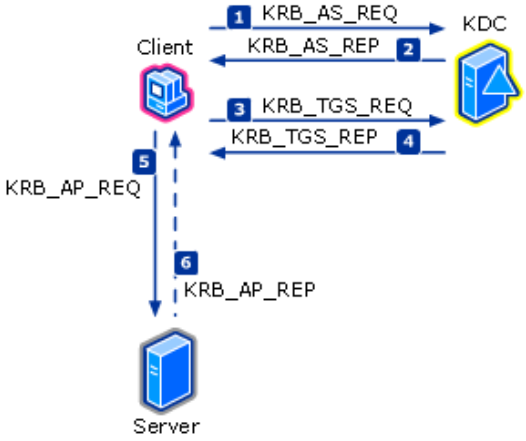


tartalma:

- TGT titkosítva (TGS Key)
- TGT részleges tartalma titkosítva (User Key) - érvényességi időadatok miatt
- TGS session kulcs titkosítva (User Key) - későbbi TGS kommunikációhoz

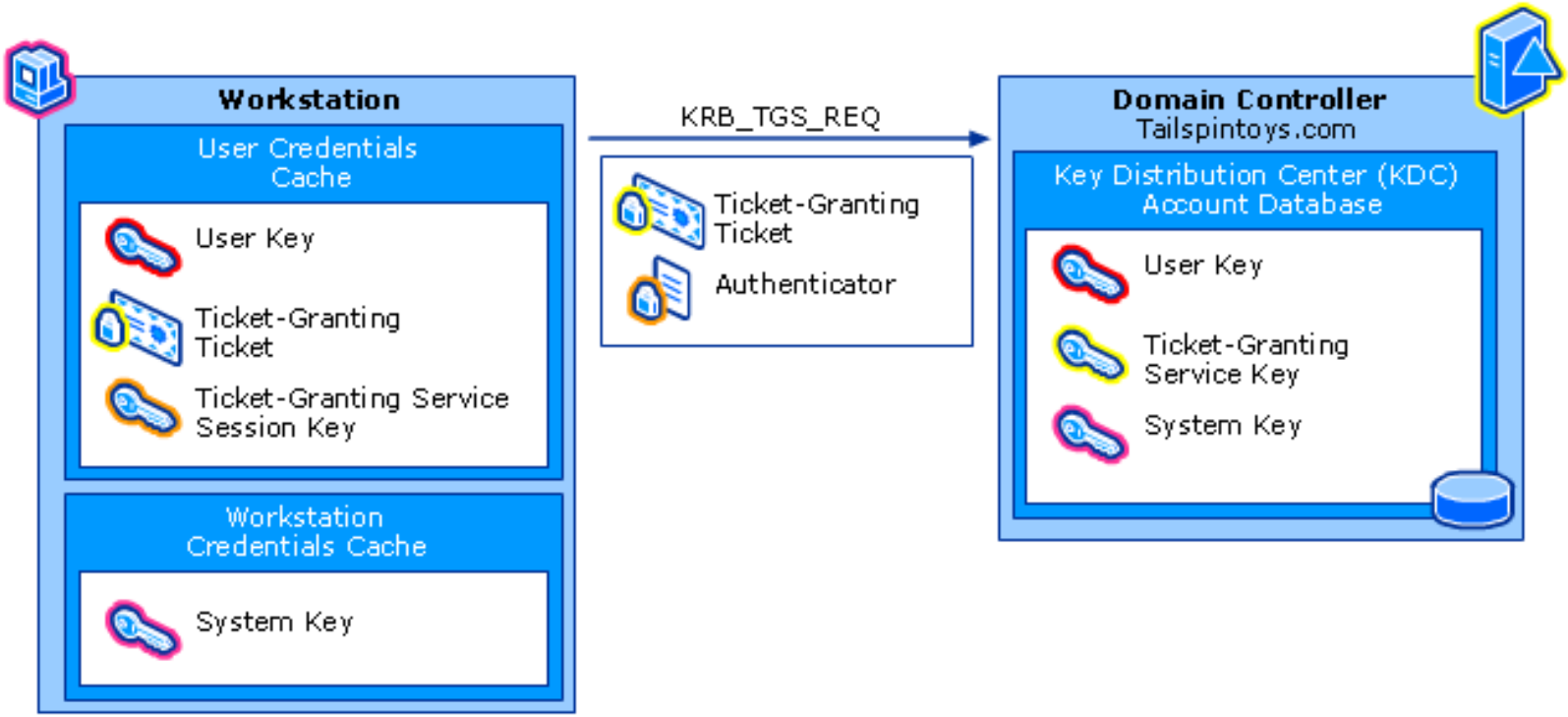


Kerberos workstation logon / 1

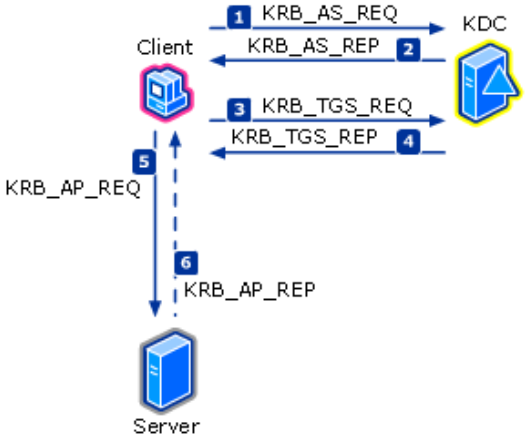


tartalma:

- workstation principal neve, tartomány neve
- TGT titkosítva (TGS Key) - ennek része a TGS session kulcs
- authenticator: user principal neve + időbélyegző titkosítva (TGS session key)

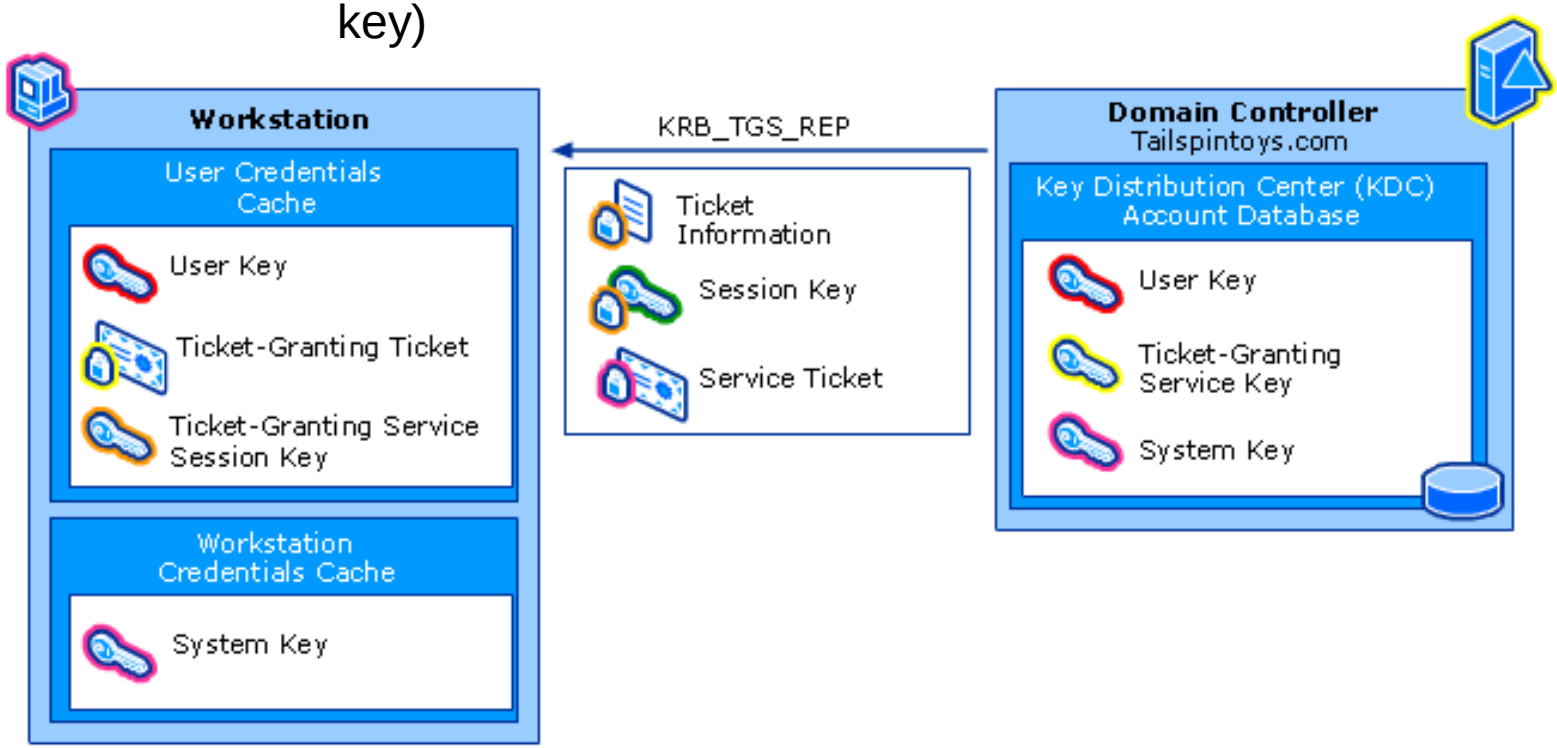


Kerberos workstation logon / 2

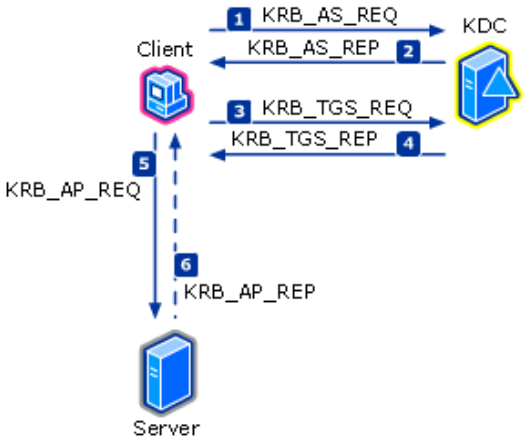


tartalma:

- Service ticket titkosítva (wokstation system key) - ennek része a user és a workstation közt megosztott session kulcs
- a Service ticket tartalma + a user és a workstation közt megosztott session kulcs titkosítva (TGS session key)



Kerberos workstation logon / 3



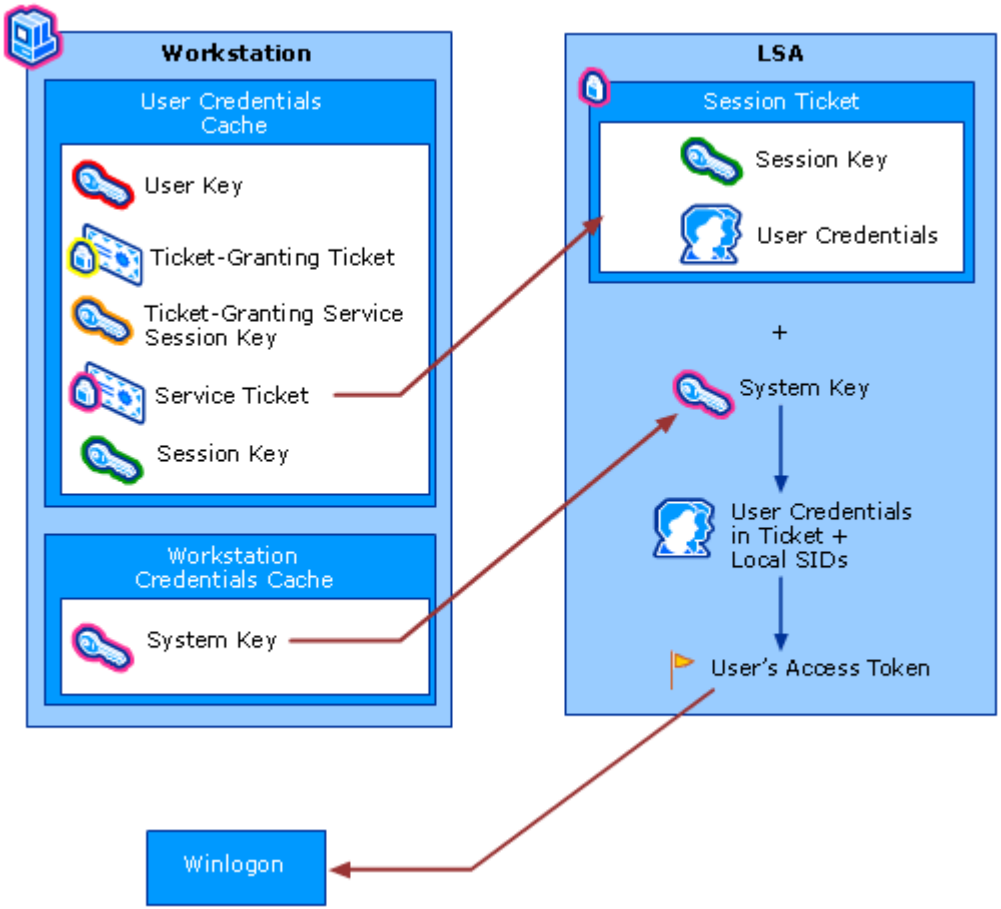
Local Security Authority feladata:

Service ticket dekódolása. Ez tartalmazza a User SID-jét: Security identifier (ez azonosítja a felhasználót, mert a név változhat).

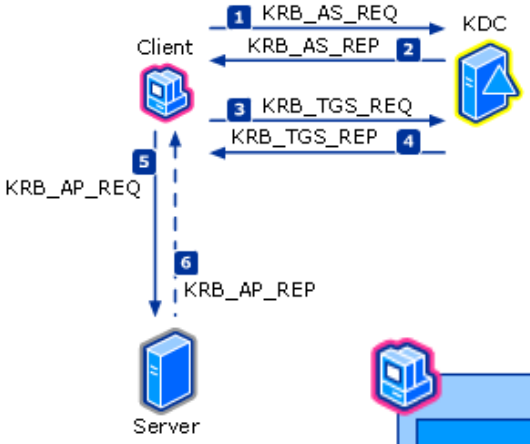
User jogosultságok (groups, privileges) lekérdezése

Access Token létrehozása, ami az adott felhasználó minden process-éhez hozzákapcsolódik

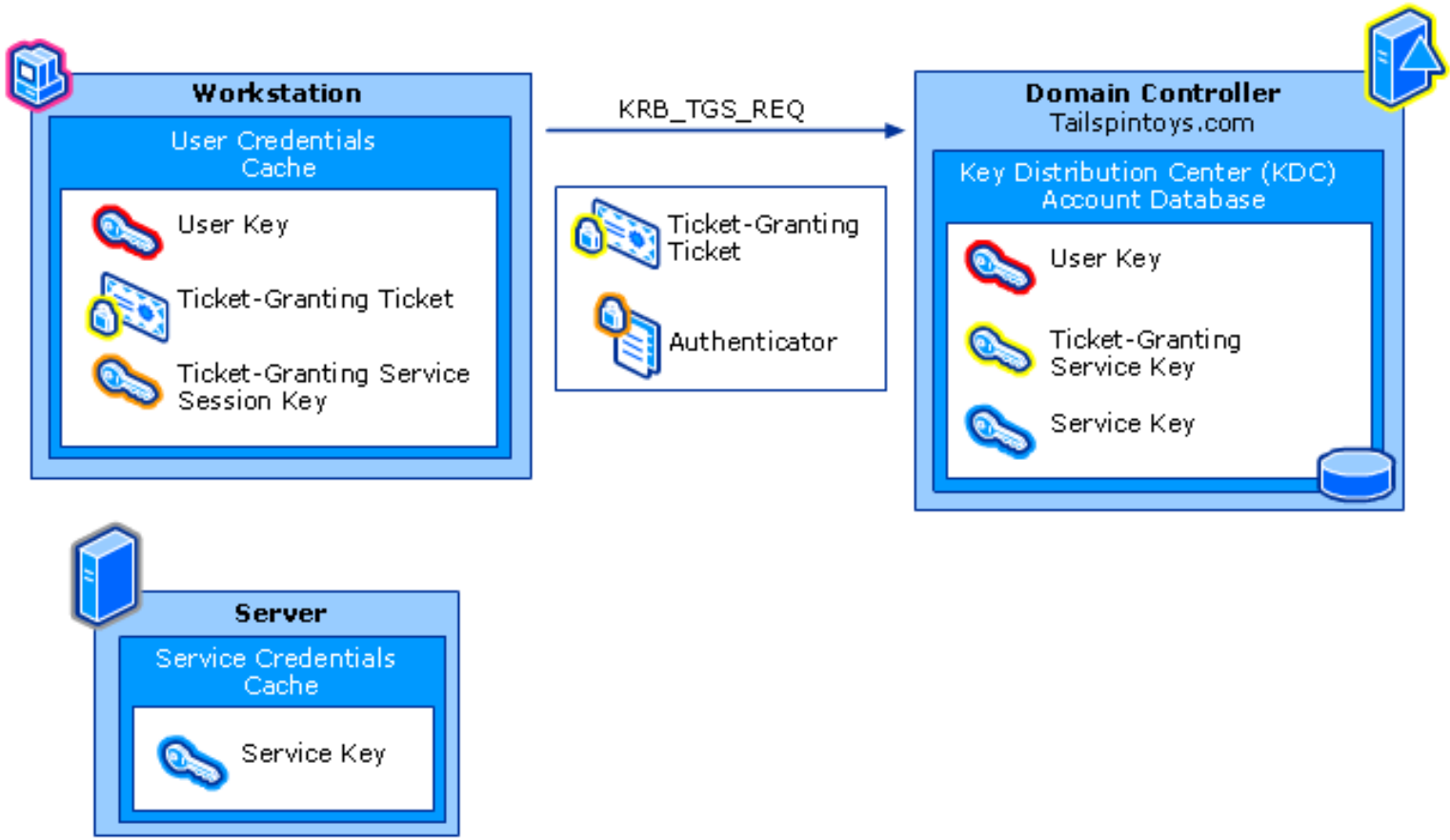
Windows shell indítása



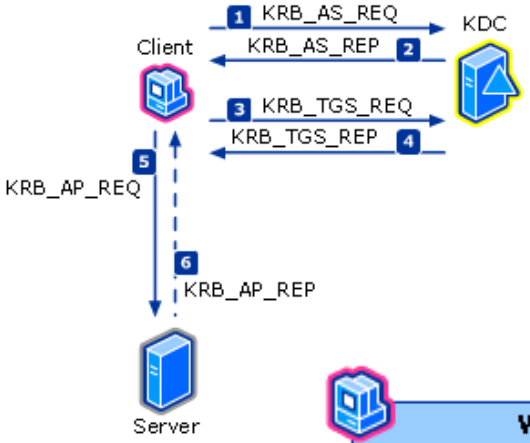
Kerberos szolgálatás logon / 1



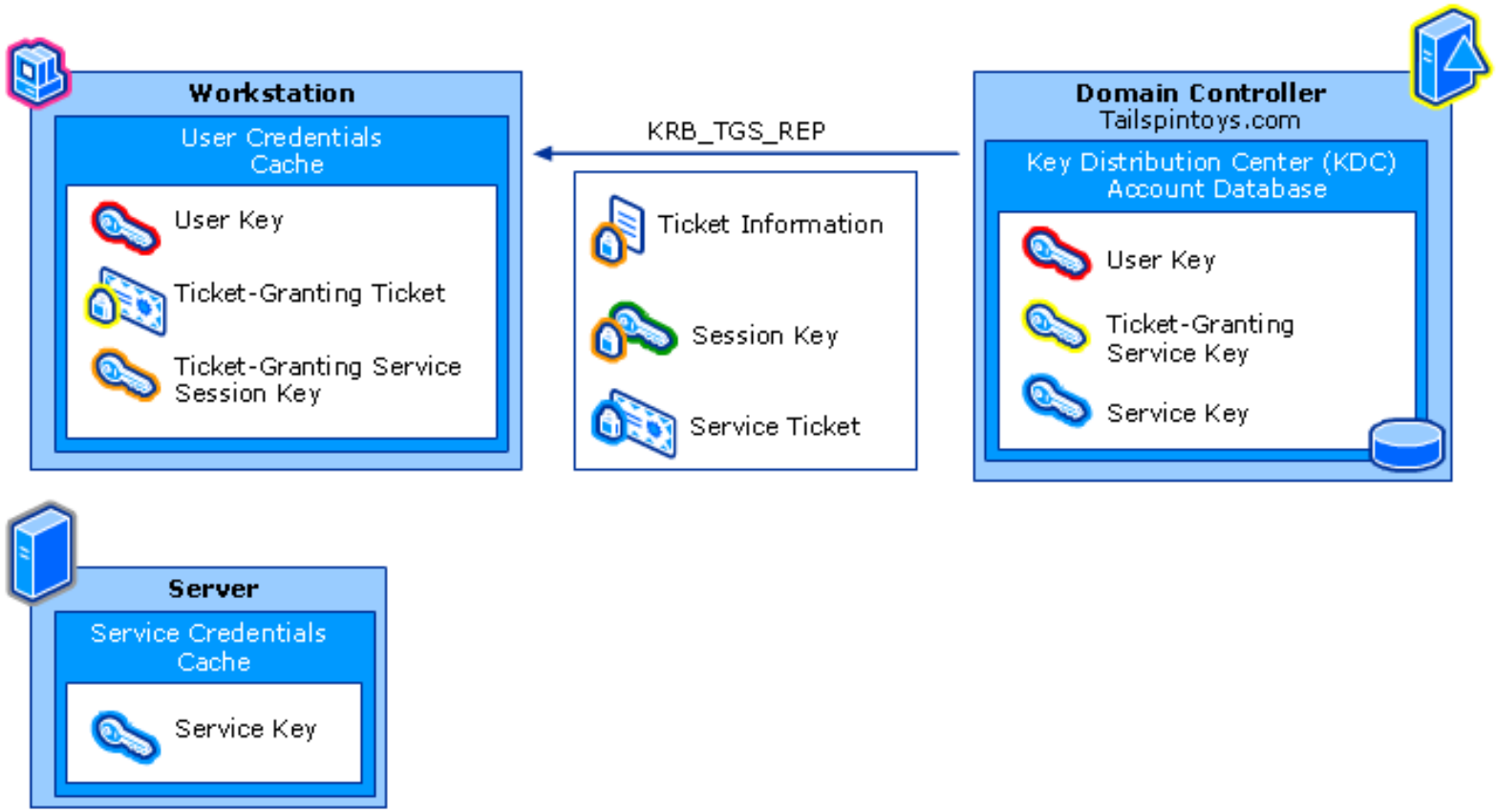
tartalma ugyanaz, mint a wokstation logon esetén, de a szolgáltatás principal neve és domain neve szerepel benne



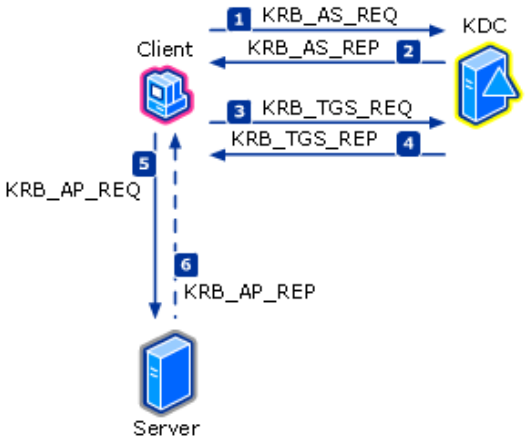
Kerberos szolgáltatás logon / 2



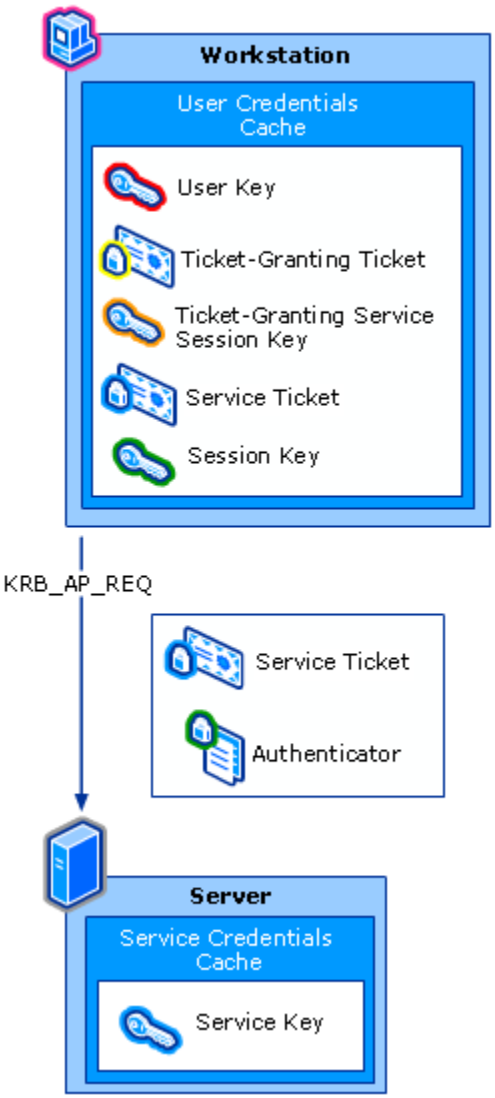
tartalma ugyanaz, mint a workstation logon esetén, de a ticket a szolgáltatás statikus kulcsával van titkosítva

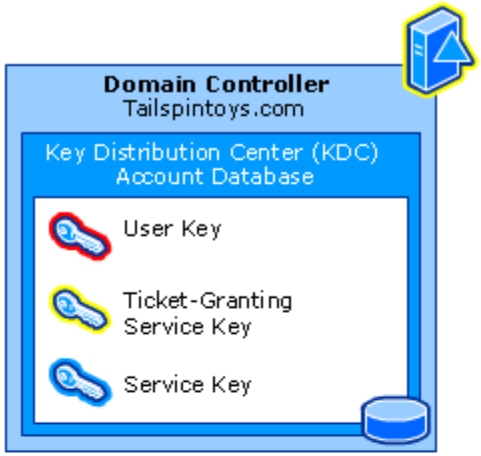


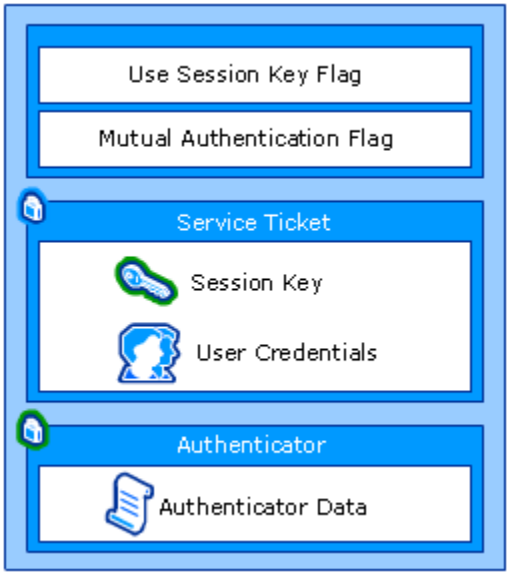
Kerberos szolgáltatás logon / 3



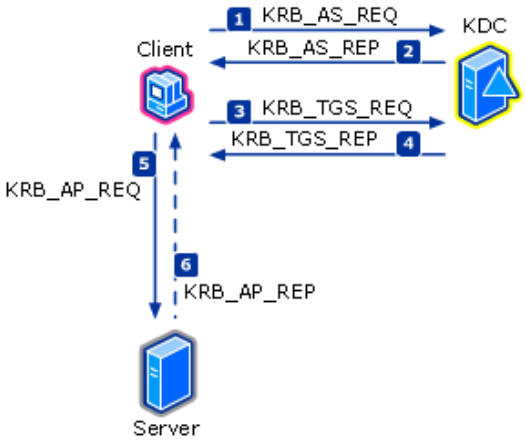
- Ticket information tartalmazza a flag-eket: titkosítás, szerver hitelesítés
- Service ticket tartalmazza a Service session kulcsot titkosítva (statikus Service key)
- authenticator: user principal neve + időbélyegző titkosítva (Service session key)







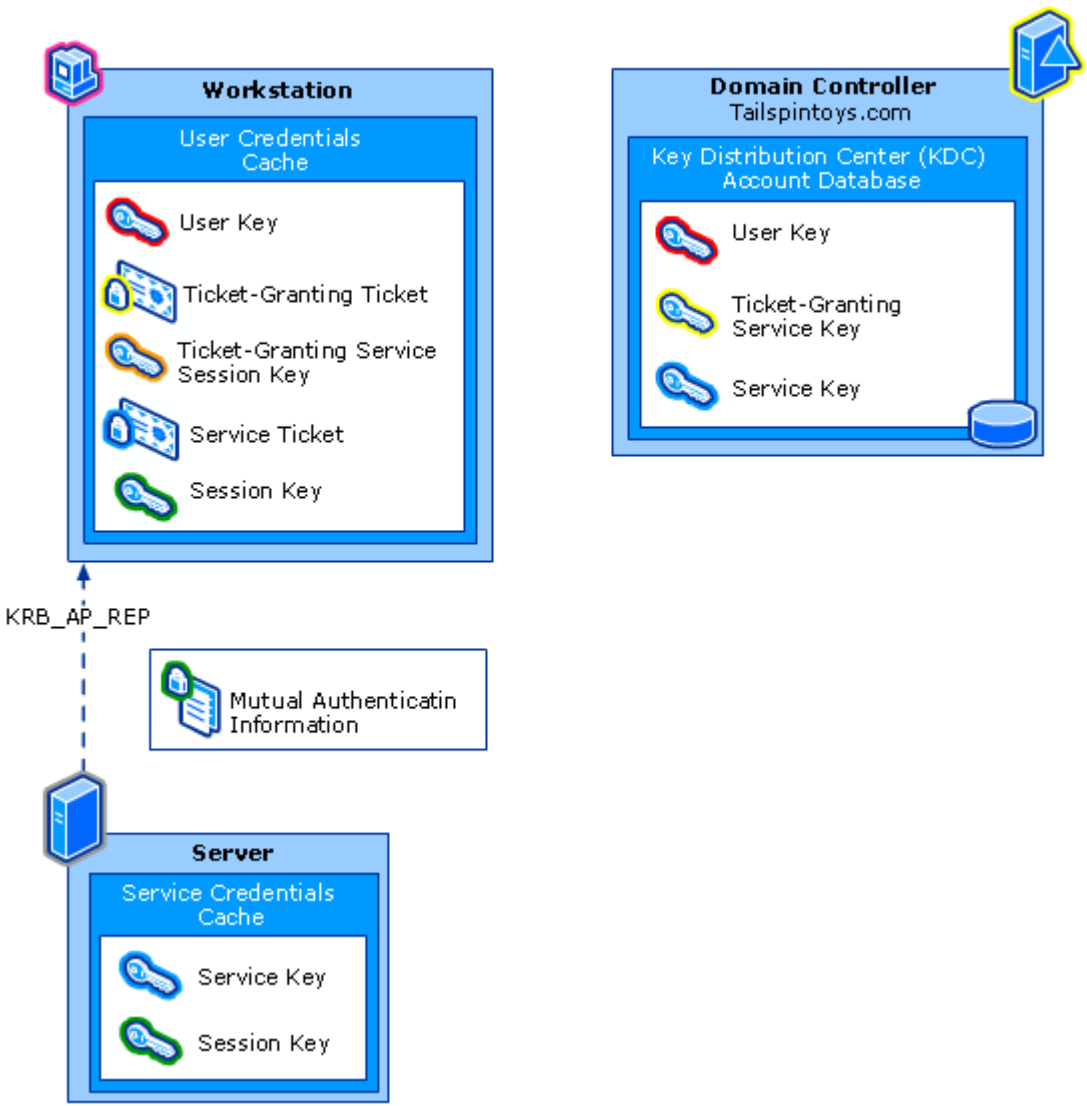
Kerberos szolgáltatás logon / 4



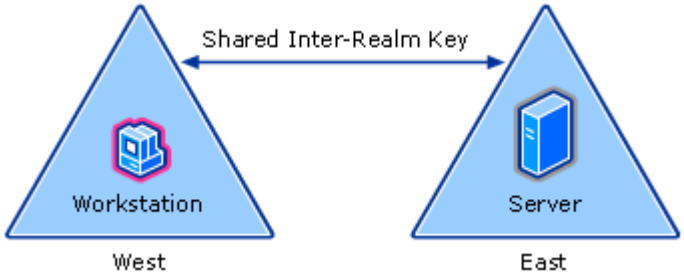
Opcionális lépés.

A KRB_AP_REQ-ben megadott időpecsétet titkosítja a Session kulccsal.

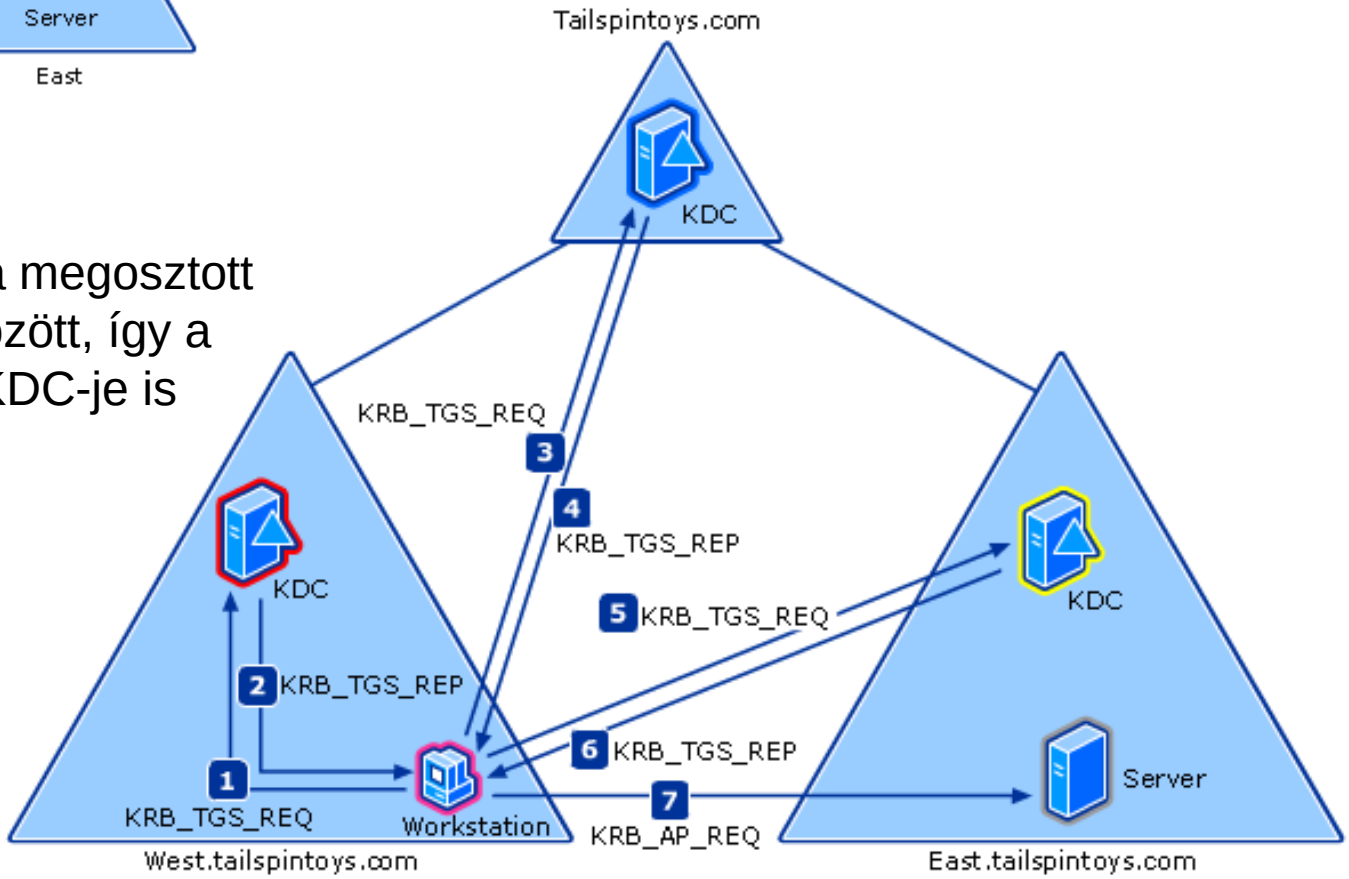
A kliens összeveti a két időpecsétet ellenérzéskor.



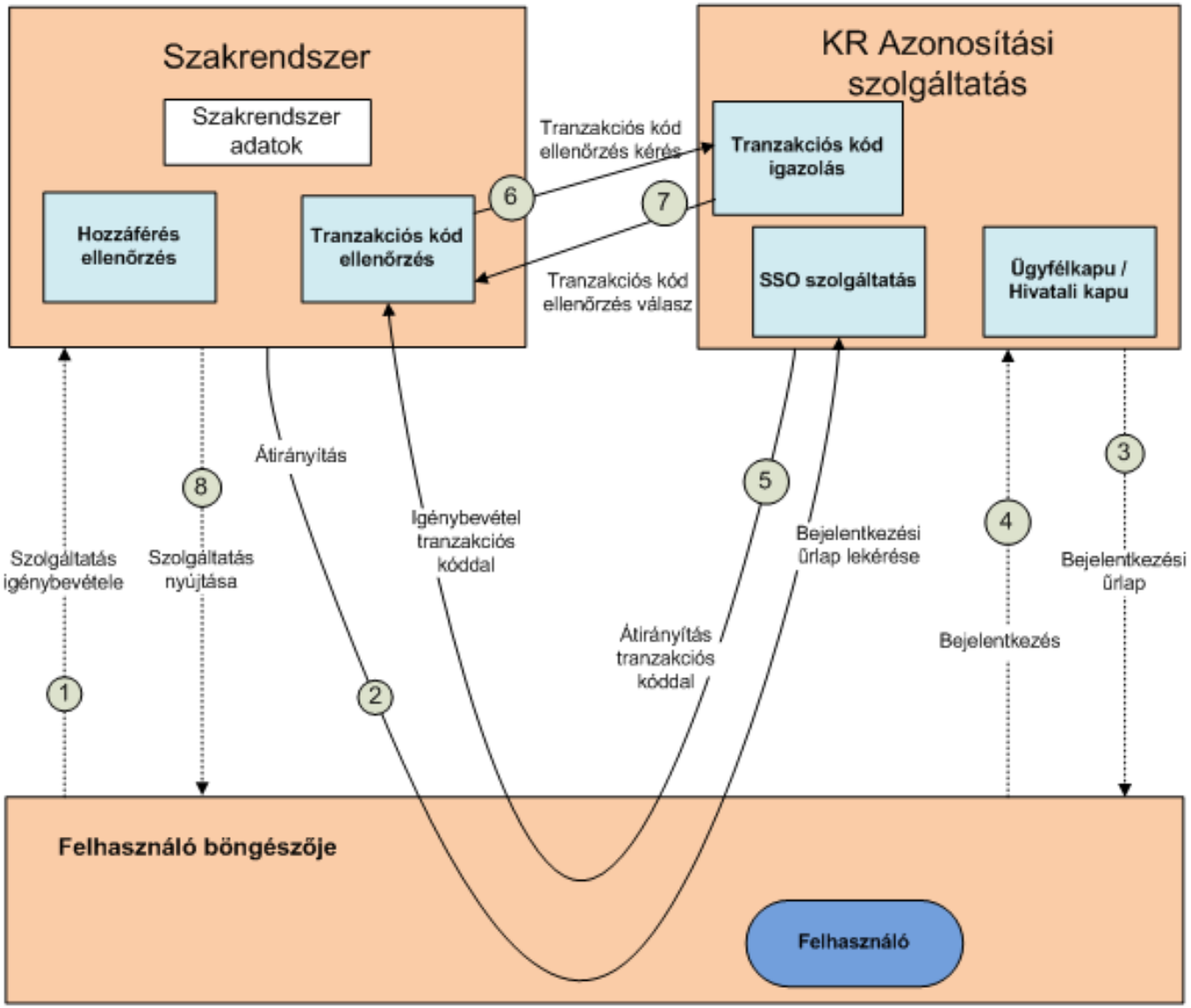
Tartományok közti autentikáció



A TGT titkosító kulcsa megosztott a két tartomány között, így a másik tartomány KDC-je is dekódolni tudja.



mo.hu (régi ügyfélkapu) SSO



HTTP alapok

HTTP alapok:

https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

HTTP cookie alapok:

https://en.wikipedia.org/wiki/HTTP_cookie

HTTP redirect alapok:

https://en.wikipedia.org/wiki/HTTP_302

https://en.wikipedia.org/wiki/HTTP_303

https://en.wikipedia.org/wiki/Meta_refresh

HTTP redirect

HTTP redirect status codes 3xx:

- 301 moved permanently
- 302 found (HTTP 1.0: temporary redirect volt, de megváltozott az értelme)
- 303 see other (HTTP 1.1, csak GET-tel)
- 307 temporary redirect (HTTP 1.1, ugyanazzal a metódussal)

```
HTTP/1.1 301 Moved Permanently
Location: http://www.example.org/
Content-Type: text/html
Content-Length: 174

<html>
<head>
<title>Moved</title>
</head>
<body>
<h1>Moved</h1>
<p>This page has moved to <a href="http://www.example.org/">http://www.example.org/</a>.</p>
</body>
</html>
```

mo.hu login / 1

Kérés:

```
GET https://magyarorszag.hu/ HTTP/1.1
Host: magyarorszag.hu
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; rv:51.0) Gecko/20100101 Firefox/51.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: hu-HU,hu;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

Válasz:

```
HTTP/1.1 200 OK
Date: Thu, 23 Feb 2017 07:44:01 GMT
Set-Cookie: JSESSIONID=3D97B54A31BC4812BF8D40932DE85787.portal3; Path=/; Secure
Content-Type: text/html;charset=UTF-8
Keep-Alive: timeout=15, max=100
Connection: keep-alive
Transfer-Encoding: chunked
```

mo.hu login / 2

Kérés:

```
GET https://gate.gov.hu/sso/ap/ApServlet?partnerid=mohu HTTP/1.1
Host: gate.gov.hu
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; rv:51.0) Gecko/20100101 Firefox/51.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: hu-HU,hu;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate, br
Referer: https://magyarorszag.hu/
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

Válasz:

```
HTTP/1.1 200 OK
Date: Thu, 23 Feb 2017 07:44:34 GMT
Set-Cookie: JSESSIONID=D0719722E5F6614A412D89E87153AB33.sso2; Path=/
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Content-Type: text/html;charset=UTF-8
Content-Length: 7293
Keep-Alive: timeout=15, max=100
Connection: keep-alive
```

mo.hu login / 3

Kérés:

```
POST https://gate.gov.hu/sso/ap/ApServlet HTTP/1.1
Host: gate.gov.hu
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; rv:51.0) Gecko/20100101 Firefox/51.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: hu-HU,hu;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate, br
Referer: https://gate.gov.hu/sso/ap/ApServlet?partnerid=mohu
Cookie: JSESSIONID=D0719722E5F6614A412D89E87153AB33.sso2
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 65

partnerid=mohu&target=&felhasznaloNev=          &jelszo=
```

Válasz:

```
HTTP/1.1 302 Moved Temporarily
Date: Thu, 23 Feb 2017 07:44:52 GMT
Content-Length: 0
Location: https://gate.gov.hu/sso/ap/ApServlet
Keep-Alive: timeout=15, max=100
Connection: keep-alive
Content-Type: text/plain; charset=UTF-8
```

mo.hu login / 4

Kérés:

```
GET https://gate.gov.hu/sso/ap/ApServlet HTTP/1.1
Host: gate.gov.hu
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; rv:51.0) Gecko/20100101 Firefox/51.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: hu-HU,hu;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate, br
Referer: https://gate.gov.hu/sso/ap/ApServlet?partnerid=mohu
Cookie: JSESSIONID=D0719722E5F6614A412D89E87153AB33.sso2
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

Válasz:

```
HTTP/1.1 200 OK
Date: Thu, 23 Feb 2017 07:44:52 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Content-Type: text/html;charset=UTF-8
Content-Length: 5212
Keep-Alive: timeout=15, max=99
Connection: keep-alive
```

```
<meta http-equiv="refresh" content="3;URL=../InterSiteTransfer?TARGET=&amp;PARTNER=mohu"/>
```

mo.hu login / 5

Kérés:

```
GET https://gate.gov.hu/sso/InterSiteTransfer?TARGET=&PARTNER=mohu HTTP/1.1
Host: gate.gov.hu
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; rv:51.0) Gecko/20100101 Firefox/51.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: hu-HU,hu;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate, br
Cookie: JSESSIONID=D0719722E5F6614A412D89E87153AB33.sso2
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

Válasz:

```
HTTP/1.1 302 Moved Temporarily
Date: Thu, 23 Feb 2017 07:44:56 GMT
Content-Length: 0
Location: https://ugyfelkapu.magyarorszag.hu/mohu_portlet_frame/auth/SSOLogin?SAMLart=J5xaw%2FsCqMi
Keep-Alive: timeout=15, max=98
Connection: keep-alive
Content-Type: text/plain; charset=UTF-8
```

```
portlet_frame/auth/SSOLogin?SAMLart=J5xaw%2FsCqMibaiJEarMO5bNtpJ2l1XccZC72MQamocQn62F9n59fA&TARGET=
```

mo.hu login / 6

Kérés:

```
GET https://ugyfelkapu.magyarorszag.hu/mohu_portlet_frame/auth/SSOLogin?SAMLart=J5xaw%2FsCgM
Host: ugyfelkapu.magyarorszag.hu
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; rv:51.0) Gecko/20100101 Firefox/51.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: hu-HU,hu;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate, br
Cookie: cssSize=0
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

Válasz:

```
HTTP/1.1 302 Moved Temporarily
Date: Thu, 23 Feb 2017 07:45:38 GMT
Set-Cookie: sso_auth=KmyK353jV3Efm/bmSe44Uv/fEgvNgA88M4iLLpokBFg=; Domain=magyarorszag.hu; Path=/; Secure
Set-Cookie: JSESSIONID=87EA18C81643BF5334D98798D460D3E7.portal3; Path=/; Secure
Location: https://ugyfelkapu.magyarorszag.hu
Content-Length: 0
Keep-Alive: timeout=15, max=100
Connection: keep-alive
Content-Type: text/plain; charset=UTF-8
```

mo.hu login / 7

Kérés:

```
GET https://ugyfelkapu.magyarorszag.hu/ HTTP/1.1
Host: ugyfelkapu.magyarorszag.hu
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; rv:51.0) Gecko/20100101 Firefox/51.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: hu-HU,hu;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate, br
Cookie: cssSize=0; sso_auth=KmyK353jV3Efm/bmSe44Uv/fEgvNgA88M4iLLpokBFg=; JSESSIONID=87EA18C81643BF
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

Válasz:

```
HTTP/1.1 302 Moved Temporarily
Date: Thu, 23 Feb 2017 07:45:38 GMT
Set-Cookie: JSESSIONID=87EA18C81643BF5334D98798D460D3E7.portal3; Path=/; Secure
pragma: no-cache
Cache-control: no-cache,no-store,must-revalidate
Location: https://ugyfelkapu.magyarorszag.hu/szolgaltatasok/tarhely/beerkezett
Content-Length: 0
Keep-Alive: timeout=15, max=99
Connection: keep-alive
Content-Type: text/plain; charset=UTF-8
```

mo.hu login / 8

Tranzakciós kód ellenőrzése:

A szakrendszerhez történő visszairányítás egyik paramétere a sikeres bejelentkezéskor generált, a bejelentkezés idején érvényes egyedi azonosító: a tranzakciós kód (SAMLart).

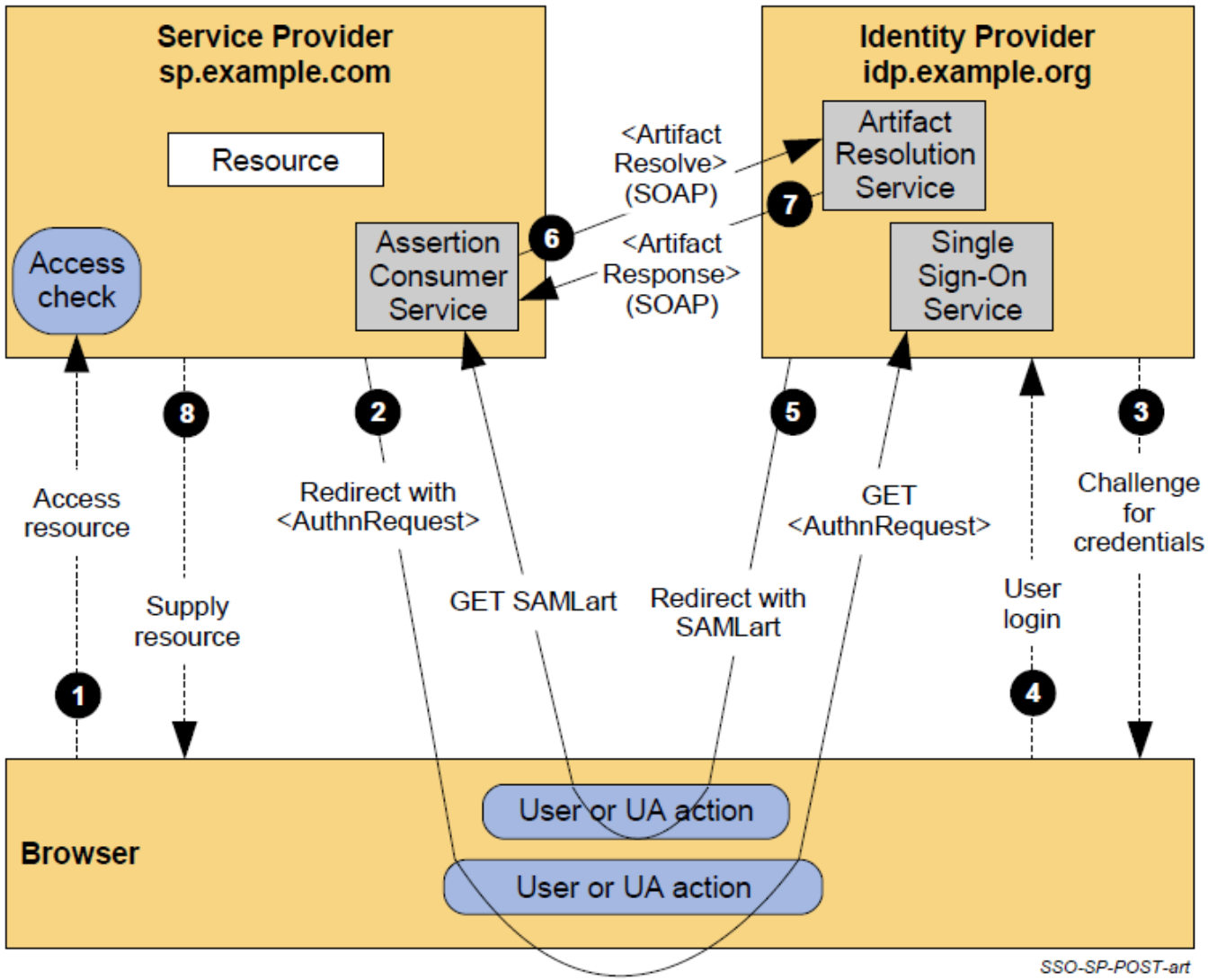
A szakrendszer ellenőrzi (szerver-szerver kommunikáció), hogy az adott felhasználó valóban sikeresen belépett a gate.gov.hu-n.

A tranzakciós kód ellenőrzési kérésre küldött válaszüzenetben a szakrendszer hozzájut az ügyfél ún. kapcsolati kódjához is, amely abban a rendszerben egyedi azonosítóként működik.

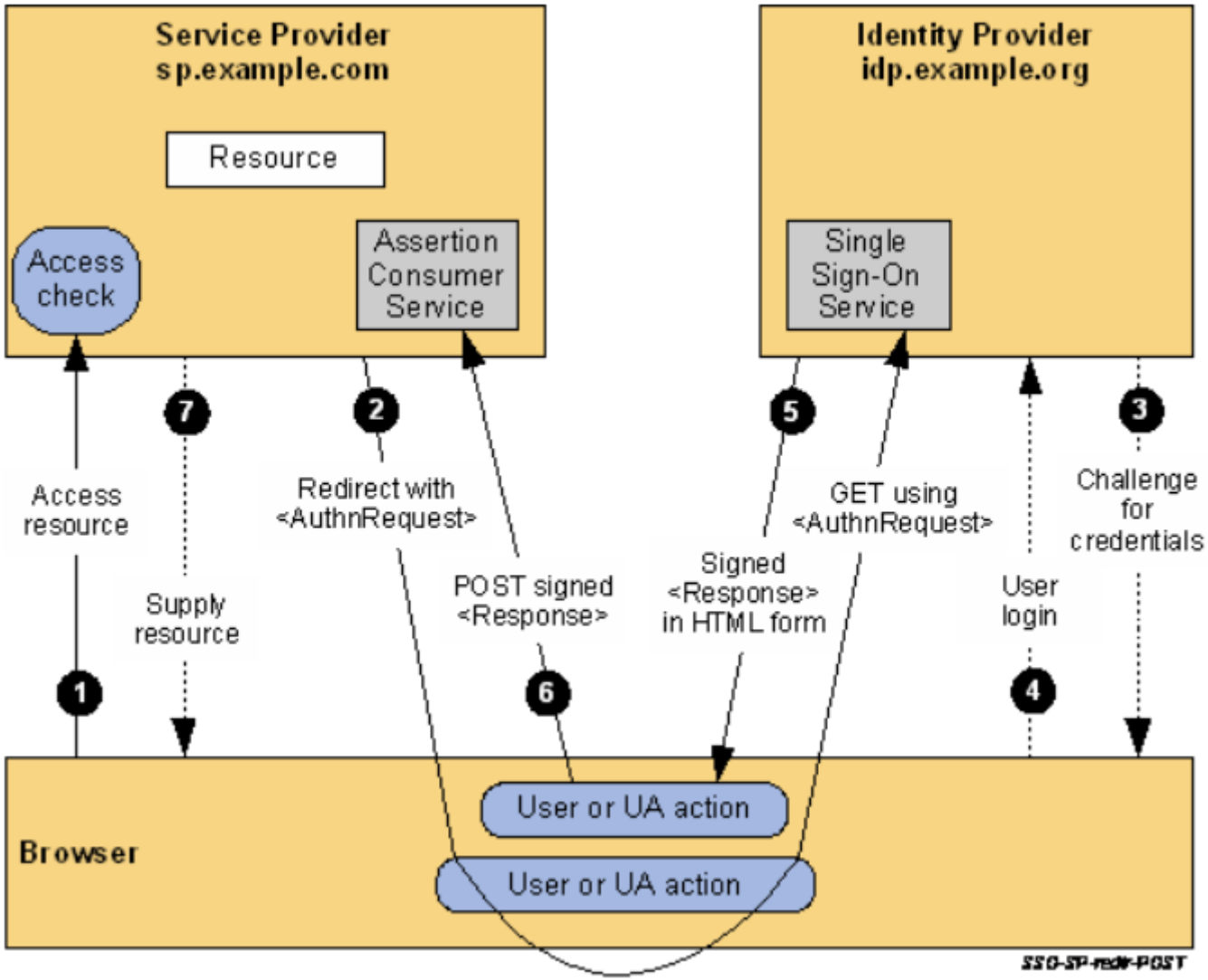
Viszontazonosítás:

A kapcsolati kód használatával a szakrendszer ellenőrizheti az adott felhasználó gate.gov.hu-n tárolt személyes adatait.

Security Assertion Markup Language (SAML) V2.0



SAML V2.0 a KAÜ esetében



kau.gov.hu login

Kérés:

```
POST https://kau.gov.hu/proxy/saml/authnrequest HTTP/1.1
Host: kau.gov.hu
Connection: keep-alive
Content-Length: 9928
Cache-Control: max-age=0
Origin: https://nyilvantarto.hu
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.
Content-Type: application/x-www-form-urlencoded
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Referer: https://nyilvantarto.hu/ugyseged/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,hu-HU;q=0.8,hu;q=0.6,en;q=0.4

SAMLRequest=PD94bWwgdmVyc2lvbjOiMS4wIiBlbmNvZGluZzOiVVRGLTgiPz4KPHNhbWwycDpBdXRoblJlcXVl%0D%0Ac3
```

kau.gov.hu login

SAMLRequest tartalma:


```
▼<saml2p:AuthnRequest xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 AssertionConsumerServiceURL="https://www.nyilvantarto.hu/ugyseged/"
 Destination="https://kau.gov.hu/proxy/saml/authnrequest?lang=hu" ID="ID_e70459e4-
 88ec-45d6-810e-811b31556d55" IssueInstant="2017-03-09T15:48:19.772Z"
 ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST" Version="2.0">
   <saml2:Issuer
   xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">wu.kekkh.gov.hu</saml2:Issuer>
  ▶<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">...</ds:Signature>
  ▼<saml2p:Extensions>
    ▼<saml2:AttributeStatement xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">
      ▶<saml2:Attribute Name="urn:eksz.gov.hu:1.0:ony:adatigenylo"
       NameFormat="urn:eksz.gov.hu:1.0:ony:adatigenylo">...</saml2:Attribute>
      ▶<saml2:Attribute Name="urn:eksz.gov.hu:1.0:ony:cel"
       NameFormat="urn:eksz.gov.hu:1.0:ony:cel">...</saml2:Attribute>
      ▶<saml2:Attribute Name="urn:eksz.gov.hu:1.0:ony:felhasznalo"
       NameFormat="urn:eksz.gov.hu:1.0:ony:felhasznalo">...</saml2:Attribute>
      ▶<saml2:Attribute Name="urn:eksz.gov.hu:1.0:ony:jogalap"
```

kau.gov.hu login

SAMLRequest tartalma:

```
▼<saml2:Attribute Name="urn:eksz.gov.hu:1.0:ony:szervezetkod"
 NameFormat="urn:eksz.gov.hu:1.0:ony:szervezetkod">
   <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:type="xs:string">KEKKH_WU</saml2:AttributeValue>
 </saml2:Attribute>
▼<saml2:Attribute Name="urn:eksz.gov.hu:1.0:4t"
 NameFormat="urn:eksz.gov.hu:1.0:4t">
   <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:nil="true" xsi:type="xs:string"/>
 </saml2:Attribute>
▶<saml2:Attribute Name="urn:eksz.gov.hu:1.0:felhasznalo_azonosito"
 NameFormat="urn:eksz.gov.hu:1.0:felhasznalo_azonosito">...</saml2:Attribute>
</saml2:AttributeStatement>
</saml2p:Extensions>
<saml2p:NameIDPolicy AllowCreate="false" Format="urn:oasis:names:tc:SAML:2.0:nameid-
format:persistent" SPNameQualifier="urn:eksz.gov.hu:1.0:rnytkk"/>
▼<saml2:Conditions xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">
 ▼<saml2:AudienceRestriction>
   <saml2:Audience>urn:eksz.gov.hu:1.0:azonositas:kau:1</saml2:Audience>
 </saml2:AudienceRestriction>
```

kau.gov.hu login

AuthnRequest értelme:

A szakrendszer által, a felhasználó számára létrehozott és a KAÜ felé továbbítandó SAML v2.0 szabvány szerinti AuthnRequest üzenet tartalmazza az adott szakrendszerhez való hozzáférés feltételeit:

- végre kell hajtani a megfelelő felhasználó-hitelesítést és
- a kért típusú azonosítóadat lekérdezését.

A sikeres kommunikációhoz szükség van:

- a kérést küldő szakrendszer címére (akinek a válasz visszaküldésre kerül),
- egy egyedi tranzakció azonosítóra,
- az azonosítási szolgáltatás meghatározására.

kau.gov.hu login

AuthnRequest alá van írva (XML aláírás szabványnak megfelelően):

```
▼<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
  ▼<ds:SignedInfo>
     <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
     <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
    ▼<ds:Reference URI="#ID_e70459e4-88ec-45d6-810e-811b31556d55">
      ▼<ds:Transforms>
         <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
         signature"/>
        ▼<ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
           <ec:InclusiveNamespaces xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#"
           PrefixList="xs"/>
         </ds:Transform>
       </ds:Transforms>
       <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
       <ds:DigestValue>QsDRNNAVqdssgp5gJfbf0sf3Bjk=</ds:DigestValue>
     </ds:Reference>
   </ds:SignedInfo>
  ▼<ds:SignatureValue>
     1vRmUCdYvLrsgcbh01z5BzJdltheIgDXDAKYpjki2Thz8IB4JXt6MG0sVpIW4uBpCydjAcyh9JsAe6wD
   </ds:SignatureValue>
```

kau.gov.hu login

AuthnRequest alá van írva (XML Signature szabványnak megfelelően):


```
<ds:KeyInfo>
  <ds:X509Data>
    <ds:X509Certificate>
      MIIHXjCCBkagAwIBAgIIbMyQynJvmtkwDQYJKoZIhvcNAQELBQAwgZYxCzAJBgNVBAYTAkhVMRE
      DwYDVQQHDAhCdWRhcGVzdDE8MDoGA1UECgwzTklTWiBOZW16ZXRpIEluZm9rb21tdW5pa8OhY2n
      s3MgU3pvbGfDoWx0YXTDsyBacnQuMTYwNAYDVQQDDC1NaW7FkXPDrXRldHQgVGFuw7pzw610dsO
      bnlraWFkw7MgdjIgLSBHT1YgQ0EwHhcNMTYwOTI4MDczNzA1WhcNMTcwOTI4MDczNzA1WjCBszE
      MAkGA1UEBhMCSFUxETAPBgNVBAcMCEJ1ZGFwZXN0MQ8wDQYDVQQKDAZLRUsgS0gxGTAXBgNVBAU
      EERPMjAxMzEyMDItMURPNzcxNTAzBgNVBAMMLFdlYmVzIMOcZ3lzZWfDqWQgU1AgQWzDocOtcsO
      IHRhbsO6c8OtdHbDoW55MQ0wCwYDVQQRDAQxMDk0MR8wHQYDVQQJDBZCYWzDoXpzIELDqWxhIHV
      Y2EgMzUuMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA6GMWHgEVNom/+S9kmkTreGR
      EUjbMQ0jwpyGtrjPQZ5rMHEDVu3VgoJX4bl2yCq/Qz0w02Z52XhnE1ZA5ahnnLTU9PlZyNrZb8/
      zd/Wo5w6kM7jf9yAR+h+yrxNqmMNkeHPVck4PYQ8qJ1c1Q6dgH2zrA9Iss4HmcBICcfvUEHijdj
      Lhd1flMJabDVPfYCkYbxXmPadUNXyM6Bf93oJqGWbRNMLWhPWqNsyz0mlsfdAeHcsMrgSGoQ5bd
      HhKAS9K13V5BU2DGZB4DfkQmTyNFnxbfdJeI2tmUTDBYFY135a3DylIFANJ/+iGYlaIzoiQfHqg
      h8S+yvhAutjc7wIDAQABo4IDjzCCA4swdQYIKwYBBQUHAQEEaTBnMDcGCCsGAQUFBzAChitodHR
      Oi8vcWNhLmhpdGVsZXMuZ292Lmh1L2Nlci9HT1ZDQS1RdjIuY2VyMCwGCCsGAQUFBzABhiBodHR
      Oi8vcW9jc3AuaGl0ZWxlcy5nb3YuaHUvb2NzcDAdBgNVHQ4EFgQUcSyCc3UWetOGLDnM9ZE8nMT
      fDQwDAYDVR0TAQH/BAIwADAfBgNVHSMEGDAWgBT0ozuPKXoxKM9ryIb4JqjL3SeIgDBIBggrBgE
```

kau.gov.hu login

Sikeres belépést követő visszairányítás:

```
POST https://www.nyilvantarto.hu/ugyseged/ HTTP/1.1
Host: www.nyilvantarto.hu
Connection: keep-alive
Content-Length: 32102
Cache-Control: max-age=0
Origin: https://kau.gov.hu
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/5
Content-Type: application/x-www-form-urlencoded
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Referer: https://kau.gov.hu/proxy/saml/response
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,hu-HU;q=0.8,hu;q=0.6,en;q=0.4
Cookie: _ga=GA1.2.1976521315.1489074501; _gat=1

SAMLResponse=PD94bWwgdmVyc2lvbjOiMS4wIiBlbmNvZGluZzOiVVRGLTgiPz48c2FtbDJwOlJlc3BvbnNlIHht%0D%0A
OAbmNlIFVSSTOiI18yNDBlNTQOYzE3MDBjZmQzNWJjYWUyOWJhODVlYzhhYSIvPgogICAgICAgICAg%0D%0AICAgICA8L3h
```

kau.gov.hu login

SAMLResponse tartalma (aláírva, titkosítva):

```
▼<saml2p:Response xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol" Destination="kau.gov.hu"
 ID="ID_8ccf1ab6-3d90-4cf7-af62-8b8057148a22" InResponseTo="ID_51a722c5-5f13-4a05-ae9b-f63f61e8a1df"
 IssueInstant="2017-03-09T15:48:39.552Z" Version="2.0">
   <saml2:Issuer xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">idp.gov.hu</saml2:Issuer>
 ▶<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">...</ds:Signature>
 ▼<saml2p:Status>
     <saml2p:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
   </saml2p:Status>
 ▶<saml2:EncryptedAssertion xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">...
   </saml2:EncryptedAssertion>
 </saml2p:Response>
```

kau.gov.hu login

Titkosított adat és titkosított szimmetrikus kulcs (XML Encryption szabvány):

```
▼<saml2p:Response xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol" Destination="kau.gov.hu"
 ID="ID_8ccf1ab6-3d90-4cf7-af62-8b8057148a22" InResponseTo="ID_51a722c5-5f13-4a05-ae9b-f63f61e8a1df"
 IssueInstant="2017-03-09T15:48:39.552Z" Version="2.0">
   <saml2:Issuer xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">idp.gov.hu</saml2:Issuer>
 ▶<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">...</ds:Signature>
 ▶<saml2p:Status>...</saml2p:Status>
 ▼<saml2:EncryptedAssertion xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">
   ▶<xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
    Id="_ea58bcbe55e27d99c65cea3b60314105" Type="http://www.w3.org/2001/04/xmlenc#Element">...
    </xenc:EncryptedData>
   ▶<xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
    Id="_76d8e1fcf52e45e7cecaa80801ec325b" Recipient="ID_2481511d-bf02-43c0-bd67-56abc1d8103f">...
    </xenc:EncryptedKey>
  </saml2:EncryptedAssertion>
 </saml2p:Response>
```

kau.gov.hu login

Titkosított adat:


```
<xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
Id="_ea58bcbe55e27d99c65cea3b60314105" Type="http://www.w3.org/2001/04/xmlenc#Element">
  <xenc:EncryptionMethod xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
  Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
  <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#" Id="ID_7298dcf3-0f6c-4c7d-b8a1-
  c5c57562a2dc">
    <ds:RetrievalMethod Type="http://www.w3.org/2001/04/xmlenc#EncryptedKey"
    URI="#_76d8e1fcf52e45e7cecaa80801ec325b"/>
  </ds:KeyInfo>
  <xenc:CipherData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
    <xenc:CipherValue>
      v9/Y89veXDzB0uzAUra+2ejuPrZYSb4rNjDEM+nf12VKWigU+FbVgI5rI4e/d5wA/jZpcJAGSQzOQTOOxpEnZCzX
    </xenc:CipherValue>
  </xenc:CipherData>
</xenc:EncryptedData>
```

kau.gov.hu login

Titkosított szimmetrikus kulcs:


```
▼<xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
 Id="_76d8e1fcf52e45e7cecaa80801ec325b" Recipient="ID_2481511d-bf02-43c0-bd67-56abc1d8103f">
   <xenc:EncryptionMethod xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
   Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
  ▼<ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#" Id="ID_2481511d-bf02-43c0-bd67-
   56abc1d8103f">
    ▼<ds:X509Data>
      ▼<ds:X509Certificate>
          MIIHOjCCBiKgAwIBAgIIXoA3sQ3Z4WIwDQYJKoZIhvcNAQELBQAwgbQxCzAJBgNVBAYTAkhVMREw
          DwYDVQQHDAhCdWRhcGVzdDE8MDoGA1UECgwzTklTWiBOZW16ZXRpIEluZm9rb21tdW5pa8OhY2nD
          s3MgU3pvbGfDoWx0YXTDsyBacnQuMVQwUgYDVQQDDEtUaXRrb3PDrXTDsyBUYW7DunPDrXR2w6Fu
          eWtpYWTDsyAtIEtvcm3DoW55emF0aSBIaXRlbGVzw610w6lzIFN6b2xnw6FsdGF0w7MwHhcNMTUw
          NjMwMTMwMzMwWhcNMTcwNjI5MTMwMzMwWjCBuTELMAkGA1UEBhMCSFUxETAPBgNVBAcMCEJ1ZGFw
          ZXN0MTwwOgYDVQQKDDNOSVNaIE5lbXpldGkgSW5mb2tvbW11bmlrw6FjacOzcyBTem9sZ8OhbHRh
          dMOzIFpydC4xGjAYBgNVBAUTEURPMjAxMzA5MTctMURPMTMwMRMwEQYDVQQDDAprYXUuZ292Lmh1
          MQ0wCwYDVQQRDAQxMDgxMRkwFwYDVQQJDBBDc29rb25haSB1dGNhIDMuMIIBIjANBgkqhkiG9w0B
          AQEFAAOCAQ8AMIIBCgKCAQEAmeF7PJ9Q8aV526x9ShxS5G5O8dOXdgYoB3XYbxFobekxT+IE5Aa0
          esTHJtpWjSwUwIiTBFVvAr7t9xjf1Rw/inUOt1b/ig24awBofxaJlzyMcIQmZy5LQJIZPovDTd5k
          IUq502nyuzIP36tva7cVJZfALImQBU+FbMSiY69JKHSa4DeUauCeY27RAg6U03bERArgfGaYqFsL
          DGaW8HVps2Fr19K02RRSx7v01fcqydg0uRiyfBU+PRvXjRC95OdPMHVPi7A7W0CeD2QpUMA0fT1Q
```

kau.gov.hu login

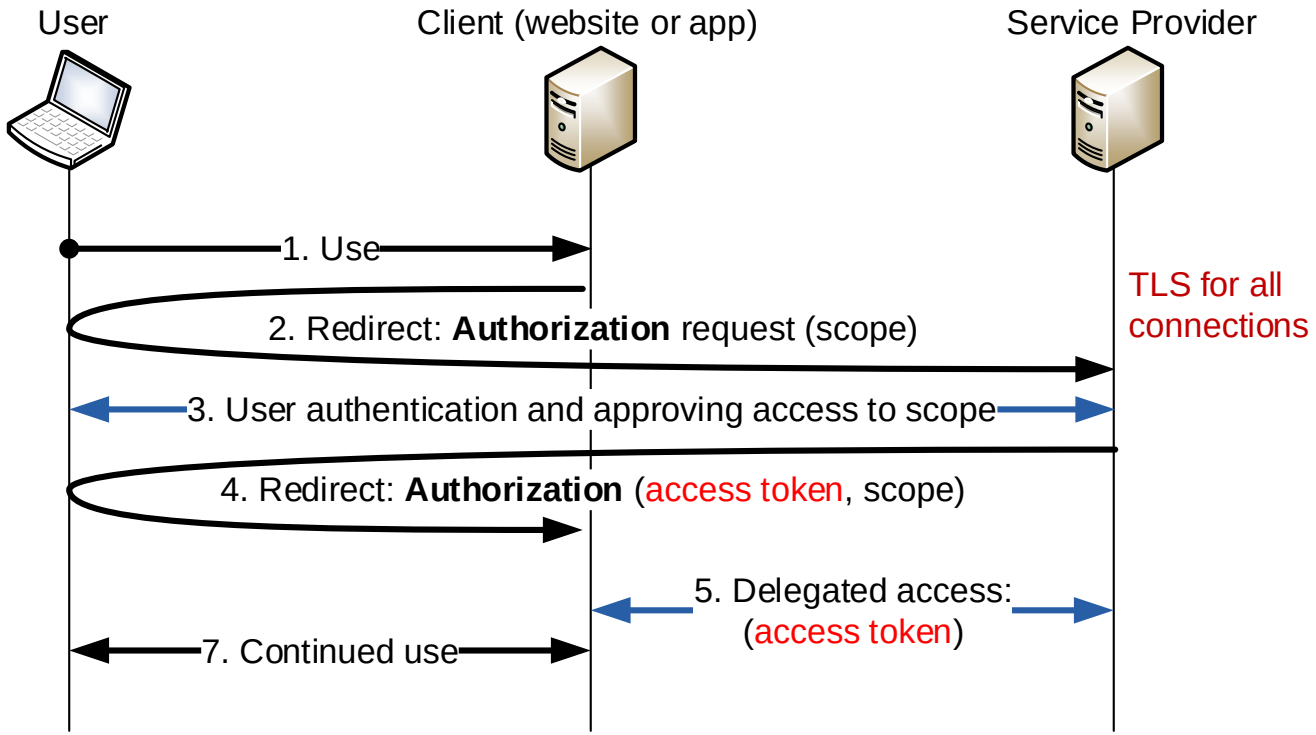
Titkosított szimmetrikus kulcs:

```
aXRlbGVzLmdvdi5odS9jcmwvR09WQ0EtTlEtU0VDLmNybDAOBgNVHQ8BAf8EBAMCBDAwEwYDVR0l
BAwwCgYIKwYBBQUHAwQwFwYDVR0RBBAwDoEMaW5mb0BuaXN6Lmh1MA0GCSqGSIb3DQEBCwUAA4IB
AQBoRCZd/jGQnBoIUL4jOo0XAJ+0SpLsG+6jUcq4+G2rsXjI5FJQUaBbm9SqxAC5PWR6QeNGx3r4
gaRH9B8YzvhYYfiR9lIapcmNc3puN6ZULmQna+1k2XSxODXvbGALzfbkIZZBXauSEsXTRxpSWCHx
E789JGD2SABKwCRWLDQLNX5c7dwpRkyPS0DBt+bi++IThA5c1glHWljiLVhJN9AAehL0kZk8n6l0
pAiN++H7UF1A0Gy9T5fGPVRc0IL3QMScbMscTO4kTkBdnW7immec1bdIZRmu1QcjU8c3tKy7Govz
h4jaWEEAyN1sgNPqyxtprr0h9CxMEp+KuSzRqMUE
      </ds:X509Certificate>
    </ds:X509Data>
  </ds:KeyInfo>
▼<xenc:CipherData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
  ▼<xenc:CipherValue>
      G0b8O9jxao01lpfZ/AapbA1CNwEqiYwKy77tByuO2dDUGD/+EirTnkEuFFShV8mcnb1s4ALYG6hnp4XRNTRvAce
    </xenc:CipherValue>
  </xenc:CipherData>
▼<xenc:ReferenceList>
    <xenc:DataReference URI="#_ea58bcbe55e27d99c65cea3b60314105"/>
  </xenc:ReferenceList>
</xenc:EncryptedKey>
```

OAuth 2.0

- OAuth was designed for authorization (i.e. delegation)
  - Allow an external web site or app to access a service on the user's behalf
- Examples:
  - Authorize an external web site or app to update your Facebook profile or page
- Standardized by IETF (RFC 6749, RFC 6750)
  - Many implementation options cause interoperability issues
  - OAuth 2.0 security is based only on TLS (Oath 1.0 used client signatures)

OAuth 2.0 protocol



- User authorizes the client (web site or mobile app) to access the service
  - Client may restrict the scope of delegated access
- Security based on an opaque access token and TLS
  - Token is typically a random number
  - Service providers remembers the scope of authorized access for each token

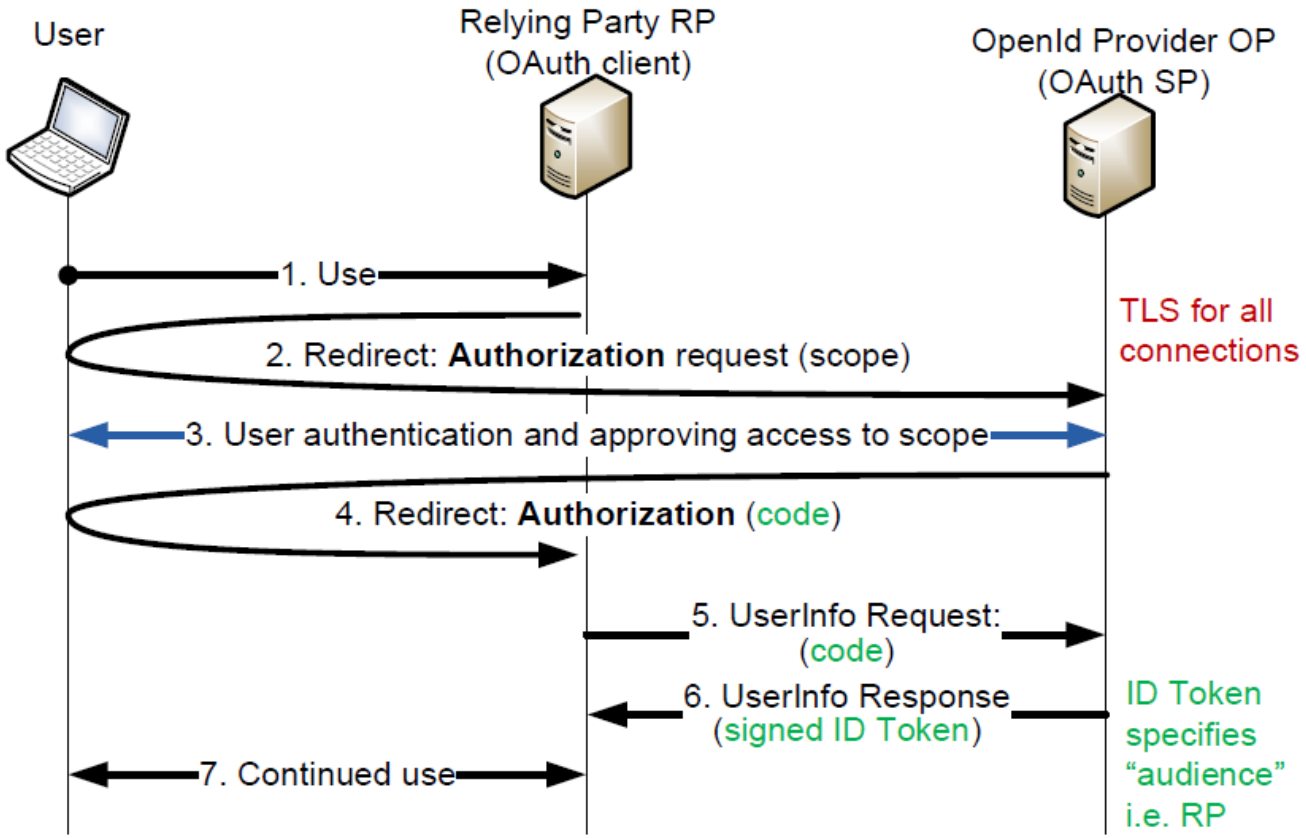
OAuth for authentication?

- The message flow in OAuth looks like SAML → tempting to use the same protocol for authentication
  - User would prove its identity by delegating access to a (dummy) resource associated with the user account
- In principle, this is a bad idea:
  - OAuth access token enables client to access a resource on the service provider
  - The client in OAuth does not know or care who gives it the token, as long as the token works for accessing SP
  - The protocol does not prevent the client from sharing the token with others

  ➔ Malicious client can impersonate its users to other clients
- Autentication based on OAuth standardized as OpenId Connect

  http://openid.net/developers/specs/

OpenId Connect



- Authentication built on OAuth 2.0, REST API, JSON data formats
- Access token is replaced with a code that is used only for retrieving user identity and other attributes from OP

ID token encoding

```
{
   "iss"        : "https://c2id.com",
   "sub"        : "alice",
   "aud"        : "s6BhdRkqt3",
   "nonce"      : "n-0S6_WzA2Mj",
   "exp"        : 1311281970,
   "iat"        : 1311280970,
   "auth_time"  : 1311280969,
   "acr"        : "urn:2fa",
   "at_hash"    : "MTI..."
 }
```

- Encoded as a JSON Web Token (JWT).
- The claims about the subject are packed in a simple JSON object.
- It is signed. Typically with the provider's private RSA key or a shared secret (HMAC) issued to the client during registration.
- Is URL-safe.

Álvéletlen generátorok

PPKE, ITK
Csapodi Márton

Miért fontos a véletlen szám

A kriptográfia sarokköve:

- Akkor nehéz kitalálni a titkot, ha az kellően véletlen.

A 9 egy véletlen szám?

Hagyományos generátorok

Excel RAND fv. (Excel 5.0 és korábbi)

Első:

r(1) = 9821 * 0.5 + 0.211327 törtrésze

Továbbiak:

r(n+1) = 9821 * r(n) + 0.211327 törtrésze

randomize=1 az .INI file [Microsoft Excel] szekciójában:

r: rendszeróra alapján indul

Excel 5.0-nál alapértelmezett

rövid ciklus (~ 1 millió)

Hagyományos statisztikai tesztek:

- 1-esek és 0-k gyakorisága
- 00, 01, 10, 11 minták (stb.) gyakorisága
- 0, 00, 000, 0000, stb minták gyakorisága
- autokorreláció

FIPS 140: Security Requirements for Cryptographic Modules

Diehard test

stb.

Hagyományos generátorok

Excel RAND fv. (Excel 2003 és újabb)

Wichmann-Hill algoritmus

X(0), Y(0), Z(0) = 1 és 30000 közti egész

X(n) = 171 * X(n-1) mod 30269
Y(n) = 172 * Y(n-1) mod 30307
Z(n) = 170 * Z(n-1) mod 30323

r(n) = X(n)/30269.0 + Y(n)/30307.0 + Z(n)/30323.0 törtrésze

hosszú ciklus, FIPS, Diehard teszteknek megfelel

MS CryptGenRandom fv.

Kiinduló érték (seed) képzés az alábbi adatok (entropy input) összefűzésével és hash-elésével:

- The process ID of the current process requesting random data
- The thread ID of the current thread within the process requesting random data
- A 32bit tick count since the system boot
- The current local date and time
- The current system time of day information consisting of the boot time, current time, time zone bias, time zone ID, boot time bias, and sleep time bias
- The current hardware-platform-dependent high-resolution performance-counter value
- The information about the system's current usage of both physical and virtual memory, and pagefile
- The local disk information including the numbers of sectors per cluster, bytes per sector, free clusters, and clusters that are available to the user associated with the calling thread
- A hash of the environment block for the current process
- Some hardware CPU-specific cycle counters

MS CryptGenRandom fv.

- The system processor performance information consisting of Idle Process Time, Io Read TransferCount, Io Write Transfer Count, Io Other Transfer Count, Io Read Operation Count, Io WriteOperation Count, Io Other Operation Count, Available Pages, Committed Pages, Commit Limit,Peak Commitment, Page Fault Count, Copy On Write Count, etc.
- The system exception information consisting of Alignment Fix up Count, Exception DispatchCount, Floating Emulation Count, and Byte Word Emulation Count
- The system lookaside information consisting of Current Depth, Maximum Depth, Total Allocates,Allocate Misses, Total Frees, Free Misses, Type, Tag, and Size
- The system interrupt information consisting of context switches, deferred procedure call count,deferred procedure call rate, time increment, deferred procedure call bypass count, andasynchronous procedure call bypass count
- The system process information consisting of Next Entry Offset, Number Of Threads, Create Time,User Time, Kernel Time, Image Name, Base Priority, Unique Process ID, Inherited from UniqueProcess ID, Handle Count, Session ID, Page Directory Base, Peak Virtual Size, Virtual Size, PageFault Count, Peak Working Set Size, Working Set Size, Quota Peak Paged Pool Usage, QuotaPaged Pool Usage, Quota Peak Non Paged Pool Usage, etc.

MS CryptGenRandom fv.

A híváskor megadható:

- byte-ok száma
- opcionálisan további entropy bitek

r(n+1) generálása különböző Windows verziókban:

- Windows 2000-ig, alap XP: RC4 adatfolyam rejtjelező alapú
- XP SP3, alap Vista, 2003 SP2, 2008 Server: SHS based RNG (FIPS 186-2)
- Vista SP1, Windows 7: AES CTR_DRBG (NIST Special Publication 800-90)

CryptoAPI Next Generation (CNG, CryptoAPI kiváltása): BCryptGenRandom fv.

- Vista, Windows 7 és 2008 Server: választható algoritmus (FIPS 186-2, NIST SP 800-90 AES CTR_DRBG és Dual_EC_DRBG)
- Windows 10: nincs Dual_EC_DRBG

Irodalom

FIPS PUB 186-2

FEDERAL INFORMATION
PROCESSING STANDARDS PUBLICATION

2000 January 27

U.S. DEPARTMENT OF COMMERCE/National Institute of Standards and Technology

DIGITAL SIGNATURE STANDARD (DSS)

FIPS186-2

Van már FIPS 186-3 is (2009. június)

Digitális aláírás szabvány
Tartalmaz előírásokat véletlen számok generálására is (kulcspár)

16-18. oldal:

XKEY(1): seed

véletlen bitek (r) generálásának lépései:

XVAL(n) = XKEY(n) + XSEED(n)

r(n) = SHA-1 (IV, XVAL(n))

XKEY(n+1) = XKEY(n) + r(n) + 1

XSEED(n): opcionális további seed bitek

IV: állandó initial value

SHA-1: 160 bites lenyomat

Irodalom

NIST Special Publication 800-90

Recommendation for Random Number Generation Using Deterministic Random Bit Generators (Revised)

Elaine Barker
John Kelsey

Computer Security Division
Information Technology Laboratory

COMPUTER SECURITY

March 2007

Fogalmak

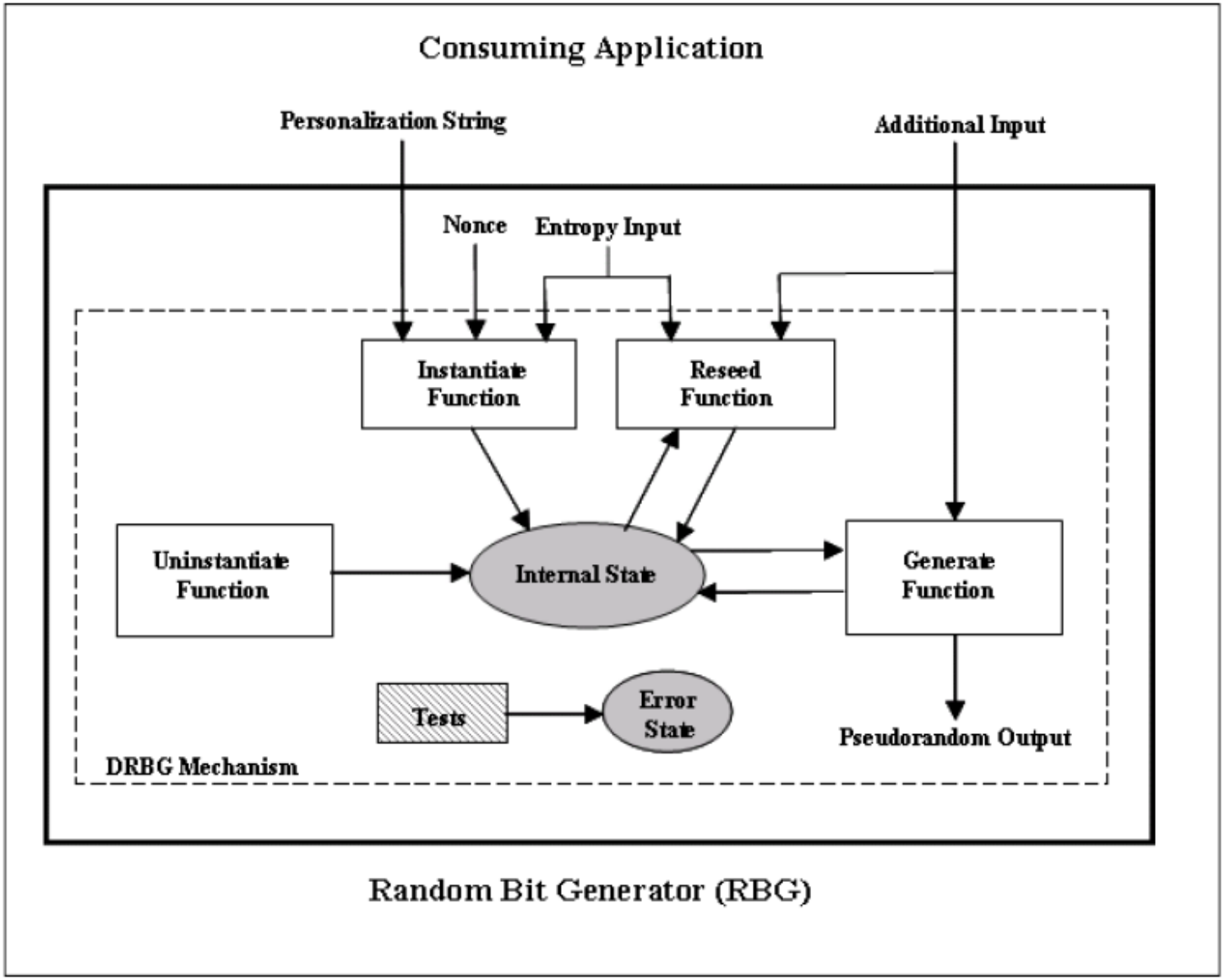
RBG - Random Bit Generator

NRBG - Non-deterministic Random Bit Generator

DRBG - Deterministic Random Bit Generator (Pseudorandom Bit Generator)

Cryptographycally Secure DRBG: csak ezzel foglalkozunk
nemcsak statisztikai tulajdonságok, hanem kriptográfiában használható

Funkcionális modell



Funkcionális modell

"Entrópia":

- magérték (seed) létrehozására
- titkos
- nem a hossza meghatározott, hanem a bizonytalansága

Egyéb bemenet:

- "nonce", személyre szabott érték, stb.
- nem feltétlenül titkos

Állapot:

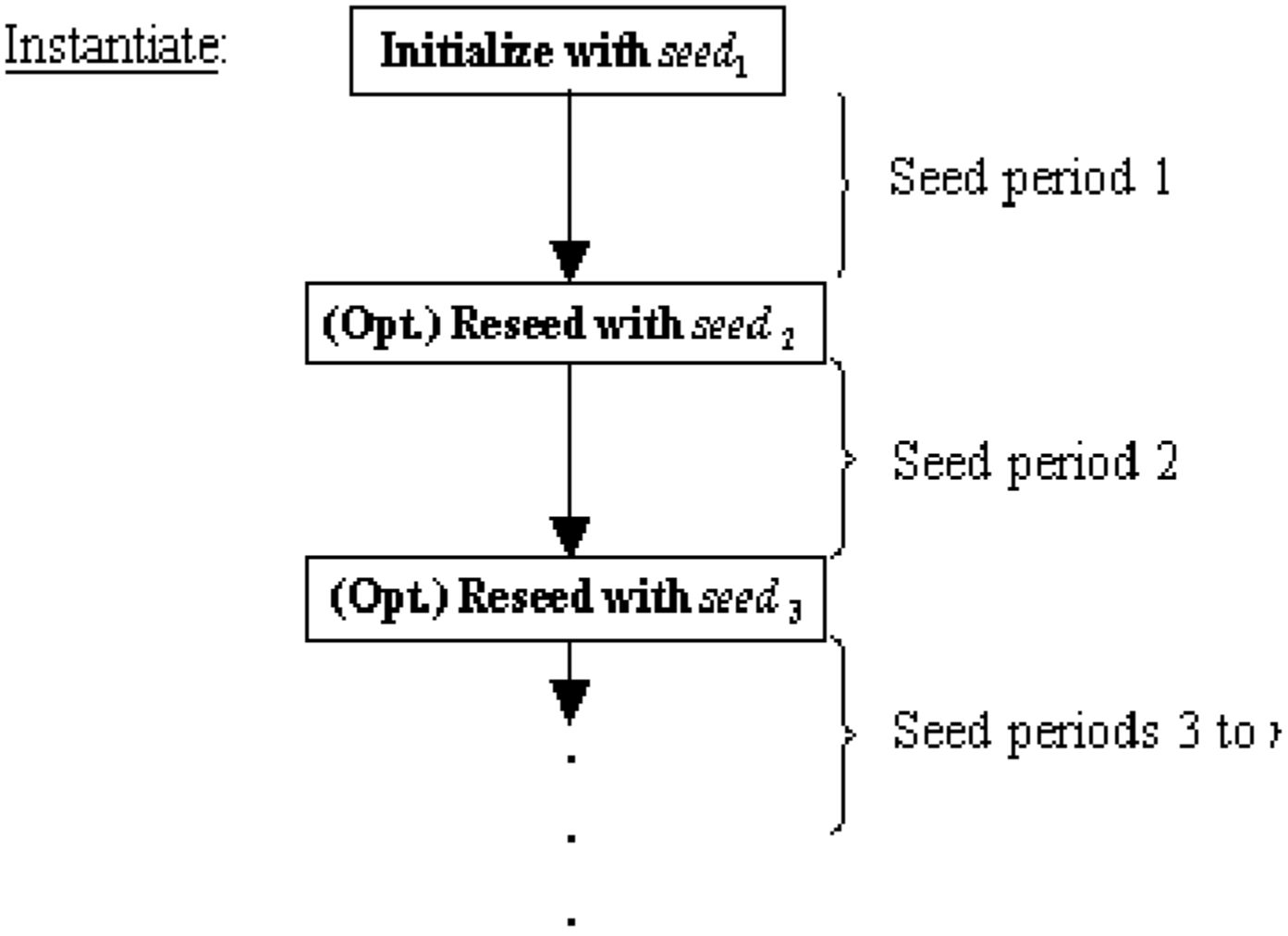
- minden belső adat, ami a következő állapot számításához kell

Műveletek:

- egyed létrehozása (instantiate)
- új magérték létrehozása (reseed)
- törlés (uninstantiate)
- gyártás (generate)

Teszt

Egyed létrehozása



Egyed létrehozása

Különböző célú véletlen számok külön egyedek lehetnek (külön seed) akkor is, ha a generálás módja azonos.

Egy egyednek lehetnek seed periódusai, ha a reseed lehetőségét kihasználjuk.

Létrejön az első állapot (titkos érték + számláló, egyéb paraméterek)

(Az állapot érték is titkos kell maradjon)

A biztonság függ:

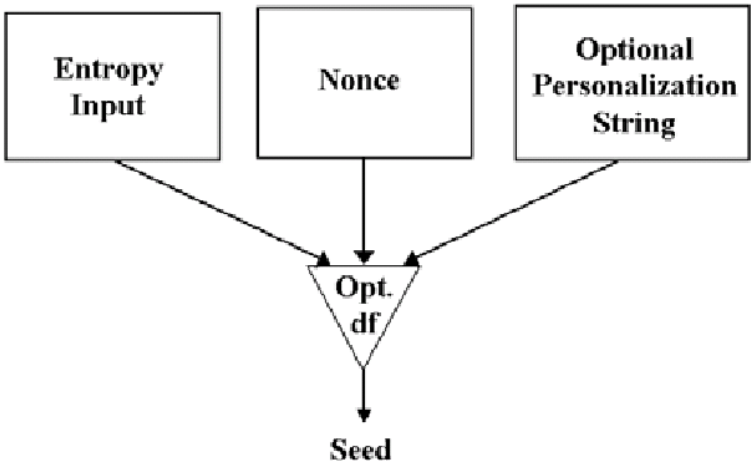
- generátortól
- egyed létrehozásától

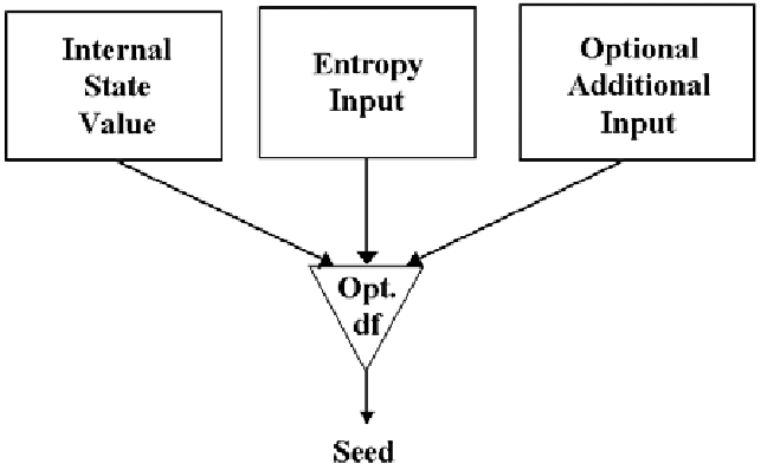
Entrópia, nonce, egyéb:

- termikus zaj a bemeneteken
- hard disk keresési idő
- rendszer óra
- felhasználó paraméterek
- hardver paraméterek
- rendszer input
- felhasználói input

Aszimmetria megszüntetése (de-skew)

Egyed létrehozása, reseed





Backtracking, prediction

Backtracking resistance:

Ismertté vált állapot alapján

- a korábbi állapotok nem állíthatók elő
- korábbi output nem különböztethető meg a véletlentől

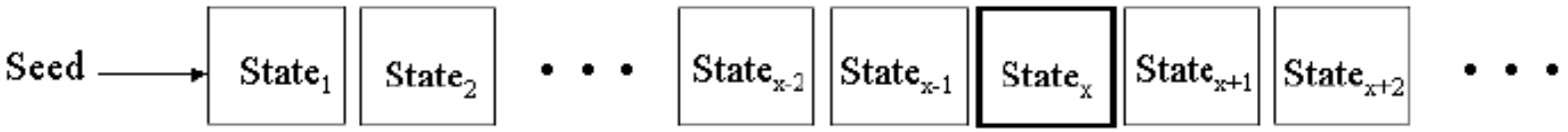
Ez mindig elvárható.

Prediction resistance:

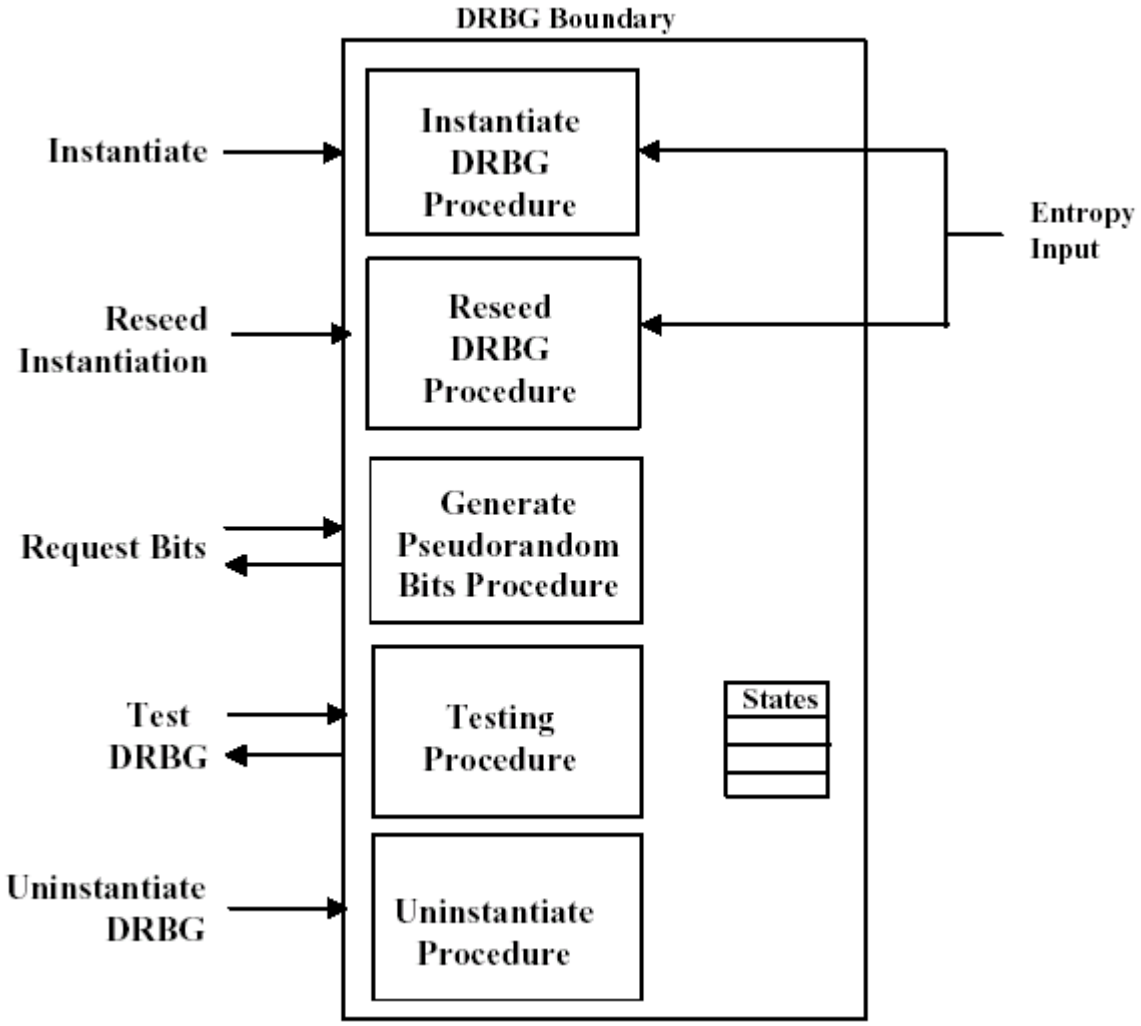
Ismertté vált állapot alapján

- későbbi állapotok nem állíthatók elő
- későbbi output nem különböztethető meg a véletlentől

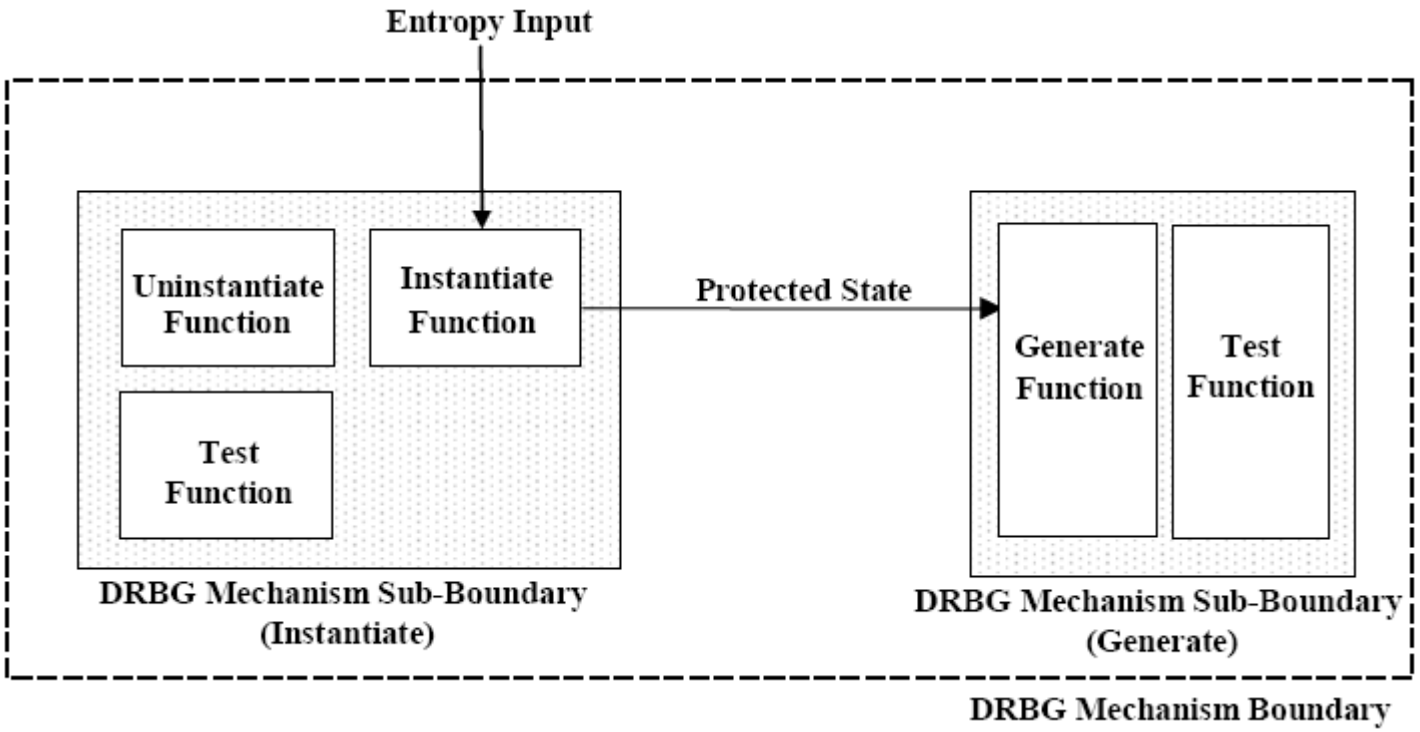
Ez nem várható el mindig. Ha szükséges, akkor reseeding kell minden alkalommal.



A generátor határai, biztonsága



A generátor határai, biztonsága



Korrekt megoldások

Alapelvek:

- erős seed
- kriptográfiailag erős generátor fv.
- állapot ne kerüljön nyilvánosságra
- bitsorozat ismerete ne okozzon problémát

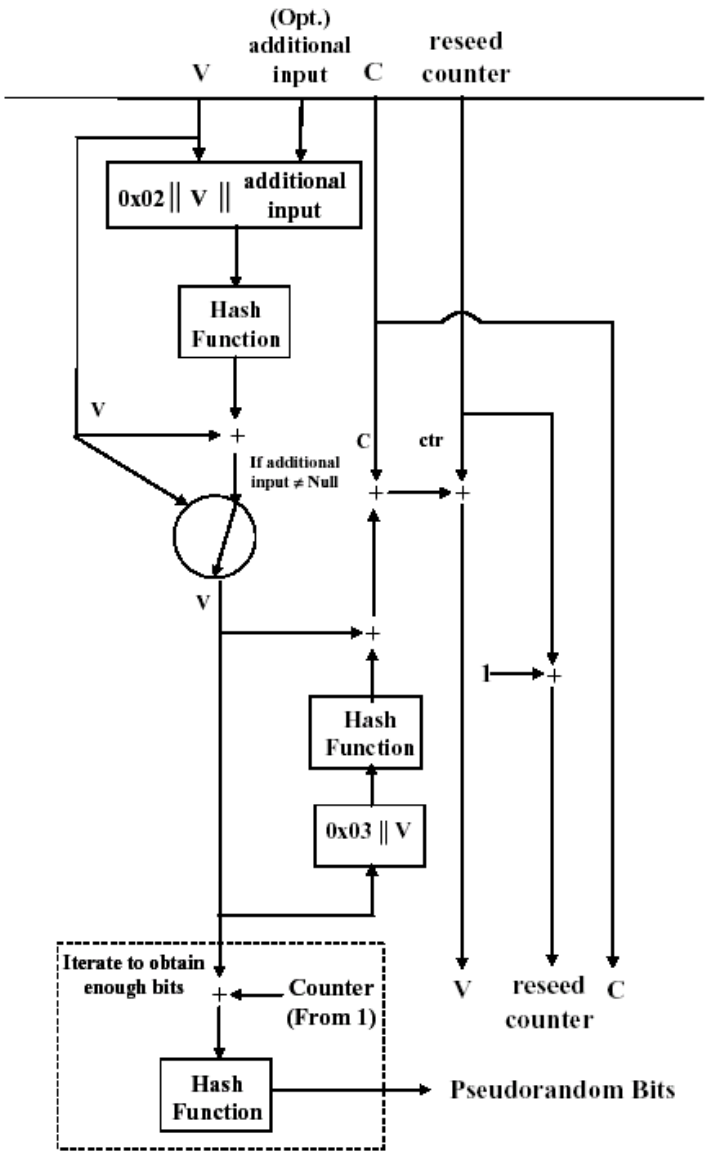
Vagyis: pl. hash alkalmazása esetén nem az output bitek leképezésével kapjuk a további biteket.

Lehetőségek:

- hash (seed | i)
- encrypt (seed | i)
- encrypt (output)
- Blum-Blum-Shub

Lehet minden ciklusban "entrópiát" hozzátenni.

Hash alapú minta

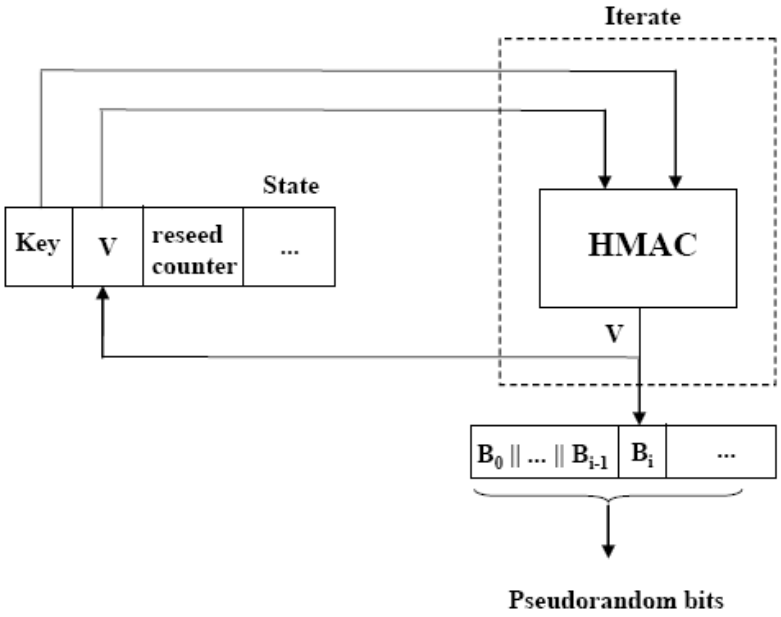


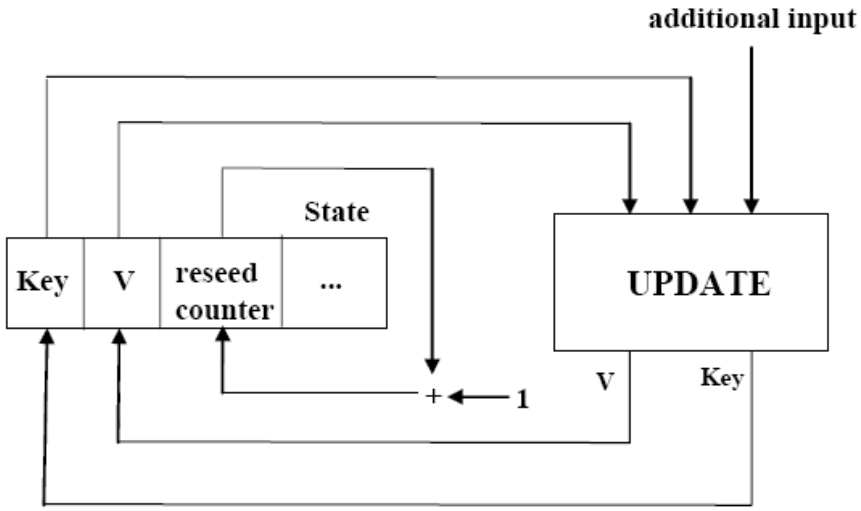
V: állapot, V_0: seed

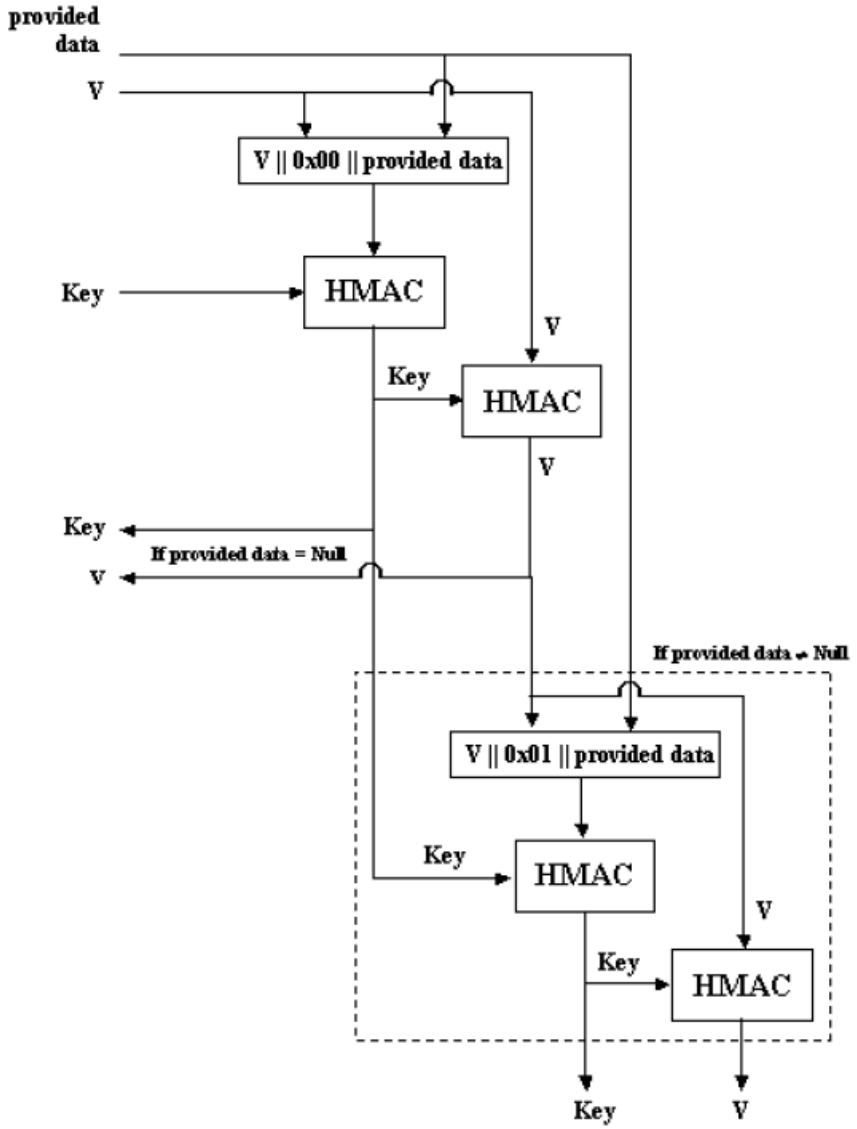
C: hash(seed)

reseed counter: számláló

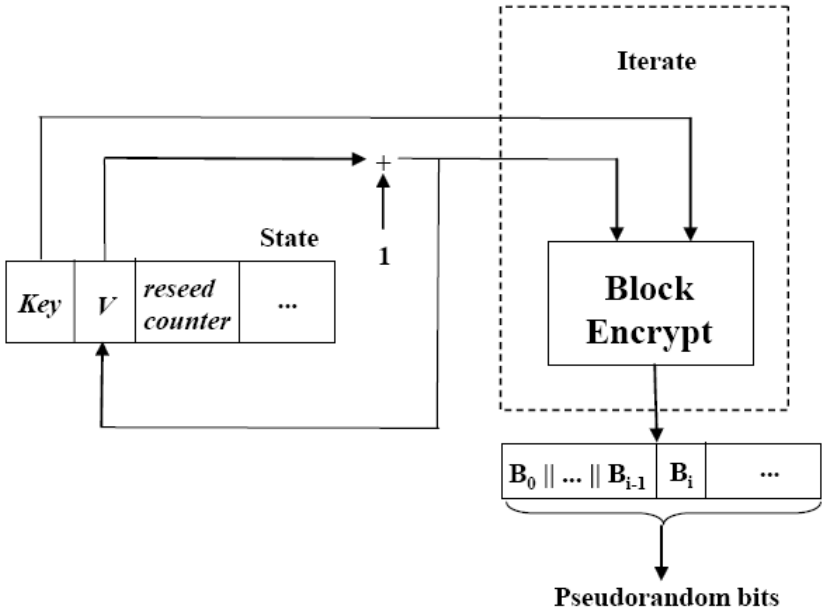
Keyed-Hash alapú minta

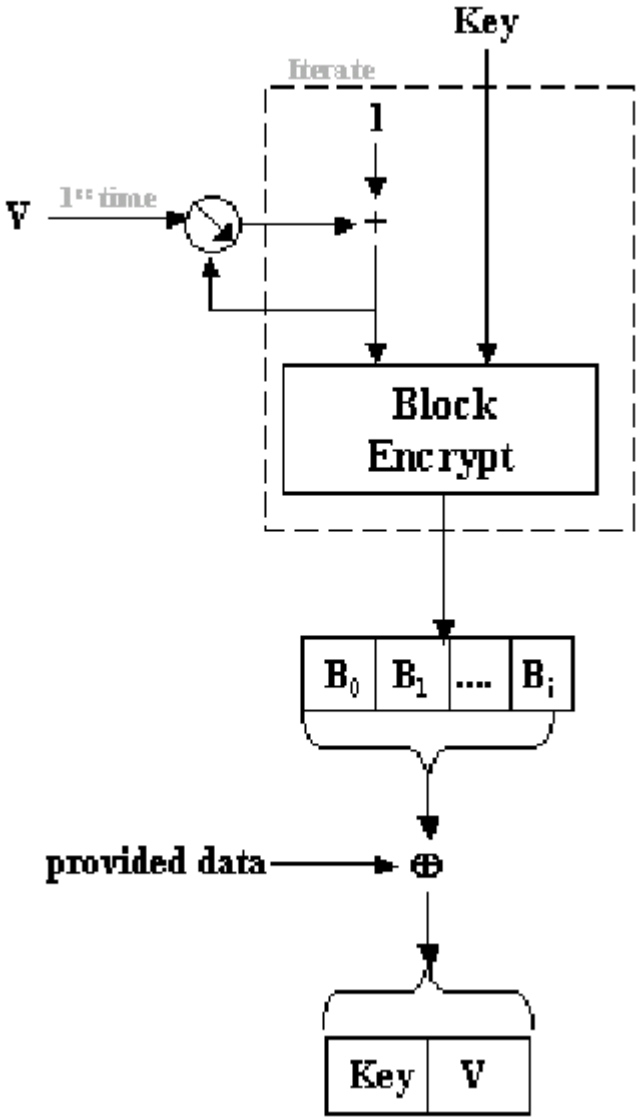


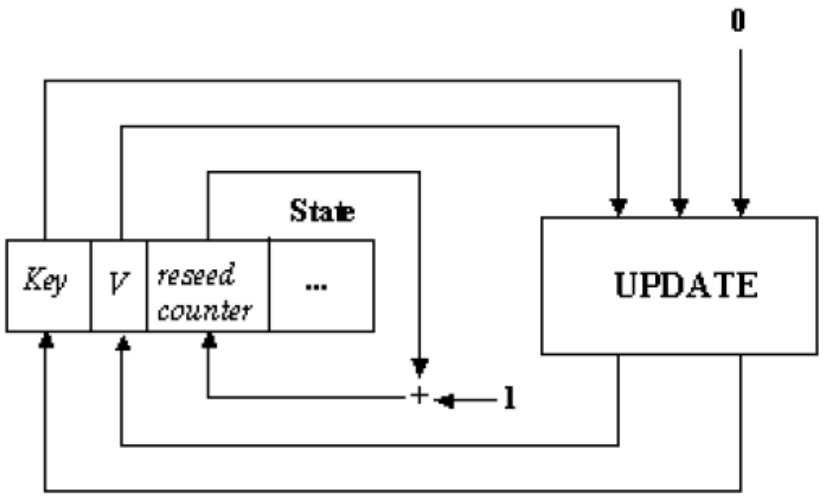




Blokk rejtjelező alapú minta







Blum-Blum-Shub generátor

The Blum-Blum-Shub PRBG is the following algorithm :

- *Generate p and q, two big Blum prime numbers.*
- $n := p \cdot q$
- *Choose $s \in_R [1, n-1]$, the random seed.*
- $x_0 := s^2 \pmod{n}$
- *The sequence is defined as $x_i := x_{i-1}^2 \pmod{n}$ and $z_i := parity(x_i)$.*
- *The output is $z_1, z_2, z_3, ...$*

where $parity(x_i)$ is defined as $R_2(x_i)$.

Let $n = p \cdot q = 7 \cdot 19 = 133$ and $s = 100$. Then we have $x_0 = 100^2 \pmod{133} = 25$. The sequence $x_1 = 25^2 \pmod{133} = 93$, $x_2 = 93^2 \pmod{133} = 4$, $x_3 = 4^2 \pmod{133} = 16$, $x_4 = 16^2 \pmod{133} = 123$ produces the output $1, 0, 0, 1$.

Problémás generátorok

MS CryptGenRandom

2007, Leo Dorrendorf et al., Hebrew University of Jerusalem and University of Haifa

Nem publikált algoritmus visszafejtése

128KByte-onként van entropy input

State kiolvasható a memóriából

Nincs backtracking resistance

Javítva:

Windows XP – Service Pack 3

Windows Vista – Service Pack 1

NIST Backdoor in Elliptical Curve DRBG

2007, Dan Shumow and Niels Ferguson, Microsoft

A görbe paraméterei lehet, hogy trükkösen vannak generálva. Ha ez igaz, akkor a trükk ismeretében az outputból a state könnyen generálható.

2013, documents released by Edward Snowden indicated that the NSA had paid RSA Security $10 million to make Dual_EC_DRBG the default in their encryption software

2014, NIST withdrew Dual EC DRBG from its draft guidance on random number generators

Problémás generátorok

PlayStation 3

2010, fail0verflow

A PS3-on futtatható kódokat hitelesítő magánkulcs megszerzése, publikálása.

Az aláírás algoritmusa (elliptic curve digital signature algorithm, ECDSA) minden aláíráshoz egyedi véletlen szám generálását igényli. Egyébként a magánkulcs előállítható.

A Sony erről megfeledkezett.

RSA publikus kulcs feltörése

2012, Lenstra, Hughes, Augier, Bos, Kleinjung, and Wachter

Néhány millió RSA publikus kulcsot (n=p*q) megvizsgálva kiderült, hogy legalább az egyik prímszám megegyezik több ezer esetben.

n1=p*q1 és n2=p*q2 esetén az lnko nem 1, hanem p. Euklideszi algoritmussal könnyen meghatározható.

Problémás generátorok

Java nonce collision

2013, Bitcoin bejelentés

Hiba a SecureRandom Java osztályban. Ismétlődések az ECDSA által igényelt véletlen számok között. A magánkulcs megszerezhető.

Ezt használta egy időben a Bitcoin Androidon. Más Bitcoinja felett lehet rendelkezni a magánkulcs megszerzésével. Több mint 50BTC veszett el. Gyorsan kijavították (nem SecureRandom-ot használ)

WEP és WPA

PPKE, ITK
Csapodi Márton

IEEE 802 (1983)

A LAN (Local Area Network) szabványai

A 7 rétegű OSI modell alsó két rétegéről szólnak: Data Link Layer, Physical Layer

A Data Link Layer két alrétegre bomlik az IEEE 802 szerint:

- Logical Link Control (LLC): a Network Layer felé elrejti a konkrét hálózat típusát
- Media Access Control (MAC): függ a hálózat típusától (Ethernet vagy WLAN)

OSI (Open Systems Interconnection) modell emlékeztető:

3. (Network Layer): IP csomagok közlekednek a forrás és a cél között
2. (Data Link Layer): Pont-pont kapcsolat az eszközök és a hálózati csomópontok között. Címzés minden hálózati eszközre egyedi MAC (media access control) address alapján.
1. (Physical Layer): A jel fizikai átvitele a médiumon.

IEEE 802.11

A WLAN (Wireless Local Area Network) szabványai

- rádióhullám: 2.4-2.5 GHz (3.6 GHz, 5 GHz is)
- bitráta: 1-300 Mbit/s (Gbit/s is)
- hatótávolság: néhányszor 10m (de több 100km is lehet parabola antennával)

Wi-Fi: A szabványnak való megfelelést igazoló kereskedelmi márkajelzés

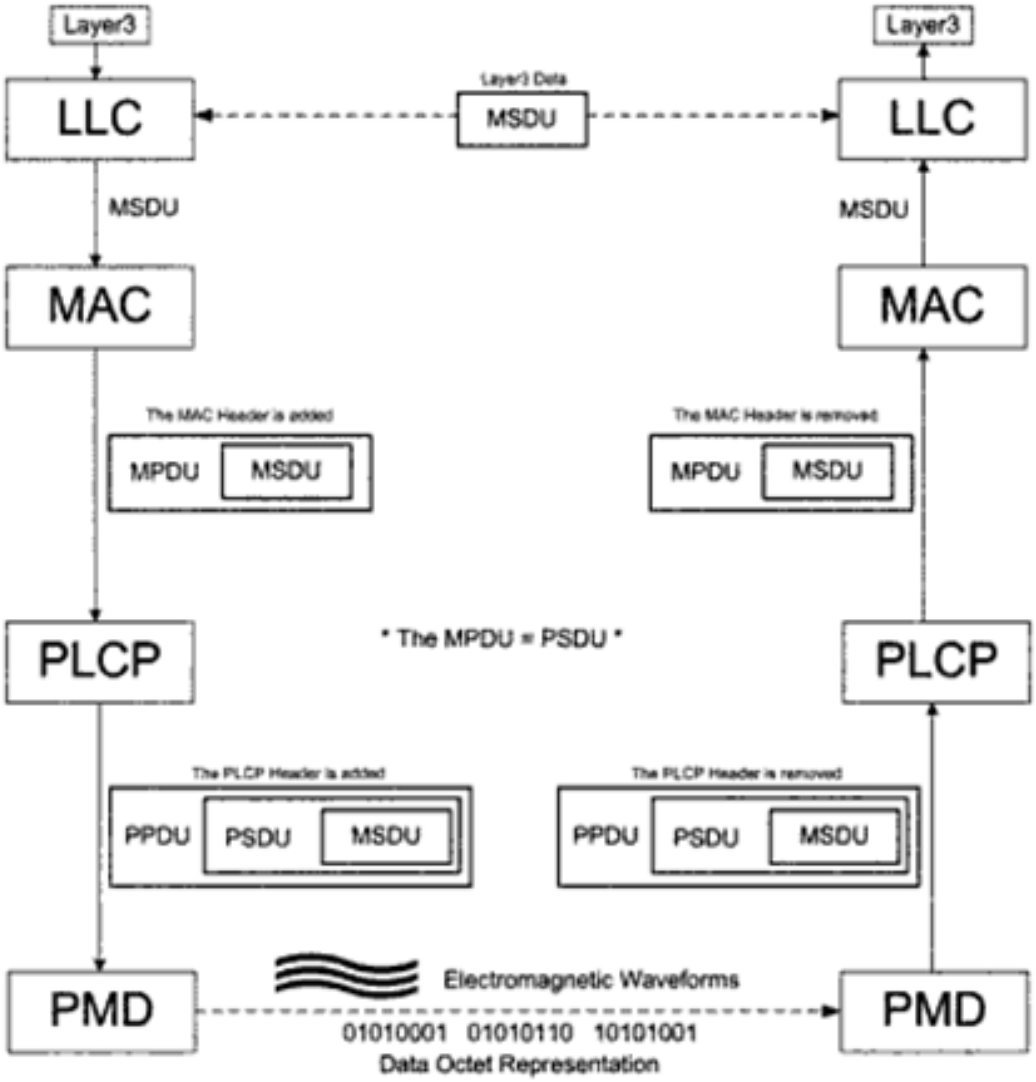
Az alap szabvány (1997, 1-2 Mbit/s) fontosabb kiegészítései:

- 802.11b: 10 Mbit/s-ig (1999)
- 802.11g: 54 Mbit/s-ig (2003)
- 802.11i: Enhanced Security (2004) - WPA2
- 802.11n: 600 Mbit/s-ig (2009)
- 802.11ad: 7 Gbit/s (2016) (új brand: WiGig)

WPA (Wi-Fi Protected Access) átmeneti megoldás volt a WPA2 szabvány elkészültéig

WLAN rétegek

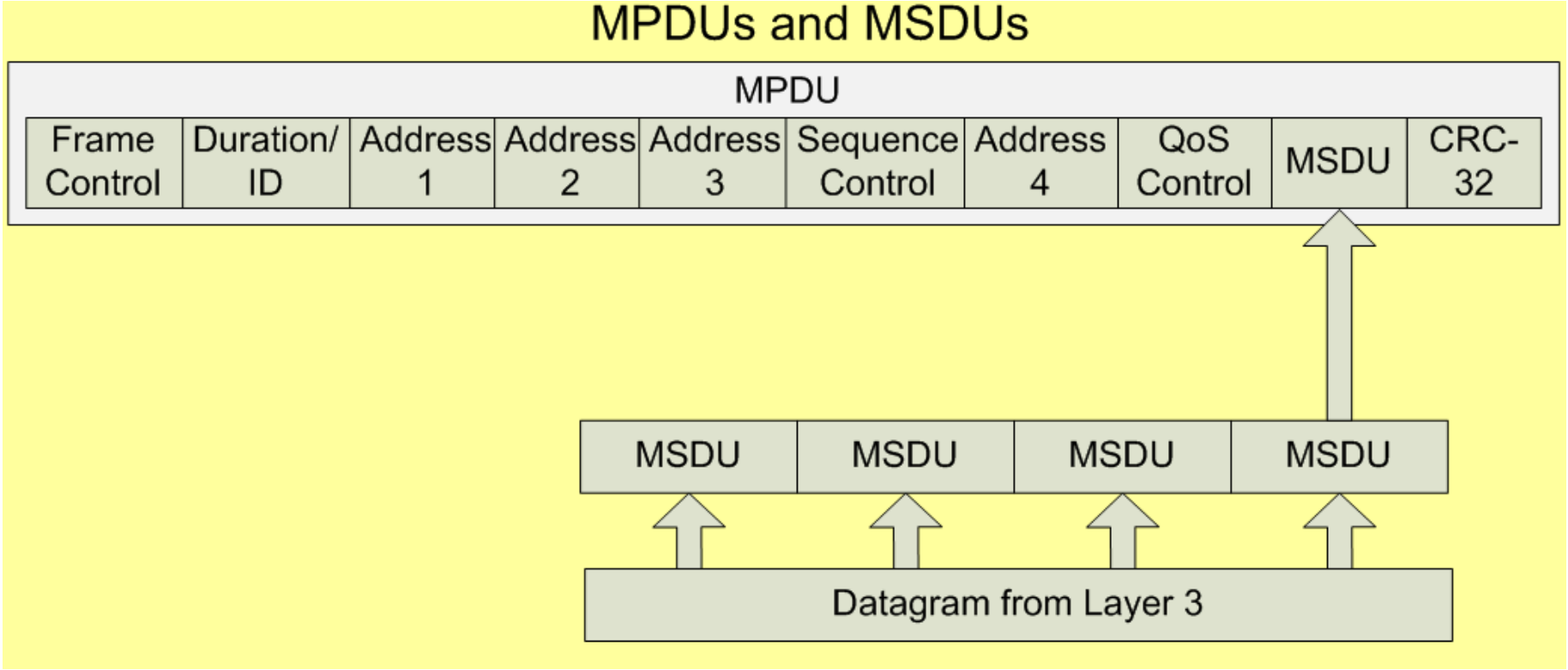
- MAC Service Data Unit (MSDU): Felette lévő OSI layerből kapott adat (pl. IP csomag), max.2304 byte-ra tördelve
- MAC Protocol Data Unit (MPDU): MSDU + header + ellenőrzőösszeg
- PLCP: Physical Layer Convergence Protocol
- PSDU: PLCP Service Data Unit (megegyezik az MPDU-val)
- PPDU: PLCP Protocol Data Unit
- PMD: Physical Medium Dependent



MAC üzenetcsomag

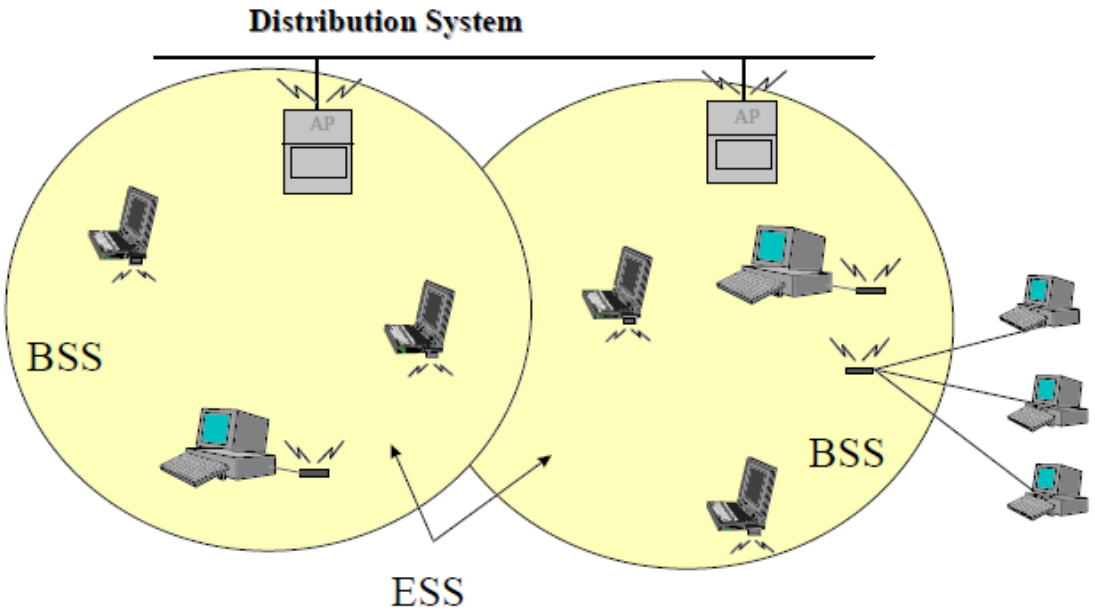
MSDU: titkosítás esetén ez megnő 8 byte-tal max. 2312 byte-ra

MAC Protocol Data Unit (MPDU): MSDU + header (max. 30 byte: 4x6 byte MAC address + 6 byte) + ellenőrzőösszeg (4 byte)



Tipikus WLAN

AP (Access Point): egy cellát kontrolláló bázisállomás
station: az AP-hez csatlakozó eszközök
BSS (Basic Service Set): egy AP és a csatlakozó eszközök együttese
BSSID (BSS Identifier): az adott BSS azonosítója (az AP MAC address-e)
IBSS (Independent BSS): ad-hoc hálózat, nincs kontrolláló AP
ESS (Extended Service Set): több BSS-ből áll, kifelé egy hálózatként látszik
SSID (Service Set Identifier):
egy vagy több AP-ból
álló WLAN azonosítója
Distribution System:
az AP-kat összekötő hálózat,
lehet Ethernet vagy wireless



Címmezők

4 címmező a kommunikáló eszközök és az irány függvényében

Frame Path	Address 1	Address 2	Address 3	Address 4
Frame between two wireless clients	Destination MAC	Source MAC	BSSID	N/A
Frame from network through AP to Client	Destination MAC	AP's MAC	Source MAC	N/A
Frame from client to network through AP	BSSID	Source MAC	Destination MAC	N/A
Frame traveling between two APs in a WDS	Receiving AP's MAC	Transmitting AP's MAC	Destination MAC	Source MAC

Frame típusok

Management Frame: access point broadcast üzenetek, autentikáció, kapcsolat létrehozása, átadása, bontása

Control Frame: Request to send, Clear to send (a csomagok egyszerre történő küldésének elkerülésére), Acknowledgement (hibamentes fogadás nyugtázása)

Data Frame: IP csomagokat szállít, csak ez kerülhet titkosításra, a többi nyílt

IEEE 802.11 biztonság

Az IEEE 802.11i kiegészítést megelőzően az alábbi lehetőségek léteztek:

	WEP Encryption	**No Encryption**
Open System Authentication	Encryption No Authentication	No Encryption No Authentication
Shared Key Authentication	Encryption Authentication	No Encryption Authentication

WEP: Wired Equivalent Privacy

IEEE 802.11 OSA

Nincs autentikáció.

Kölcsönösen elküldik a MAC address-t a kapcsolódás előtt.

Az AP-k lehetővé tesznek MAC address szűrést. Nincs értelme bekapcsolni, mert nehéz menedzselni összetett WLAN esetén, ráadásul a MAC address nyíltan közlekedik, lehallgatható és a támadó erre konfigurálhatja a hálózati kártyáját.

Az AP-k lehetővé teszik az SSID elrejtését. Nincs értelme bekapcsolni, mert ez is lehallgatható.

WEP titkosítás használható előre megosztott kulcs alapján. Ez valójában autentikációt is biztosít ezáltal. Biztonságosabb, mint a Shared Key Authentication

IEEE 802.11 SKA

Shared Key Authentication:

1. kliens kapcsolódni akar
2. access point 1024 bites véletlen challenge-et küld
3. kliens elküldi: IV, WEP([challenge,checksum], [IV,shared key])
4. access point dekódol, összevet, nyugtáz

IV: 24 bites kezdeti érték, kliens határozza meg

[challenge,checksum]: a kihívás és a CRC-32 összefűzve

[IV,shared key]: a kezdeti érték és a megosztott kulcs összefűzve

WEP(M, k): WEP algoritmussal, k kulccsal titkosított M adat

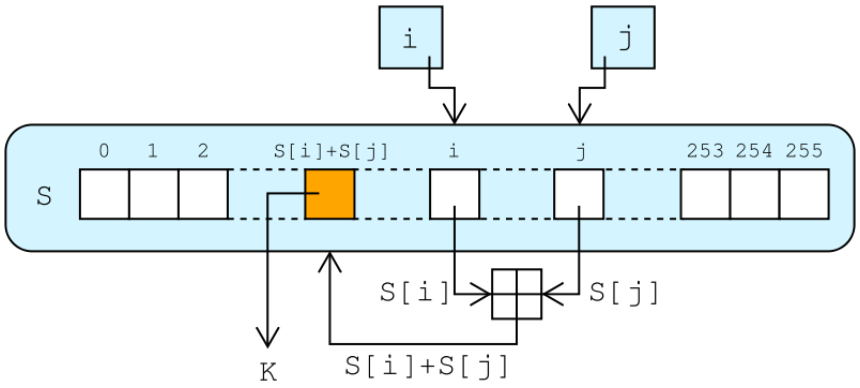
WEP - Wired Equivalent Privacy

Eredetileg 40 bit kulcs (5 karakter) + 24 bit IV (export korlátozások miatt)

Később 104 (13 karakter) + 24 bit IV is

RC4 algoritmus (64 vagy 128 bites kulccsal) a kulcsfolyam generálásra

Vernam rejtjelező (XOR)



WEP - SKA sérülékenység

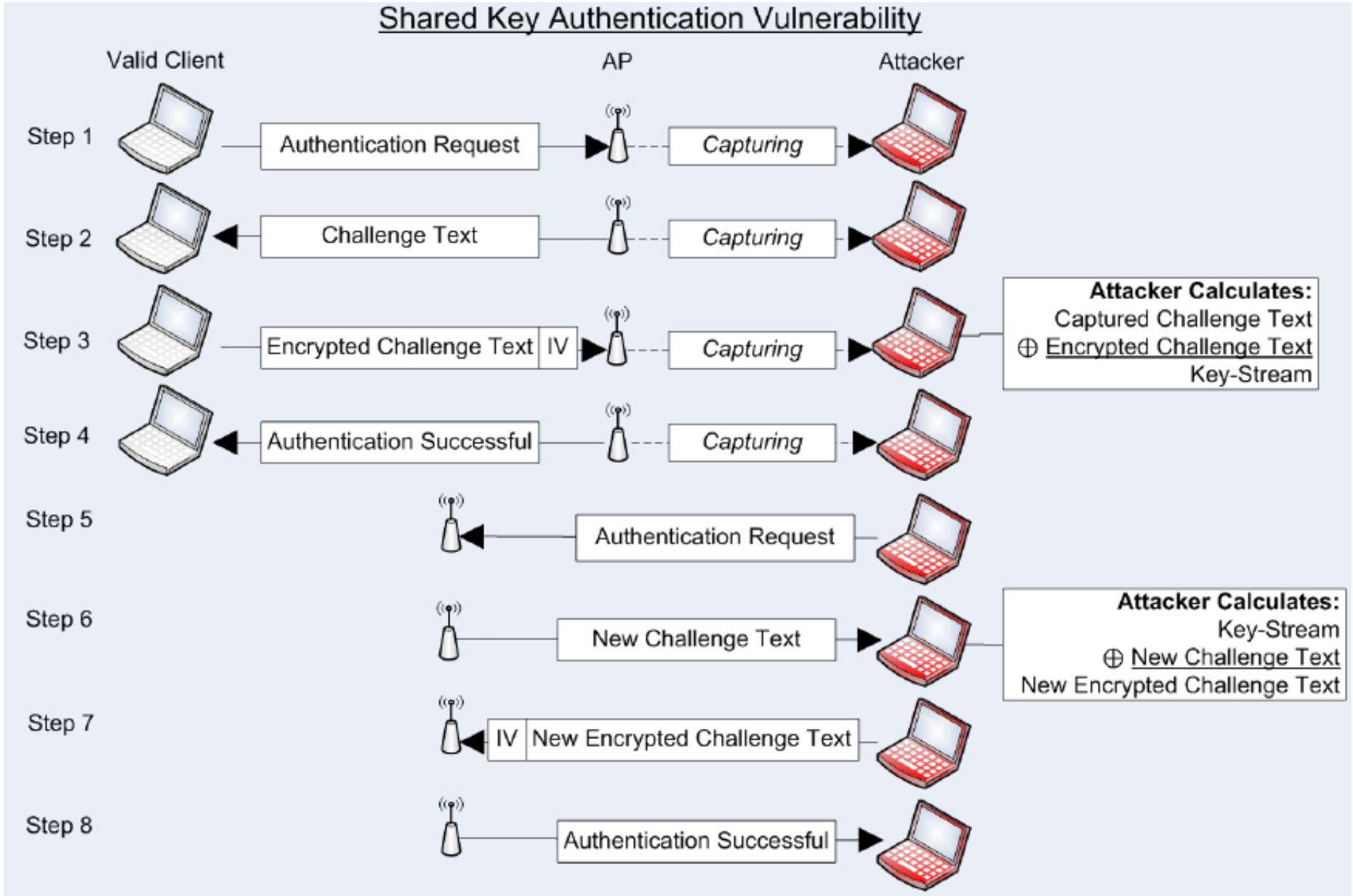
Autentikáció sérülékeny:

Challenge és WEP([challenge,checksum], [IV,shared key]) ismeretében a kulcsfolyam számítható a Vernam típusú adatfolyam-rejtjelező miatt. Ugyanazzal az IV-vel a támadó később autentikálhat.

A probléma abból ered, hogy a kliens határozza meg az IV-t és nyíltan küldi el.

Ha titkosítás is be van állítva, a támadó a sikeres autentikáció után sem tud kommunikálni.

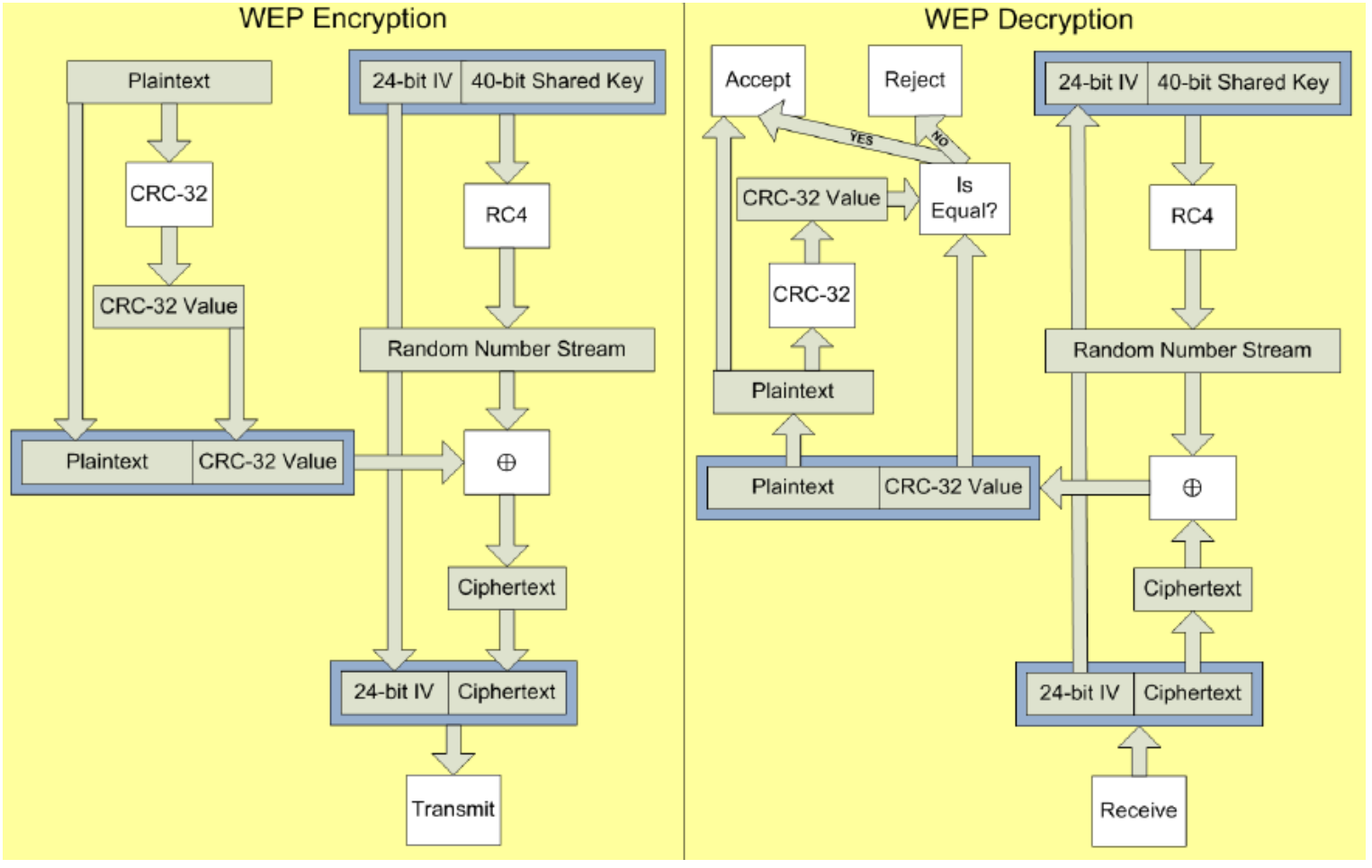
(Tilos használni az SKA-t titkosítás nélkül, mert nem biztosít autentikációt.)


Shared Key Authentication Vulnerability
Valid Client
AP
Attacker
Step 1
Authentication Request
Capturing
Step 2
Challenge Text
Capturing
Step 3
Encrypted Challenge Text
IV
Capturing
Attacker Calculates:
Captured Challenge Text
⊕ Encrypted Challenge Text
Key-Stream
Step 4
Authentication Successful
Capturing
Step 5
Authentication Request
Step 6
New Challenge Text
Attacker Calculates:
Key-Stream
⊕ New Challenge Text
New Encrypted Challenge Text
Step 7
IV
New Encrypted Challenge Text
Step 8
Authentication Successful

WEP rejtjelező

Szinkron üzemmódú adatfolyam rejtjelező: szinkronizálni kell, minden frame-nél újra indul -> kulcs változatlan, IV változik

Integritás védelem: 32 bites checksum



WEP - checksum sérülékeny

Érvényes ciphertext hozható létre a ciphertext módosításából.

Ha egy lehallgatott, érvényes ciphertext adat mezőjét megváltoztatjuk és az alábbi módon származtatjuk a módosított ciphertext-et:

C'=C XOR ([változás,CRC-32(változás)])

akkor a fogadó félnél az integritás ellenőrzés nem mutatja ki a módosítást.

Tetszőleges módon módosítható a ciphertext, a CRC-32 alapú integritásvédelem nem védi az integritást.

A probléma abból ered, hogy a CRC-32 leképezés és a Vernam rejtjelezés is lineáris függvény, így:

([M,CRC(M)] XOR K) XOR ([dM,CRC(dM)] = [(M XOR dM),CRC(M XOR dM)] XOR K

M: nyílt üzenet

dM: nyílt üzenet megváltozása

[M,CRC(M)]: összefűzött üzenet és ellenőrző összeg

WEP - IV ismétlődés

24 bites az IV

Születésnapi paradoxon: pár ezer frame-ben valószínűleg van két azonos IV

Azonos IV esetén megegyeznek a kulcsfolyamok, így a két ciphertext különbsége megegyezik a két cleartext különbségével.

WEP - frame beszúrás

DoS támadás intézhető egy érvényes MPDU üzenet ismétlésével.

Az AP elfogadja az üzenetet, mivel az IV véletlenszerű, így nem zárható ki az ismétlődés. Kicsomagolja és a tartalmát (MSDU) továbbadja a magasabb OSI rétegek felé.

WEP - kulcs management

A szabvány két kulcstípust tesz lehetővé:

- key mapping key: MAC address alapján egyedi kulcs minden kliensnek
- default key: amelyik MAC addresshez nem rendel az AP egyedi kulcsot, ez a default

Elvileg lehetséges volna minden kliensnek külön kulcsot adni és azt az AP-kben tárolni. De ez bonyolult. A gyakorlat az, hogy minden kliens ugyanazt a kulcsot használja:

- Akinek megvan a kulcs, minden kommunikációt dekódolni tud. Egymás üzeneteit dekódolni tudják a WLAN-on belül.
- Ha egy eszköz kompromittálódik vagy valaki kiadja másnak a kulcsot, idegen személy/eszköz is mindent le tud hallgatni.
- Ha megszűnik egy jogosultság (pl. kilépő dolgozó), cserélni kellene a kulcsot, de ez nem történik meg.

WEP - gyenge kulcs

A beállított jelszó nem 40, illetve 104 bit entrópiájú, így szótár alapú támadással kereshető.

(A 40 bites kulcs brute force módon is megkereshető.)

WEP - shared key felderítése az AP támadásával

ARP (Address resolution protocol): IP cím és MAC address között teremt kapcsolatot.

Mérete alapján felismerhető: mindig 42 vagy 60 byte

Fix 16 byte header: 8 byte LLC réteg header-je, 8 byte (kétféle: request v. reply) ARP header

A rejtjelezett frame első 16 byte-jából 16 byte key stream meghatározható.

A frame beszúrás támadás lehetővé teszi, hogy ugyanazt az ARP kérést számtalanszor elküldje a támadó, amire mindig új IV-vel kap választ, amiből mindig kinyerhető a 16 byte key stream.

Megfelelő mennyiségű key stream alapján a shared key megállapítható.

Szükséges idő: 1 perc (104 bites kulcsra)

WEP - shared key felderítése a kliens támadásával

Ehhez a támadáshoz nem kell az AP közelében lenni, csak a bekapcsolt kliens közelében.

A kliens állandóan poll-ozza az elérhető AP-ket.

A támadó lehallgatja az SSID-t és megszemélyesíti az AP-t: küld egy kihívást.

Kliens elküldi a titkosított kihívást.

AP visszaküldi, hogy "sikeres autentikáció"

Kliens újabb titkosított üzenetet küld.

Néhány óra alatt a 104 bites kulcs megfejtéséhez szükséges mennyiségű adat keletkezik.

IEEE 802.11i kiegészítés

Két új algoritmus:

WPA (Wi-Fi protected access): TKIP algoritmus (Temporal Key Integrity Protocol). RC4 a titkosító algoritmus, de IV és shared key keveredik, továbbá jobb az integritásvédelem.

WPA2: AES blokk rejtjelező CCMP (Counter Cipher Mode with Block Chaining Message Authentication Code Protocol) módban

Mindkettőnél:

Új kulcs hierarchia

Négy lépéses kölcsönös autentikáció

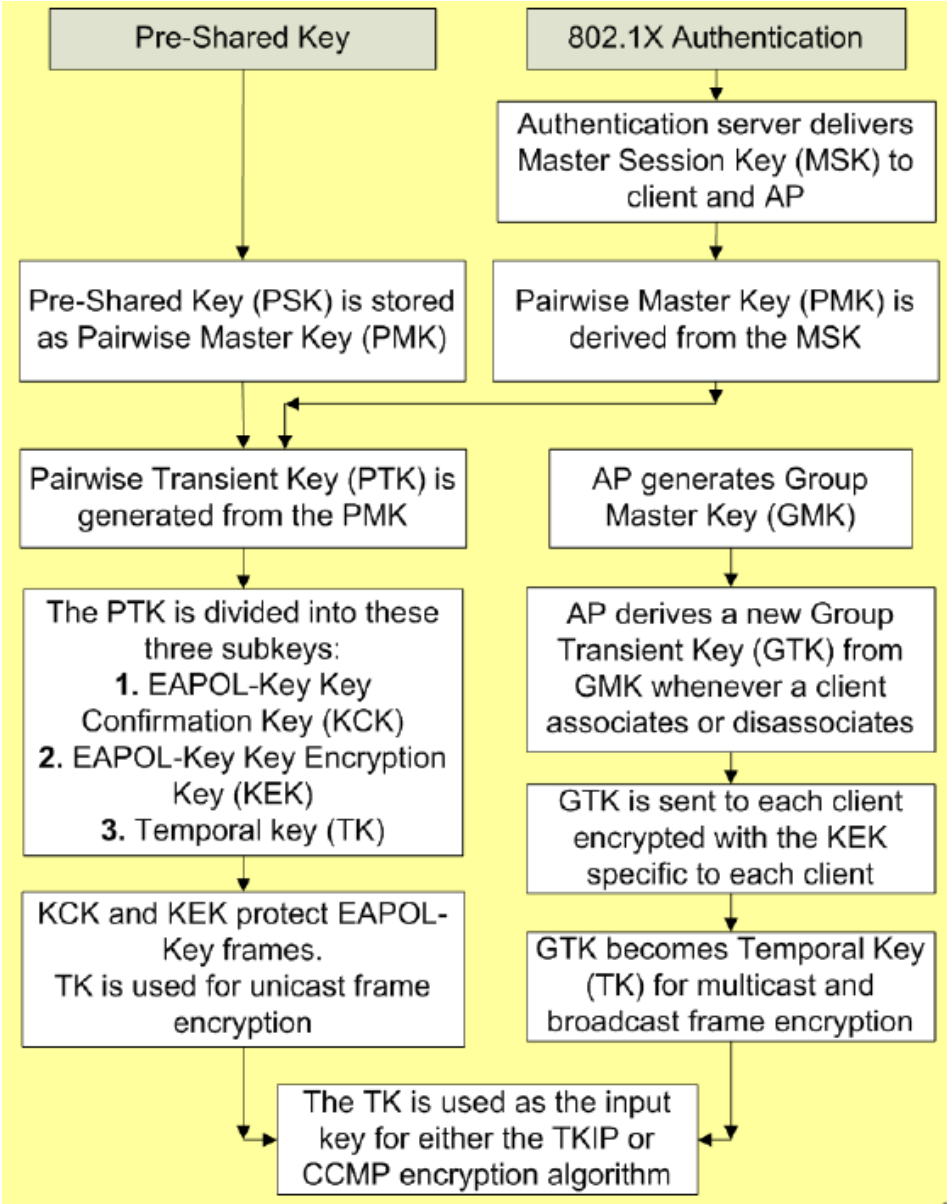
Előre megosztott kulcs vagy LAN hálózati hitelesítésből származtatott kulcs

IEEE 802.11i kulcs hierarchia

PMK: megosztott kulcsból (állandó) vagy LAN autentikációból (kliensenként és kapcsolódásonként különböző)

PTK: Kapcsolatonként eltérő akkor is, ha PMK fix. Ebből származnak a kapcsolat kulcsai.

GMK: Broadcast üzenetek közös kulcsa (ne kelljen n példányban kódolni). Minden kapcsolat bontáskor újra generálódik, így a kilépő kliens nem rendelkezik vele.

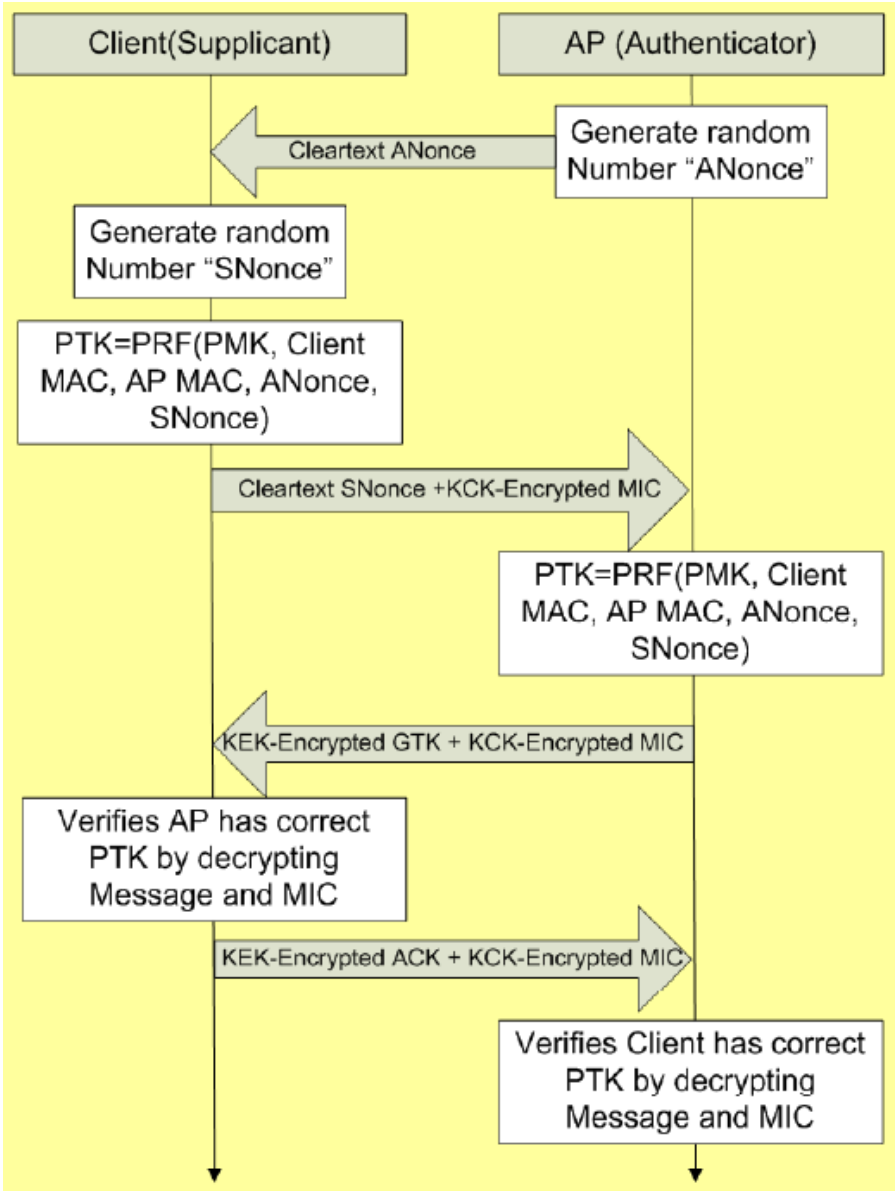


IEEE 802.11i handshake

Egyedi PTK létrehozása a PMK-ból, eszköz azonosítókból és mindkét fél véletlen számából.

GTK megküldése.

Kölcsönös hitelesítés.



Adatfolyam rejtjelezők

PPKE, ITK
Csapodi Márton

Szimmetrikus kulcsú rejtjelezés

Általában a rejtjelező kulcs és a dekódoló kulcs megegyezik, vagy egyikből a másik meghatározása "könnyű".

PÉLDA:

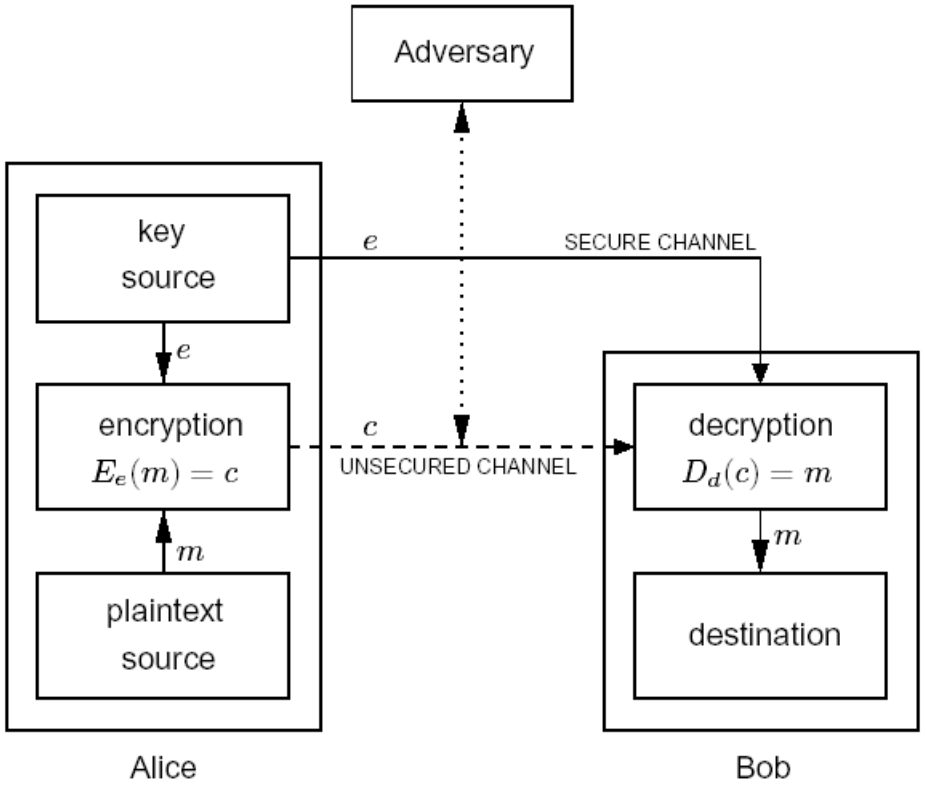
A: {A, Á, B, C,..., Z}

M, C: 5 hosszúságú karakterlánc

e: egy permutációja A-nak

A kulcsot el kell juttatni a feleknek és titokban kell tartani.

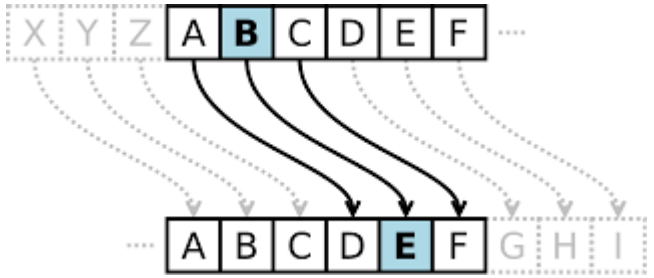
A kiosztás a legnagyobb probléma.



Blokk rejtjelezők

Definíció:

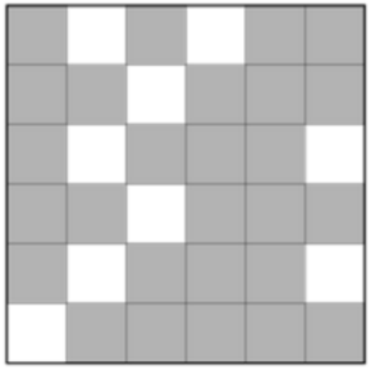
A nyílt szöveget t hosszúságú, A-beli elemekből álló láncokra bontja, és egyszerre egy blokkot rejtjelez.



Sokféle algoritmus van és sokféle felhasználási mód.

Egyszerű helyettesítéses (substitution) és felcseréléses (transposition) rejtjelezők és ezek kombinációi. Még ma is ezek a leghatékonyabb rejtjelezők (csak korszerű módon).

E	J	E	U	S	T
Y	G	L	É	Á	N
Á	E	Y	S	É	S
C	N	V	D	Í	B
O	E	R	M	M	R
N	E	Ű	N	R	Á



Adatfolyam rejtjelezők

Definíció:

Szimbólumonként (bit vagy karakter) rejtjelez. Szimbólumról-szimbólumra változó kulccsal.

kulcsfolyam: $e_1\, e_2\, e_3 \ldots e_n$

nyílt szöveg: $m_1\, m_2\, m_3 \ldots m_n$

rejtjelezett szöveg: $c_1\, c_2\, c_3 \ldots c_n$

rejtjelezés: $c_i = E_{ei}(m_i)$

dekódolás: $m_i = D_{di}(c_i)$

A transzformáció egyszerű, a kulcsfolyam generálása nehéz: teljesen véletlen, vagy kiinduló kulcsból generált folyam

Vernam rejtjelező:

A={0,1}

c=m XOR e

e=d

One-time-pad: Vernam rejtjelező teljesen véletlen kulcsfolyammal. Ez bizonyíthatóan nem feltörhető

Blokk vagy adatfolyam rejtjelező?

Minden blokk rejtjelezőből készíthető adatfolyam rejtjelező. (Blokk rejtjelezőknél tanuljuk.)

Adatfolyam rejtjelező alkalmazandó, ha:

- kis méretű vagy gyors hw megoldás kell (blokk rejtjelező alapúra nem áll fönn)
- nem lehet pufferelni, bitenként (karakterenként) kell dekódolni, várakozás nélkül
- zajos környezetben, mert adatfolyam rejtjelezésnél nincs hibaterjedés (vagy korlátozott)

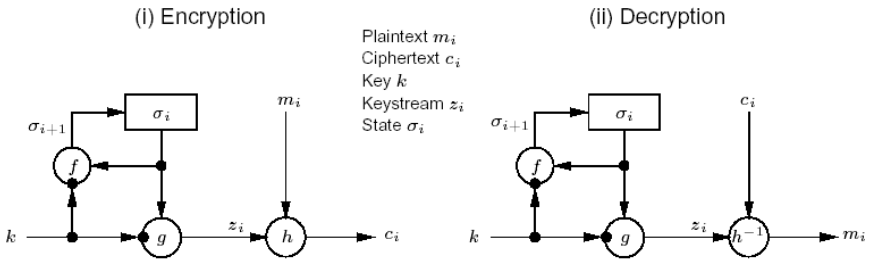
Távközlés (főleg a vezeték nélküli) általában adatfolyam rejtjelezést használ.

Kevés átfogó, széles körben alkalmazott, nyílt szabvány van adatfolyam rejtjelezőre.

Osztályozás

szinkron

álvéletlen bitsorozat a nyílt (rejtjelezett) szövegtől független

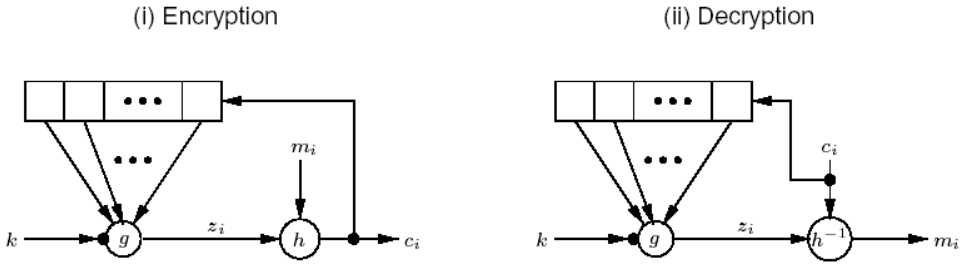


leggyakrabban: h = XOR

- szinkronizálni kell (reinicializáció, markerek, offszetek)
- bit módosulás esetén nincs hibaterjedés
- aktív támadás: bit módosítás a rejtjelezett üzenetben a nyílt üzenet bitet átbillenti

önszinkronizáló

álvéletlen bitsorozat a rejtjelezett üzenet bitjeiből



- szinkronizálja magát
- korlátozott hibaterjedés (törlés, beszúrás, módosulás esetén)

Szinkron adatfolyam rejtjelező = álvéletlen generátor?

Minden jó álvéletlen generátor jó adatfolyam kódoló is elvben.

Vagy mégse:

- gyorsaság az adatfolyam rejtjelező esetében fontosabb
- véletlenszerűség az álvéletlen generátor esetében fontosabb
- újra inicializáció az adatfolyam rejtjelező esetében fontosabb
- backtracking resistance az álvéletlen generátor esetében fontosabb

További osztályozás

LFSR alapú

HW megvalósíthatóság
nagy periódus
jó statisztikai tulajdonságok
algebrai eszközök

blokk rejtjelező alapú

minden blokk rejtjelezőhöz bizonyos működési mód esetén:

- CTR (Counter)
- OFB (Output Feedback)
- CFB (Cipher Feedback)

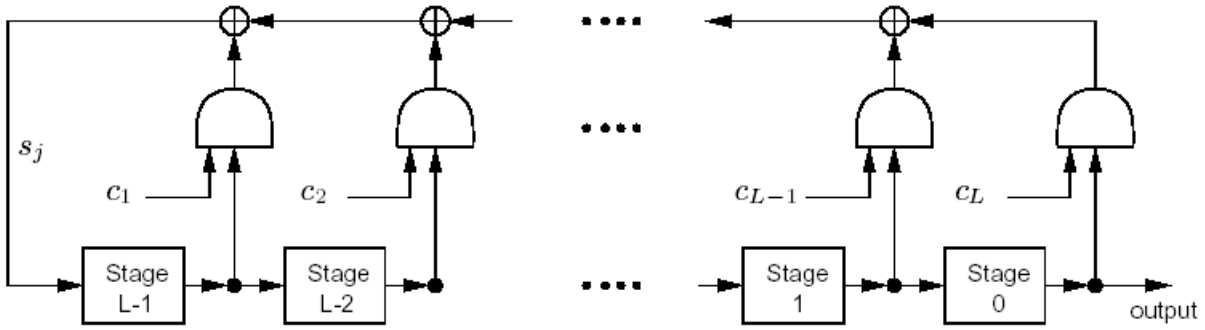
számelméleti probléma alapú

Blum-Blum-Shub (lassú)

dedikált

RC4
SEAL
SNOW
MUGI
PANAMA
TRIVIUM
stb.

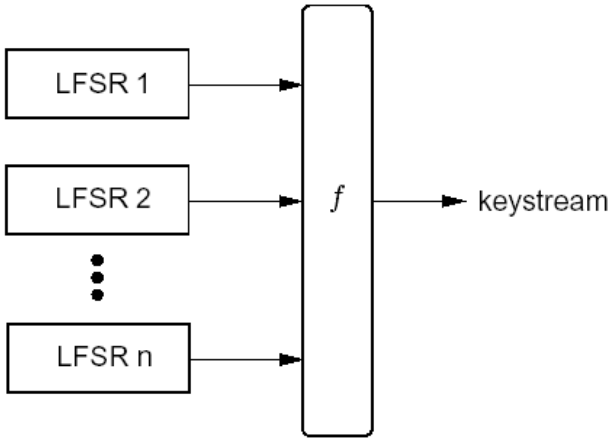
Linear Feedback Shift Registers

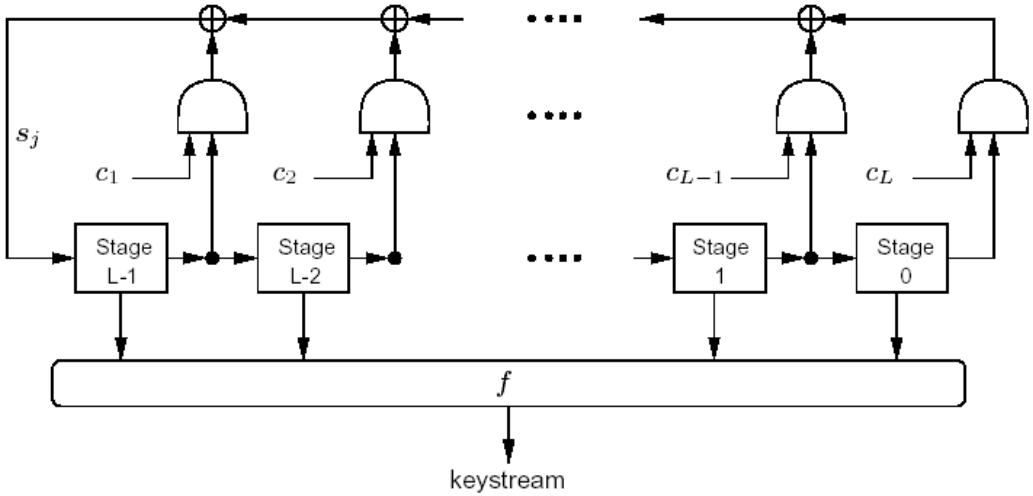


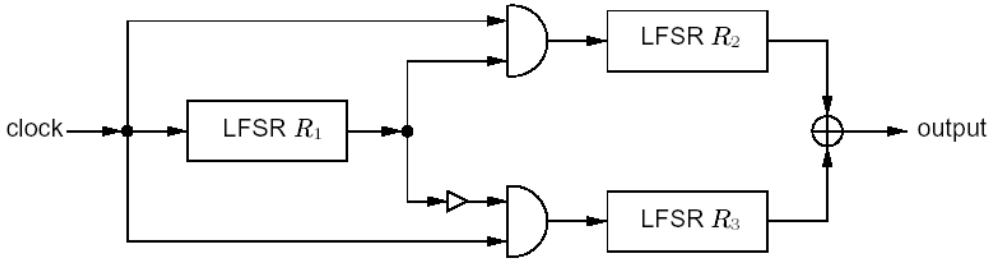
visszacsatolási (karakterisztikus) polinom:

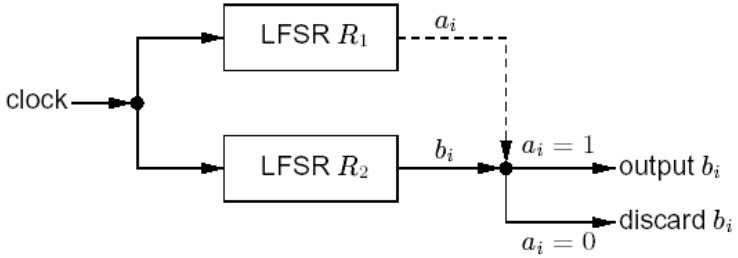
$$C(D) = 1 + c_1 D + c_2 D^2 + \cdots + c_L D^L \in \mathbb{Z}_2[D]$$

LFSR adatfolyam rejtjelezők









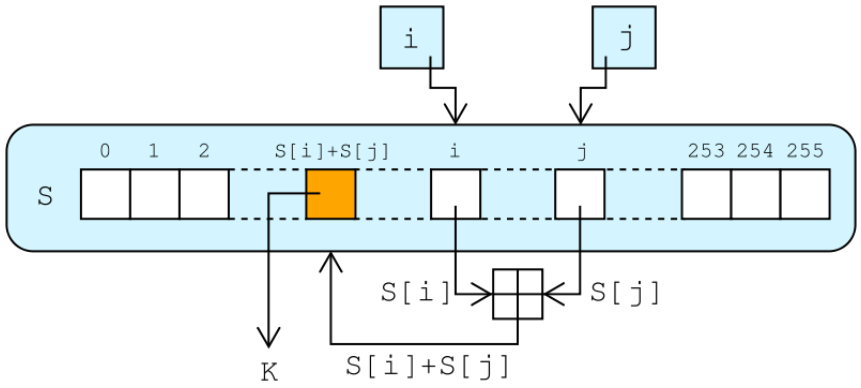
RC4

Rivest (1987), titkos algoritmus (RC2, RC4, RC5, RC6)

'94-ben nyilvánosságra került

Legszélesebb körben használt adatfolyam rejtjelező volt sokáig: WiFi, SSL/TLS, Secure Shell, Remote Desktop, PPP (point-to-point protocol)

Ma már nem biztonságos



inicializálás:

```
for i from 0 to 255
    S[i] := i
endfor
j := 0
for i from 0 to 255
    j := (j + S[i] + key[i mod keylength]) mod 256
    swap(&S[i],&S[j])
endfor
```

bit generálás:

```
i := 0
j := 0
while GeneratingOutput:
    i := (i + 1) mod 256
    j := (j + S[i]) mod 256
    swap(&S[i],&S[j])
    output S[(S[i] + S[j]) mod 256]
endwhile
```

RC4 biztonság

WEP (WiFi) problémák oka nem az RC4 volt. WPA TKIP továbbra is RC4-et használ.

2011: BEAST attack on TLS 1.0. Az SSL/TLS protokollok szimmetrikus kulcsú titkosítását érinti, ha az blokk kódoló CBC módban, vagyis gyakorlatilag a nem RC4 alapú SSL/TLS verziókat mind. Emiatt még inkább előtérbe került az RC4 használata.

2013: Nagyon nagyszámú TLS csomag lehallgatása esetére van törés. Ez a gyakorlatban nem megvalósítható, de a Snowden ügy és 2015-ben megjelent praktikusabb törések miatt egyre több a figyelmeztető jel.

2015 óta tilos használni a TLS protokollban (RFC 7465)

Szabványok

Általános célú adatfolyam rejtjelező szabvány 2001-ig nem volt.

Egyes alkalmazási területek kapcsán:

GSM: A5/1, A5/2, A5/3

GPRS: GEA3

UMTS/3GPP: f8

Bluetooth: E0

IEEE 802.11 (WiFi): RC4

GSM

A5/1: erős

A5/2: gyenge

'80-as évek közepe, titkos algoritmus

'90-es években publikussá vált

ma már mindkettő gyenge

A5/3: ETSI szabvány

a kompatibilitás veszélyezteti a biztonságot

GPRS

GEA3: A5/3-hoz hasonló

UMTS/3GPP

f8: A5/3-hoz hasonló

A5/1

clock-controlled generator, 3 LFSR

$$x^{19} \oplus x^5 \oplus x^2 \oplus x \oplus 1$$

$$x^{22} \oplus x \oplus 1$$

$$x^{23} \oplus x^{15} \oplus x^2 \oplus x \oplus 1$$

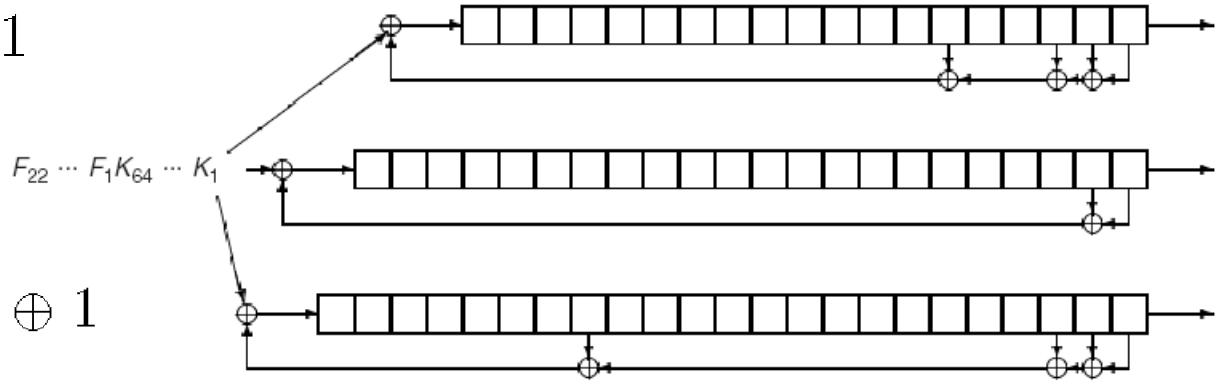


Fig. 1. Initialization of the A5/1 running-key generator

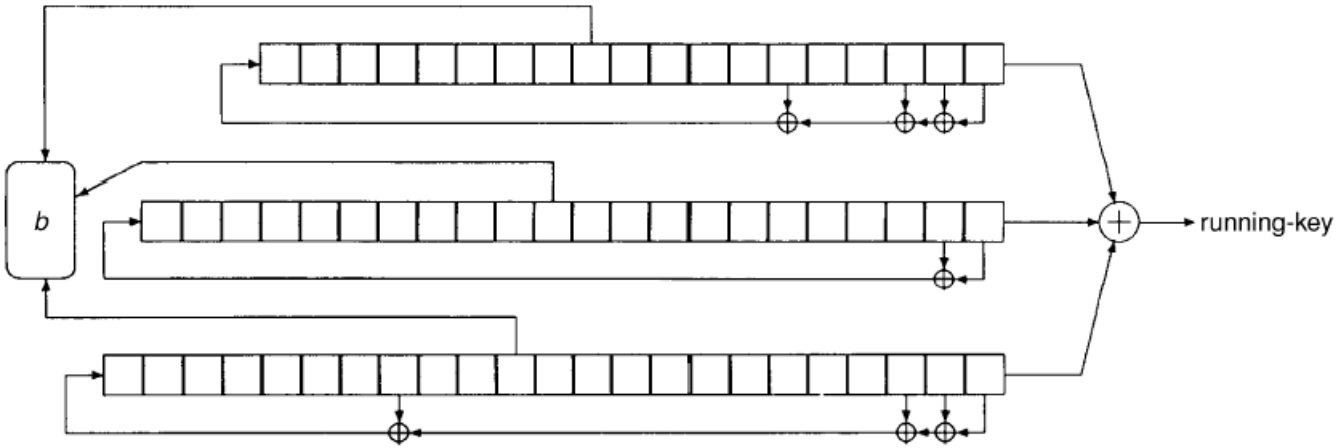


Fig. 2. A5/1 running-key generator

A5/2

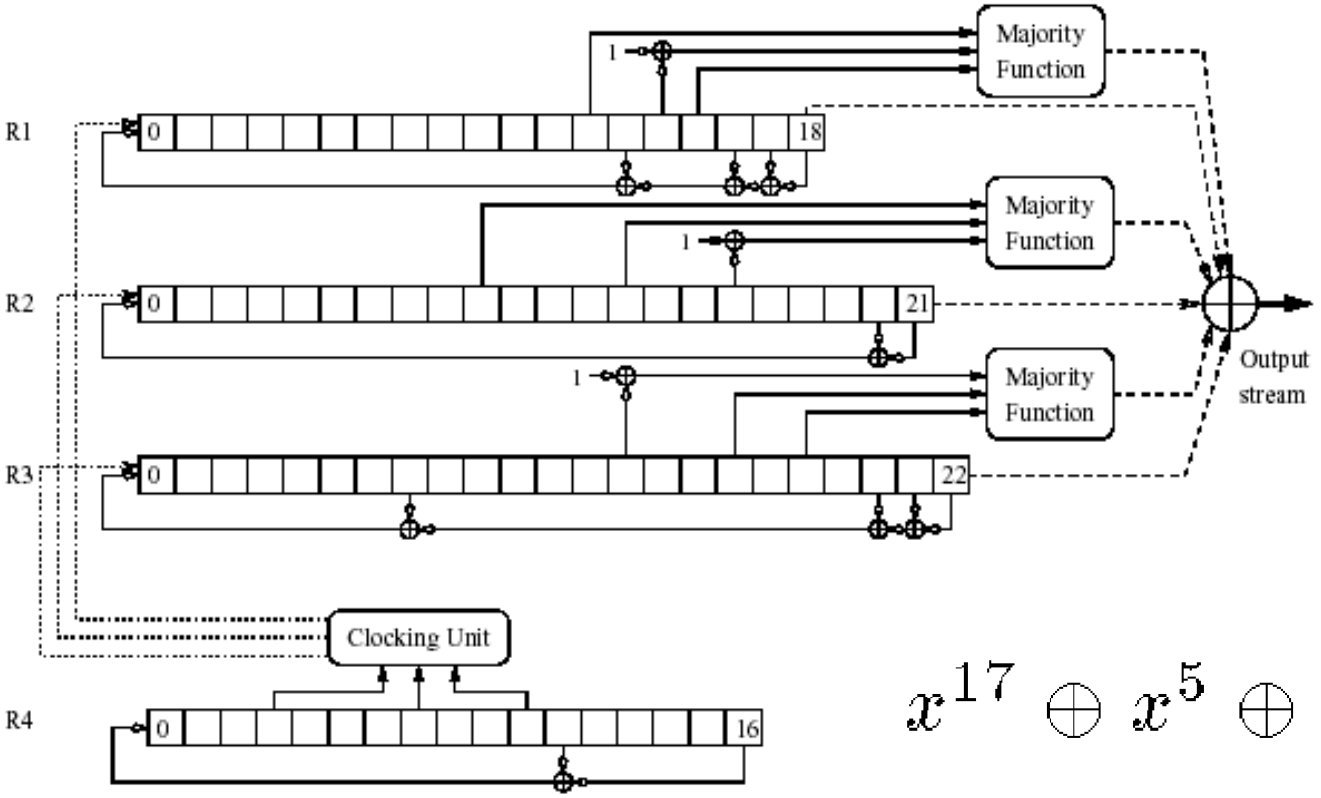


Fig. 1. The A5/2 internal structure

f8 és A5/3

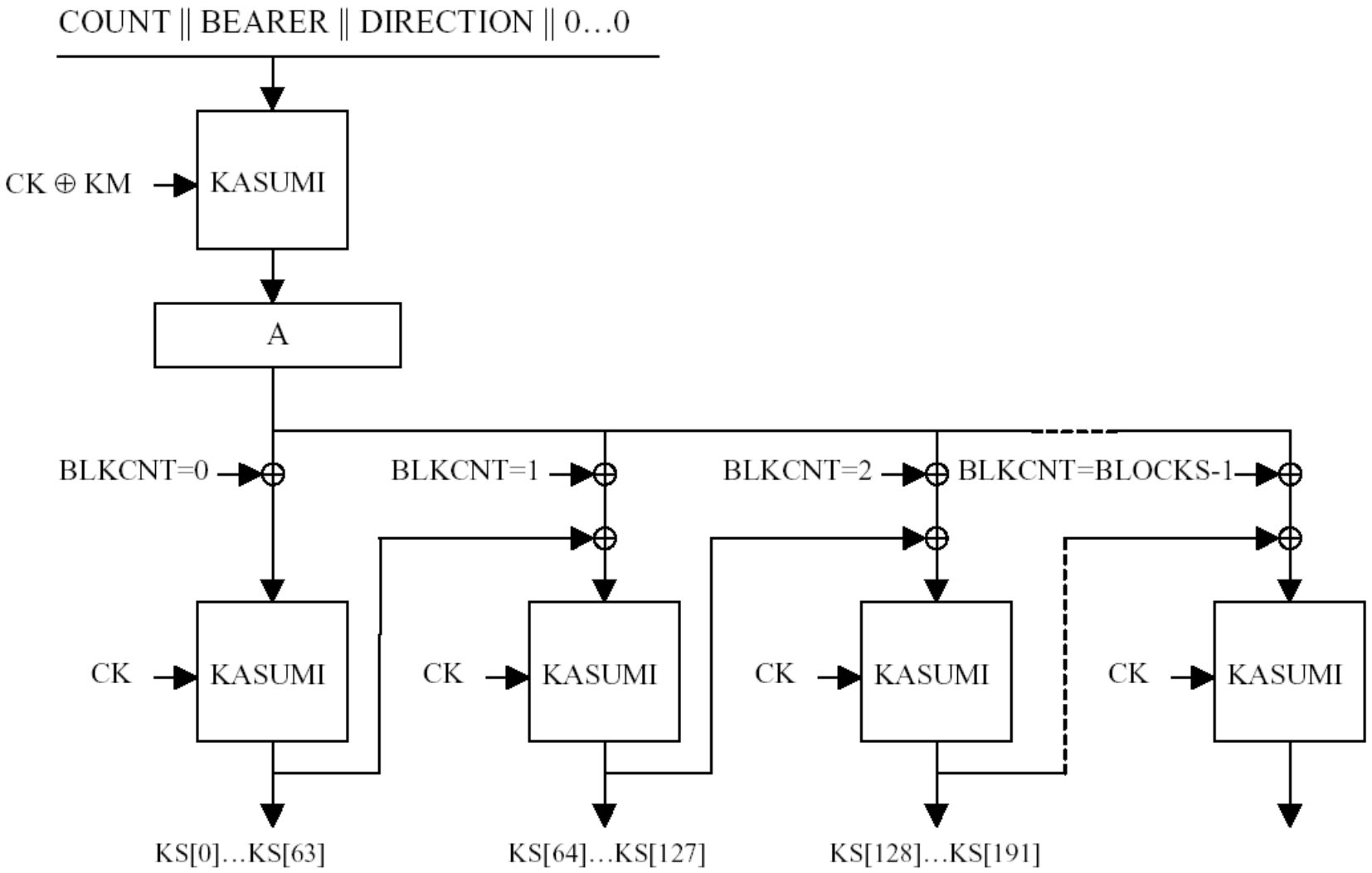
Blokk rejtjelező (KASUMI)
Output Feedback (OFB) mode-ban

KASUMI:
64 bit input
128 bit kulcs
64 bit output

f8:
input: 128 bit seed
<64 bit üzenetazonosító
output: 1-5114 álvéletlen bit

A5/3:
input: 64..128 bites kulcs kiegészítve
<64 bit üzenetazonosító
output: 2 x 114 álvéletlen bit

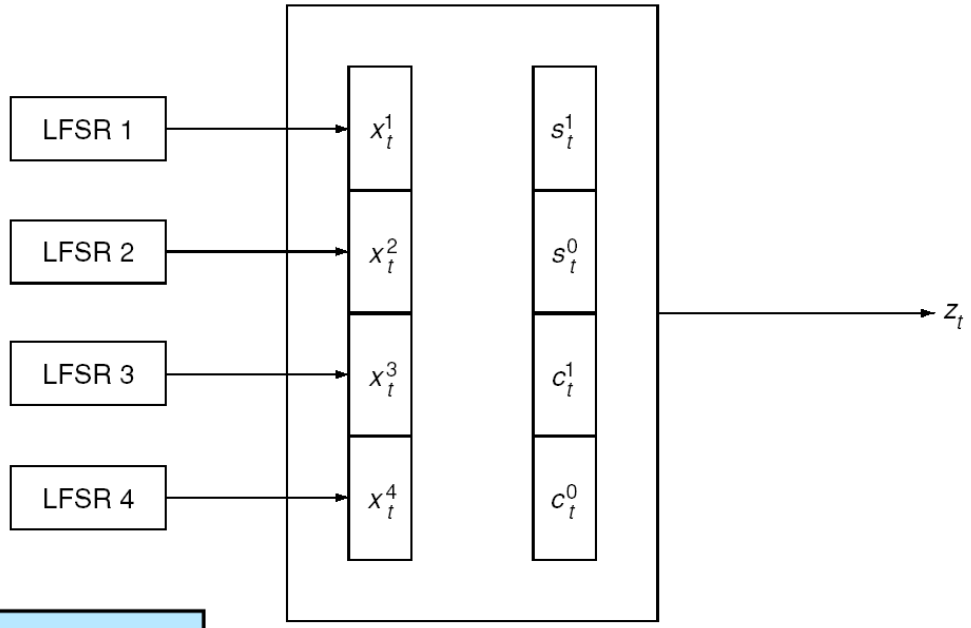
3GPP rejtjelező (f8)



Bluetooth rejtjelező (E0)

Summation generator

4 LFSR



i	L_i	feedback $f_i(t)$	weight
1	25	$t^{25} + t^{20} + t^{12} + t^8 + 1$	5
2	31	$t^{31} + t^{24} + t^{16} + t^{12} + 1$	5
3	33	$t^{33} + t^{28} + t^{24} + t^4 + 1$	5
4	39	$t^{39} + t^{36} + t^{28} + t^4 + 1$	5

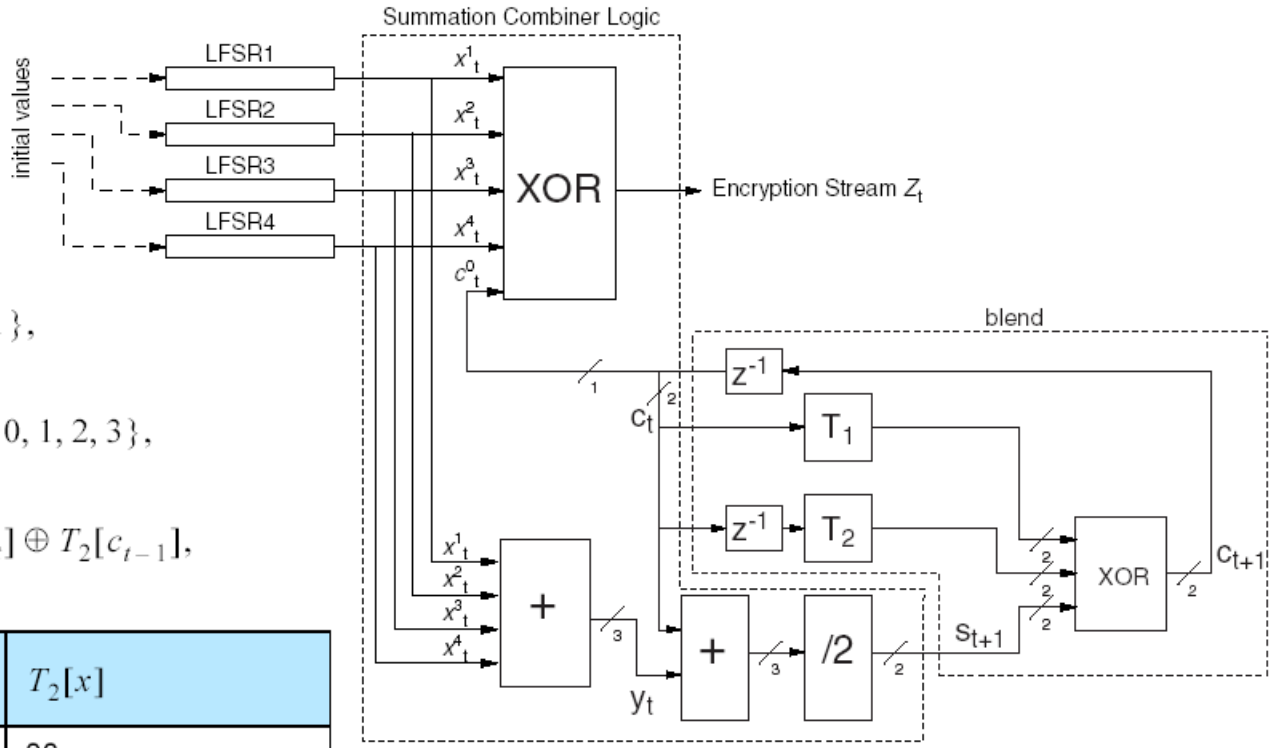
Bluetooth rejtjelező (E0)

$$y_t = \sum_{i=1}^{4} x_t^i,$$

$$z_t = x_t^1 \oplus x_t^2 \oplus x_t^3 \oplus x_t^4 \oplus c_t^0 \in \{0, 1\},$$

$$s_{t+1} = (s_{t+1}^1, s_{t+1}^0) = \left\lfloor \frac{y_t + c_t}{2} \right\rfloor \in \{0, 1, 2, 3\},$$

$$c_{t+1} = (c_{t+1}^1, c_{t+1}^0) = s_{t+1} \oplus T_1[c_t] \oplus T_2[c_{t-1}],$$



x	$T_1[x]$	$T_2[x]$
00	00	00
01	01	11
10	10	01
11	11	10

IEEE 802.11 (WiFi)

WEP (wired equivalent privacy) problémák

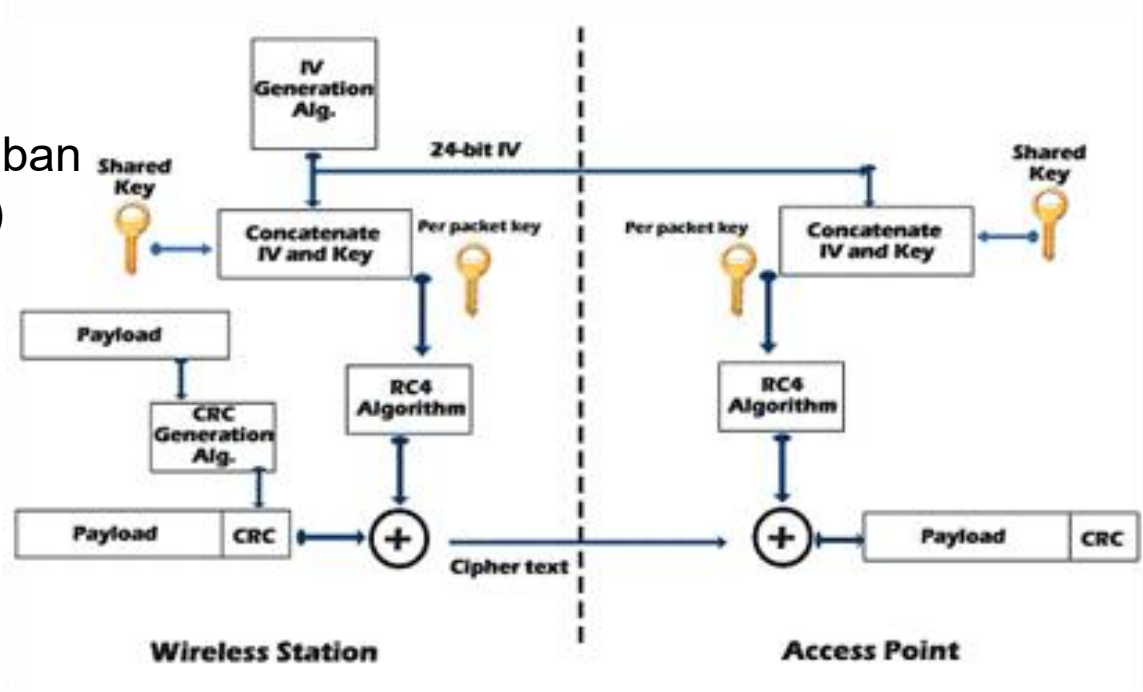
RC4 szinkron -> csomagonként kell szinkronizálni (új kulccsal inicializálni, mert egy kulcsot tilos kétszer használni - WEP amúgy megengedné)

nincs kulcsegyeztetés -> előre megosztott kulcs van (általában közös az egész hálózatban)

ezt egészíti ki egy 24 bites IV egyszerűen hozzáfűzve

WPA (WiFi protected access): RC4, de IV hozzáfűzés helyett keverve

WPA2: RC4 helyett AES



ISO/IEC 18033

2001-ben indult

ISO/IEC 18033: IT security techniques - Encryption algorithms

Part 1: General

Part 2: Asymmetric ciphers

Part 3: Block ciphers

Part 4: Stream ciphers

Part 4 osztályozás

keystream generator

- synchronous
  - blokk kódoló alapján (2)
  - dedikált (2)
- self-synchronizing
  - blokk kódoló alapján (1)

output function

- XOR
- MULTI-S01 (integritást is védi)

ISO/IEC 18033 KG

blokk kódoló alapján

Ismert technikák:

- CTR (Counter)
- OFB (Output Feedback)
- CFB (Cipher Feedback)

modes of operation

A szabvány mindegyiket megengedi

dedikált

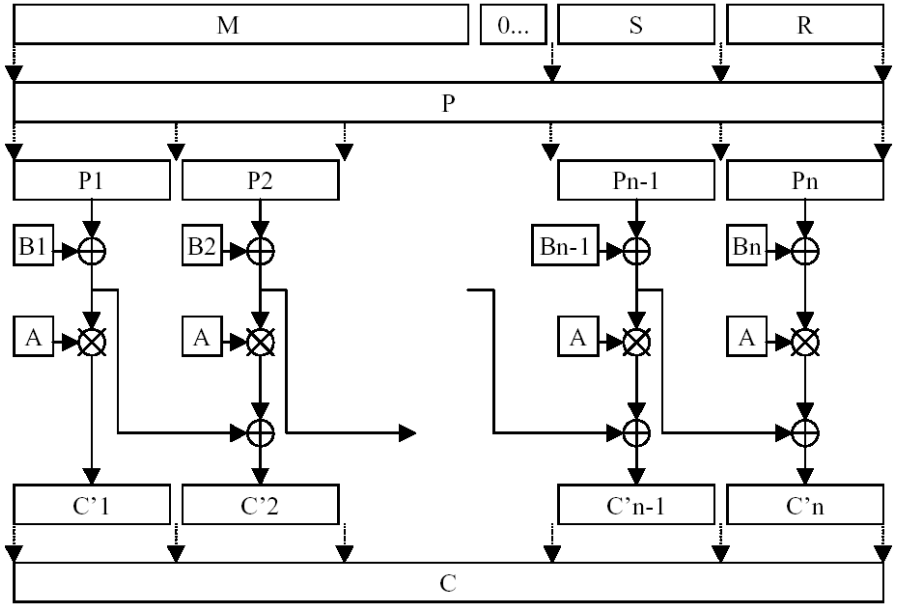
SNOW 2.0

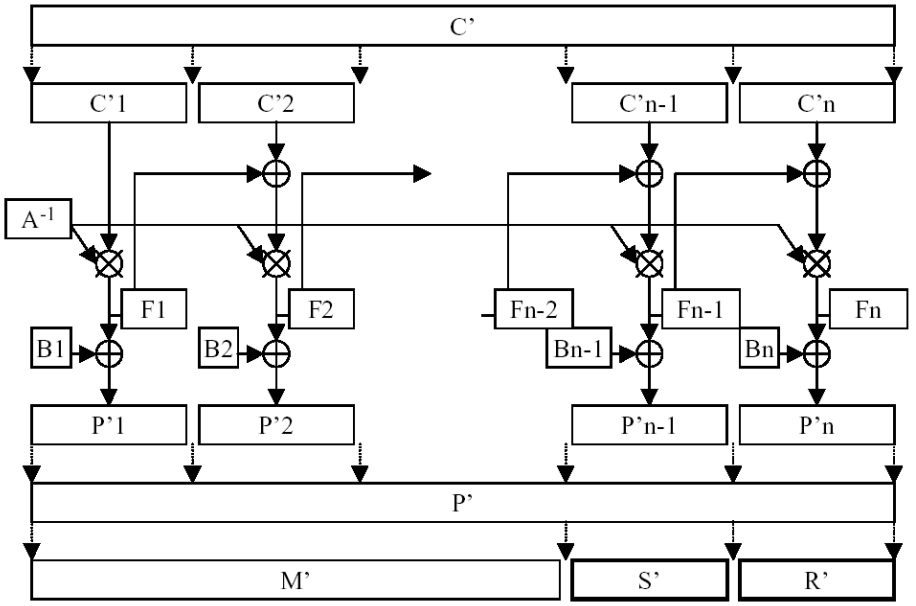
MUGI (elődje: PANAMA)

De:

- mindkettő 2002-es
- nincs self-synchronizing

ISO/IEC 18033 MULTI-S01





Encryption

Input:	Message	M ($64 \times n$ bits)
	Redundant data	R (64 bits)
	Secret key	A ($\neq 0$, 64 bits)
		B_i ($1 \leq i \leq n+2$, each 64 bits)
		S (64 bits)
Output:	Ciphertext	C (($n+2$) $\times$ 64 bits)

$$F_i = P_i \oplus B_i,$$
$$C_i = A \otimes F_i \oplus F_{i-1}.$$

Decryption

Input:	Ciphertext	C ($n' \times 64$ bits)
	Redundant data	R (64 bits)
	Secret key	A ($\neq 0$, 64 bits)
		B_i ($1 \leq i \leq n'$, each 64 bits)
		S (64 bits)
Output:	Message	Reception error or message M ($64 \times (n-2)$ bits)

$$F_i = A^{-1} \otimes (C_i \oplus F_{i-1}),$$
$$P_i = F_i \oplus B_i.$$

$F_0 = 0$, and symbols $\oplus$ and $\otimes$ represent addition and multiplication in finite field $\mathbf{F}2^{64}$

eSTREAM projekt

4 éves (2004-2008) EU kutatás:

"to identify new stream ciphers that might become suitable for widespread adoption"

Előzménye: Megelőző NESSIE projekt (2000-2003) nem eredményezett új adatfolyam rejtjelezőt

két kategória:

- gyors sw implementáció
- korlátozott képességű hw-re

Eredmény:

sw:

- HC-128
- Rabbit
- Salsa20/12
- SOSEMANUK

hw:

- Grain
- MICKEY
- Trivium

TRIVIUM

for $i = 1$ to N **do**

$t_1 \leftarrow s_{66} + s_{93}$

$t_2 \leftarrow s_{162} + s_{177}$

$t_3 \leftarrow s_{243} + s_{288}$

$z_i \leftarrow t_1 + t_2 + t_3$

$t_1 \leftarrow t_1 + s_{91} \cdot s_{92} + s_{171}$

$t_2 \leftarrow t_2 + s_{175} \cdot s_{176} + s_{264}$

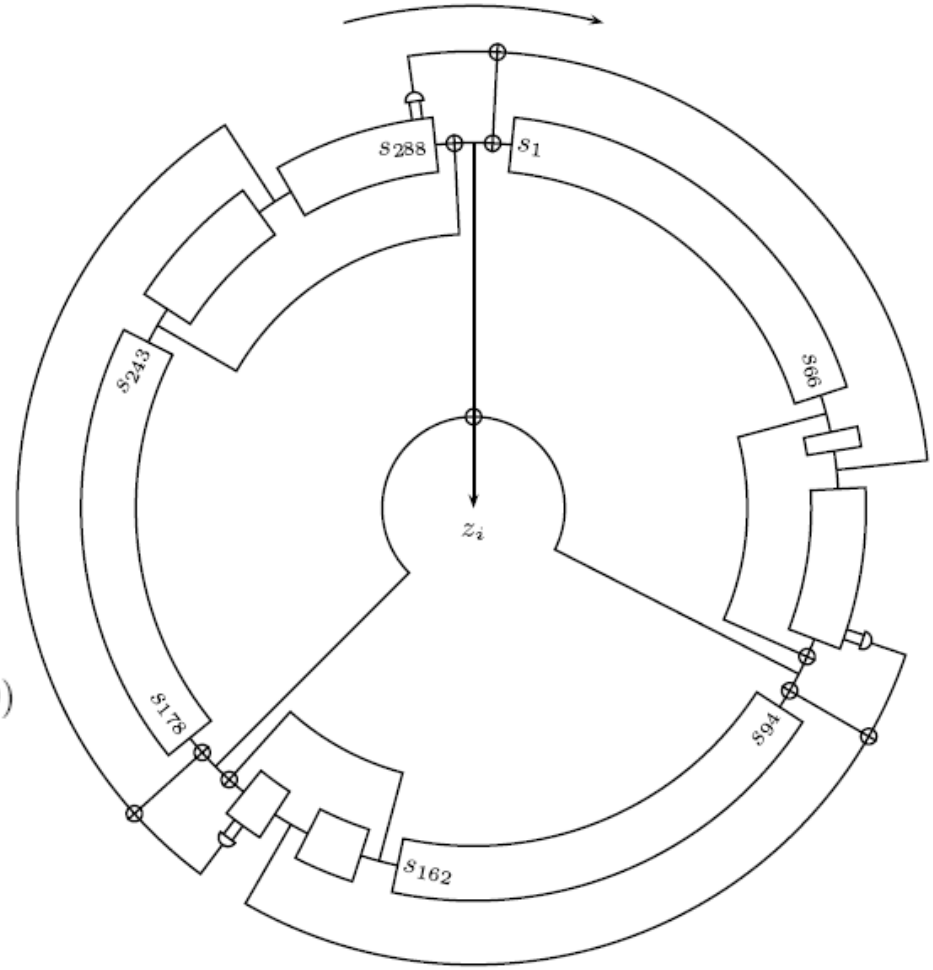
$t_3 \leftarrow t_3 + s_{286} \cdot s_{287} + s_{69}$

$(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$

$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176})$

$(s_{178}, s_{279}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$

end for



$(s_1, s_2, \ldots, s_{93}) \leftarrow (K_1, \ldots, K_{80}, 0, \ldots, 0)$

$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (IV_1, \ldots, IV_{80}, 0, \ldots, 0)$

$(s_{178}, s_{279}, \ldots, s_{288}) \leftarrow (0, \ldots, 0, 1, 1, 1)$

for $i = 1$ to $4 \cdot 288$ **do**

$t_1 \leftarrow s_{66} + s_{91} \cdot s_{92} + s_{93} + s_{171}$

$t_2 \leftarrow s_{162} + s_{175} \cdot s_{176} + s_{177} + s_{264}$

$t_3 \leftarrow s_{243} + s_{286} \cdot s_{287} + s_{288} + s_{69}$

$(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$

$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176})$

$(s_{178}, s_{279}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$

end for

Parameters	
Key size:	80 bit
IV size:	80 bit
Internal state:	288 bit

Salsa20

eSTREAM-nek benyújtotta Daniel J. Bernstein 2005-ben

állapotvektor: 16 db 32 bites word

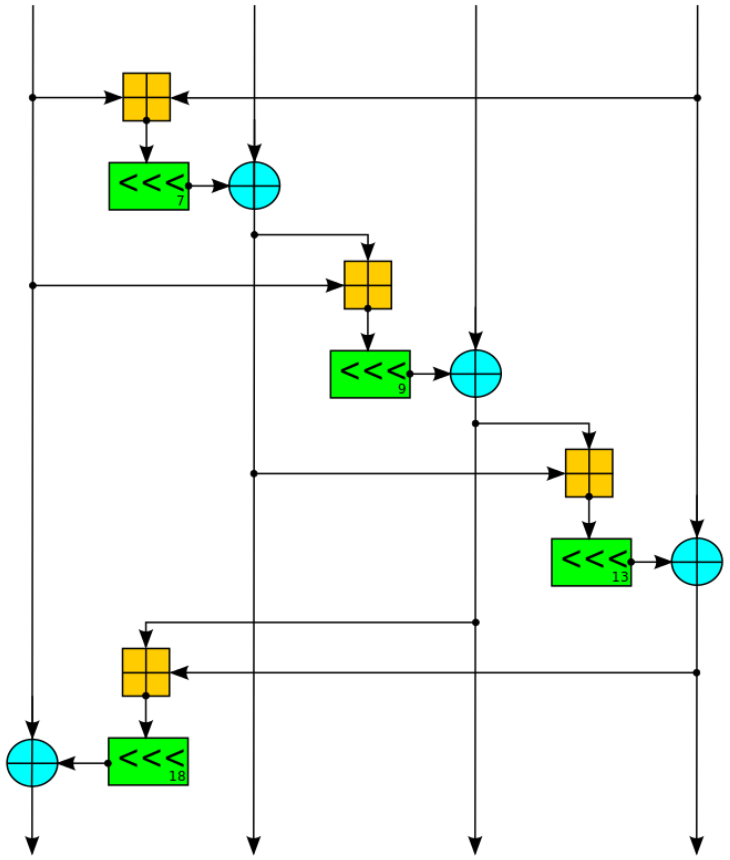
4 (párhuzamosítható) körben, egyszerre 4 wordre:

⊕: bitwise exclusive OR

⊞: 32-bit addition mod 2^{32}

<<<: bit rotáció

Egy változatát (ChaCha20) a Google használja: Android, Chrome



Bitcoin

PPKE, ITK
Csapodi Márton

Bitcoin book

Andreas M. Antonopoulos: Mastering Bitcoin

First edition (2014)

Second edition (2017) online:
https://github.com/bitcoinbook/bitcoinbook/blob/develop/book.asciidoc

Must read (learn):
1. Introduction
2. How bitcoin works
4. Keys, Addresses (some)
6. Transactions (some more)

Quick Start

Chose your wallet:

- platform: desktop, mobile, web, hardware, paper wallet
- capabilities: full node (stores the whole blockchain), lightweight (creates and validates transactions), third-party API (uses third party to create and validate transactions)

Install and start your wallet:

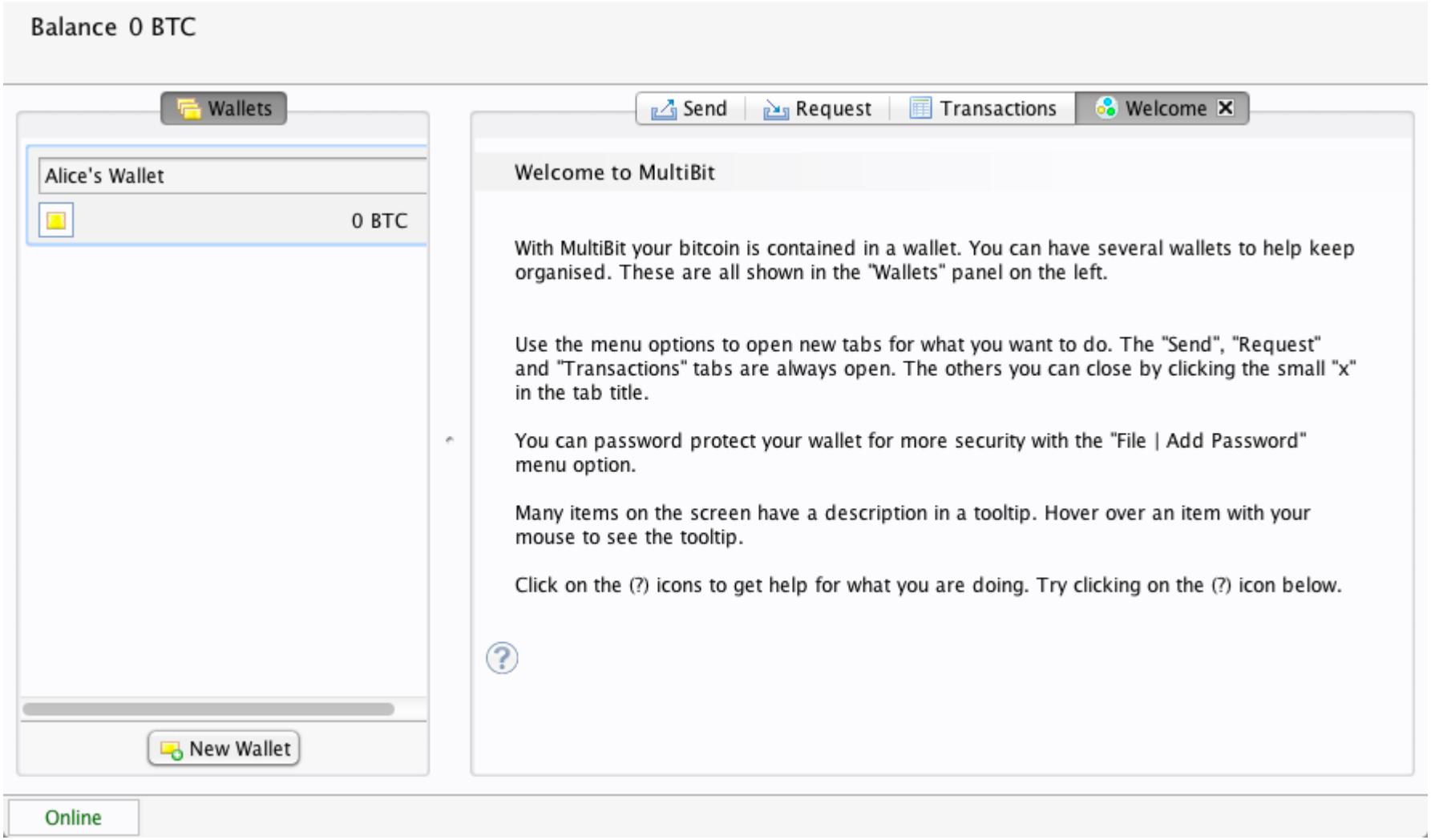
- bitcoin address and keys

Get your first bitcoin:

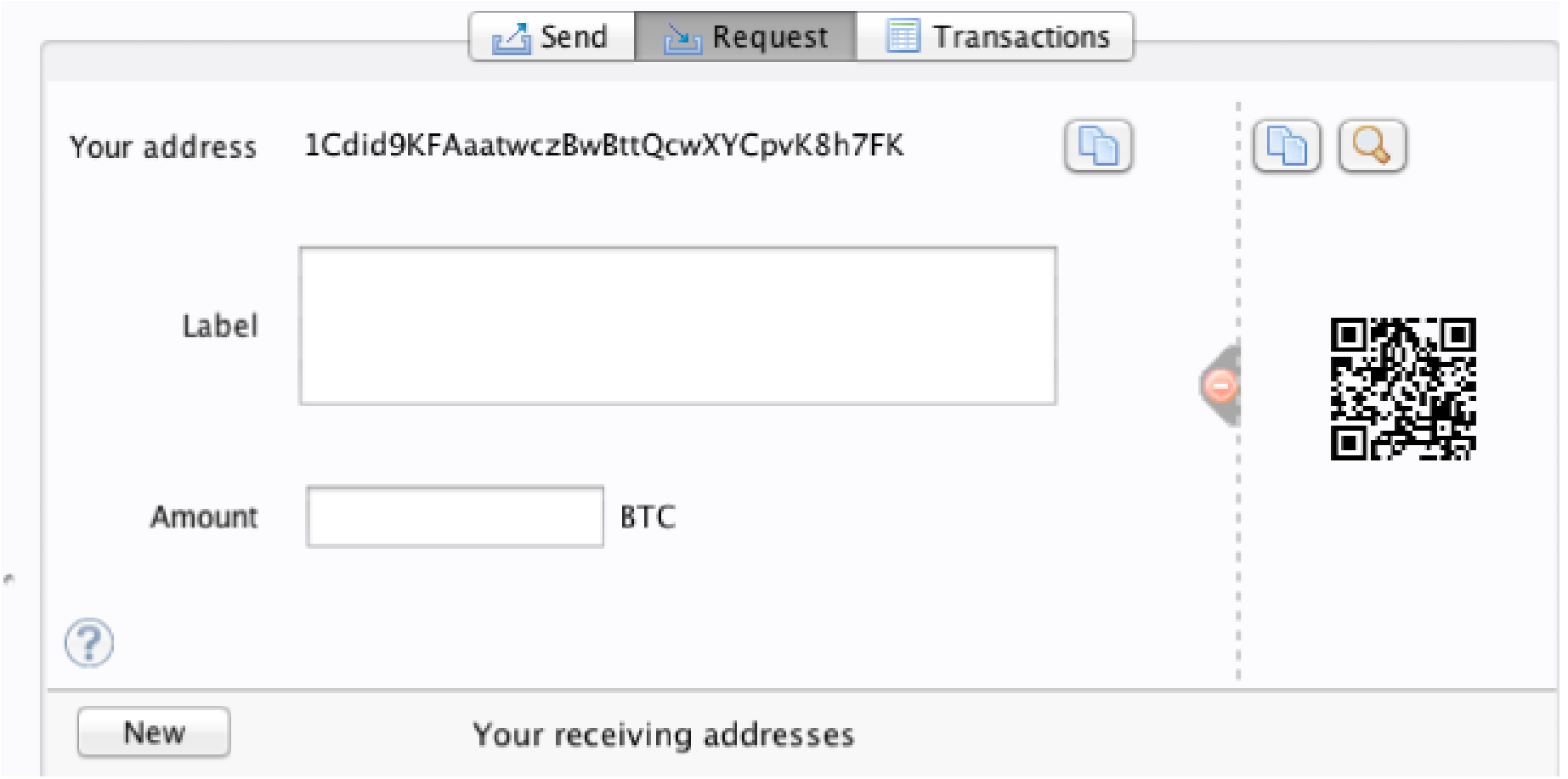
- buy (form friend, local seller, ATM, bitcoin exchange)
- earn (by selling a product/service)

Start sending and receiving bitcoins

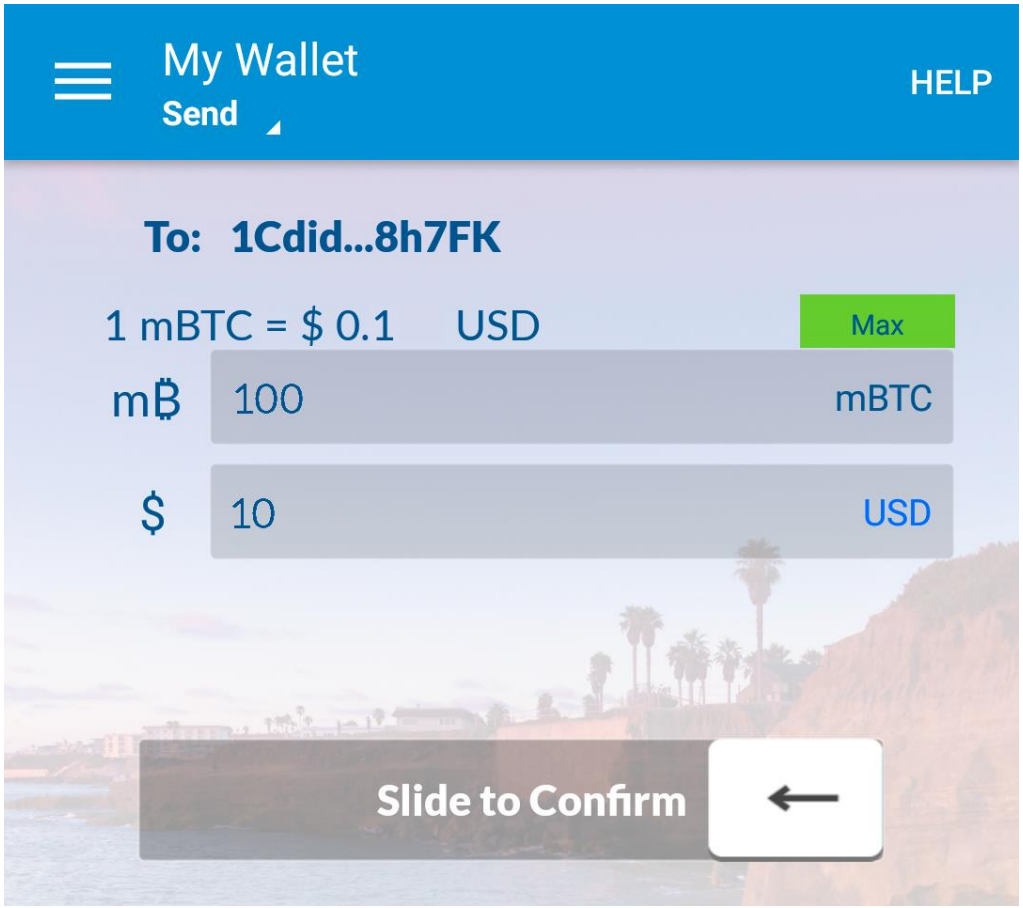
Bitcoin wallet



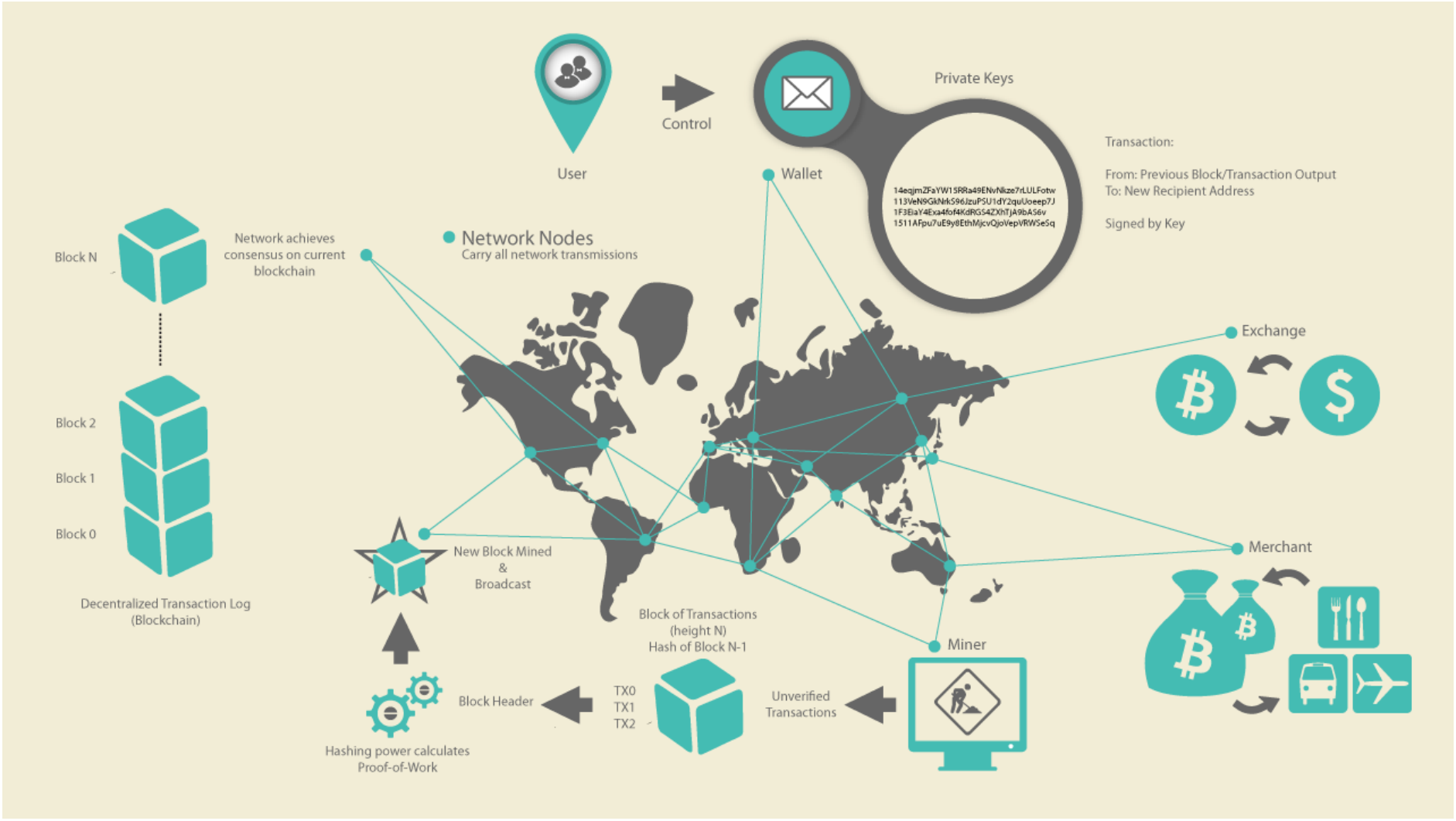
Bitcoin wallet



Bitcoin wallet



Overview


User
Control
Private Keys
Wallet
14eqjmZFaYW15RRa49ENvNkze7rLULFotw
113VeN9GkNrkS96JzuPSU1dY2quUoeep7J
1F3EiaY4Exa4fof4KdRGS4ZXhTjA9bAS6v
1511AFpu7uE9y8EthMjcvQjoVepVRWSeSq
Transaction:
From: Previous Block/Transaction Output
To: New Recipient Address
Signed by Key
Block N
Network achieves consensus on current blockchain
Network Nodes
Carry all network transmissions
Exchange
Block 2
Block 1
Block 0
Decentralized Transaction Log (Blockchain)
New Block Mined & Broadcast
Merchant
Block of Transactions (height N) Hash of Block N-1
Miner
Block Header
TX0
TX1
TX2
Unverified Transactions
Hashing power calculates Proof-of-Work

Buy a cup of coffee

Payment request QR code:

Encodes the following:

- A bitcoin address: "1GdK9UzpHBzqzX2A9JFP3Di4weBwqgmoQA"
- The payment amount: "0.015"
- A label for the recipient address: "Bob's Cafe"
- A description for the payment: "Purchase at Bob's Cafe"

View Alice’s transaction on a block explorer site (blockchain.info):

https://blockchain.info/tx/0627052b6f28912f2703066a912ea577f2ce4da4caa5a5fbd8a57286c345c2f2

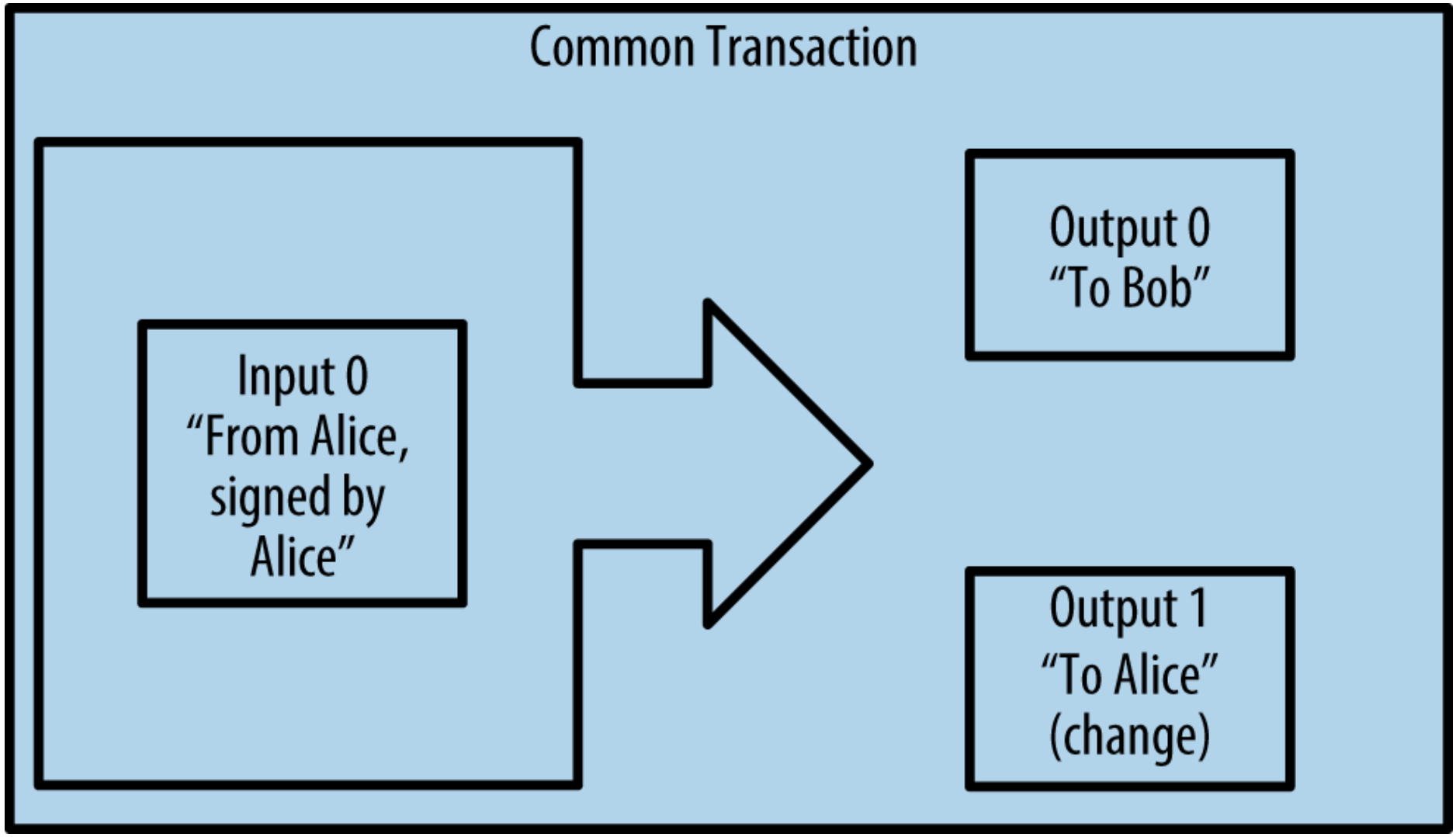
Transaction bookkeeping

Transaction as Double-Entry Bookkeeping

Inputs	Value	Outputs	Value
Input 1	0.10 BTC	Output 1	0.10 BTC
Input 2	0.20 BTC	Output 2	0.20 BTC
Input 3	0.10 BTC	Output 3	0.20 BTC
Input 4	0.15 BTC		
Total Inputs:	0.55 BTC	Total Outputs:	0.50 BTC

	Inputs	*0.55 BTC*
-	*Outputs*	*0.50 BTC*
	Difference	*0.05 BTC (implied transaction fee)*

Transaction1: pay + change



Pay and receive change

Transaction inputs cannot be divided.

If you buy an item that costs 5 bitcoin but only had a 20 bitcoin input to use:

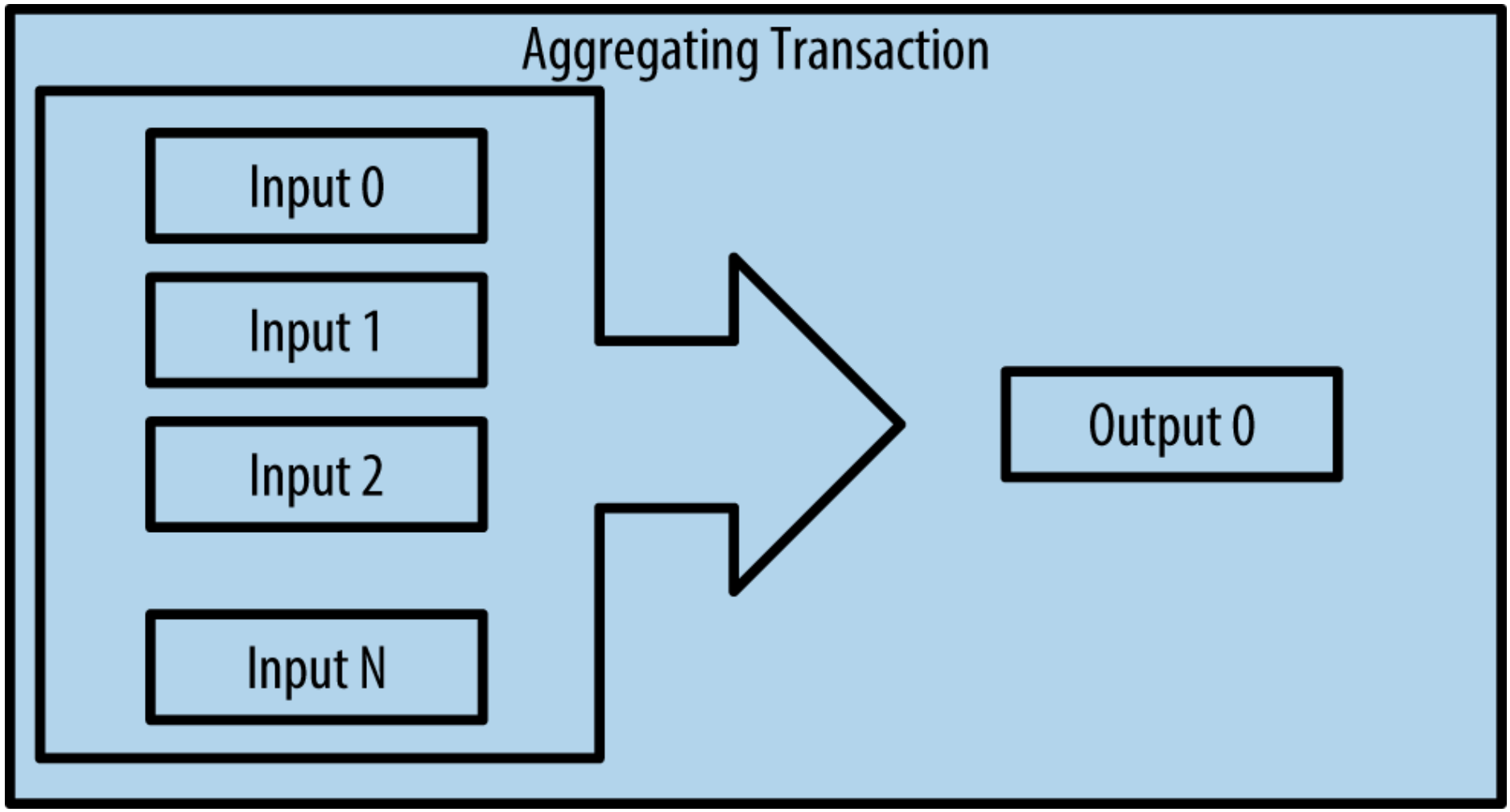
- you send one output of 5 bitcoin to the store owner and
- one output of 15 bitcoin back to yourself as change (or a little less: transaction fee)

Outputs add up to slightly less than inputs:

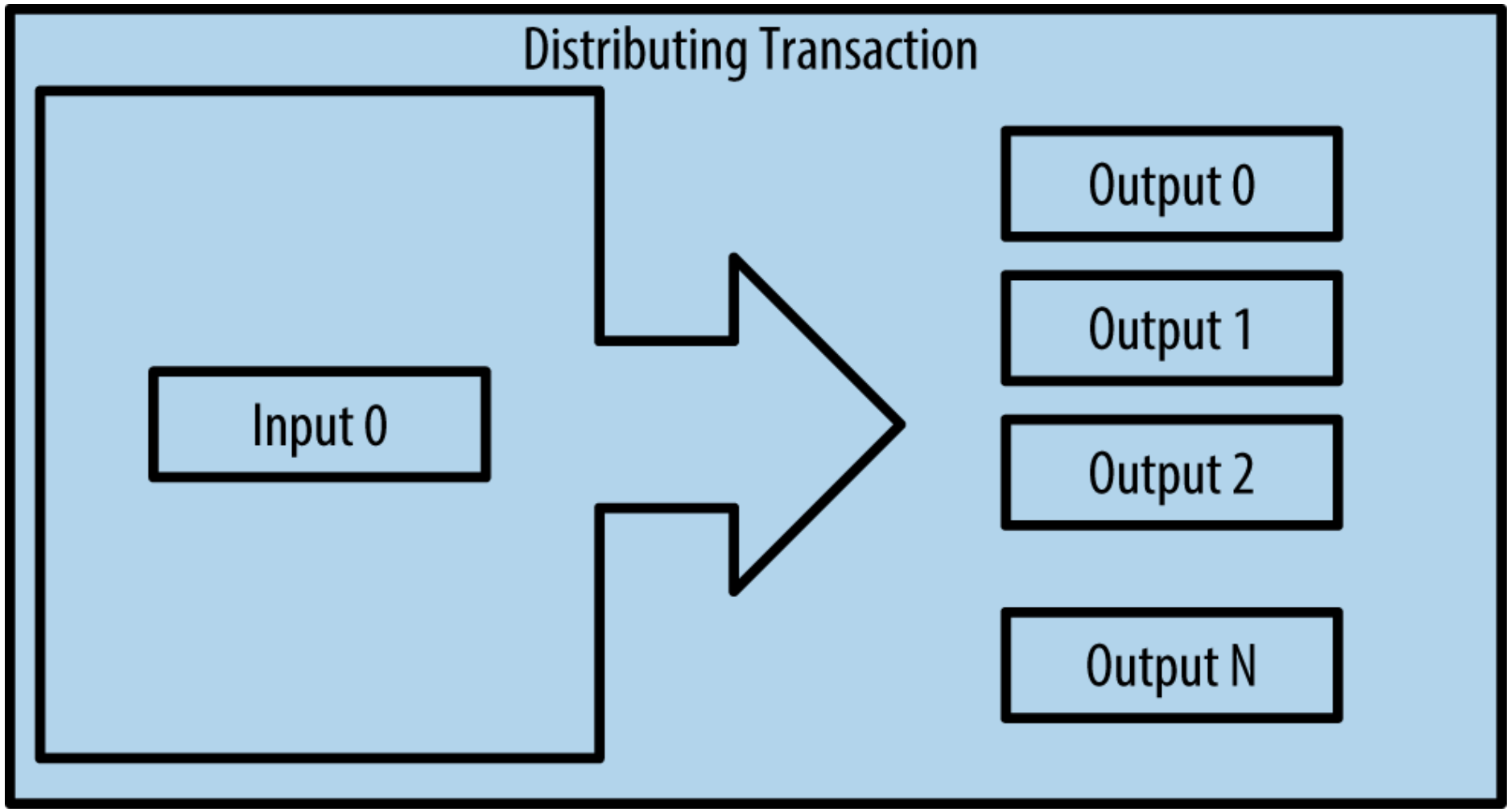
- the difference is the transaction fee, a small payment collected by the miner who includes the transaction in the ledger

Fees also serve as a security mechanism, by making it economically infeasible for attackers to flood the network with transactions.

Transaction2: clean up small amounts



Transaction3: multiple recipients



Create a transaction

Transaction View information about a bitcoin transaction

0627052b6f28912f2703066a912ea577f2ce4da4caa5a5fbd8a57286c345c2f2

1Cdid9KFAaatwczBwBttQcwXYCpvK8h7FK (0.1 BTC - Output)

1GdK9UzpHBzqzX2A9JFP3Di4weBwqgmoQA - (Unspent) 0.015 BTC

1Cdid9KFAaatwczBwBttQcwXYCpvK8h7FK - (Unspent) 0.0845 BTC

97 Confirmations | 0.0995 BTC

Summary	
Size	258 (bytes)
Received Time	2013-12-27 23:03:05
Included In Blocks	277316 (2013-12-27 23:11:54 +9 minutes)

Inputs and Outputs	
Total Input	0.1 BTC
Total Output	0.0995 BTC
Fees	0.0005 BTC
Estimated BTC Transacted	0.015 BTC

Find inputs

Keep track of your unspent coins:

- Full node wallet has all information
- Lightweight wallet keeps track of available outputs belonging to addresses in the wallet
- Or query the bitcoin network to retrieve this information

Look up all the unspent outputs for Alice's bitcoin address:

https://blockchain.info/unspent?active=1Cdid9KFAaatwczBwBttQcwXYCpvK8h7FK

Response includes:

- the reference to the transaction in which this unspent output is contained
- its value in satoshis (the smallest unit of the bitcoin currency, one hundred millionth of a single bitcoin, or 0.00000001 BTC)

Create outputs

First output: Payable to whoever can present a signature from the key corresponding to Bob's public address. This is Bob.

Second output: Change. Alice's wallet breaks her funds into two payments: one to Bob and one back to herself. She can then use (spend) the change output in a subsequent transaction.

For the transaction to be processed by the network in a timely fashion, Alice's wallet application will add a small fee. This is not explicit in the transaction; it is implied by the difference between inputs and outputs. The transaction fee is collected by the miner as a fee for validating and including the transaction in a block to be recorded on the blockchain.

The resulting transaction:

https://www.blockchain.com/btc/tx/0627052b6f28912f2703066a912ea577f2ce4da4caa5a5fbd8a57286c345c2f2

How the transaction propagates

The transaction contains all the information necessary to process, it does not matter how or where it is transmitted to the bitcoin network

The bitcoin network is a peer-to-peer network, with each bitcoin client participating by connecting to several other bitcoin clients. The purpose of the bitcoin network is to propagate transactions (and blocks) to all participants.

Any bitcoin node that receives a valid transaction it has not seen before will immediately forward it to all other nodes to which it is connected. This propagation technique is known as flooding. The transaction rapidly propagates out across the peer-to-peer network, reaching a large percentage of the nodes within a few seconds.

When the transaction (through other nodes) reaches Bob’s wallet, it checks for being an incoming transaction and verifies inputs. If the transaction is well formed, uses previously unspent inputs and contains sufficient transaction fees to be included in the next block, Bob can assume, with little risk, that the transaction will shortly be included in a block.

Small value transactions can be accepted without delay (a new block is created every 10 minutes).

Mining

The transaction (after propagation) becomes part of the blockchain by mining a new block.

The trust in bitcoin is based on computation. Mining a new block of transactions requires an enormous amount of computation, but the block can be easily verified. (like sudoku)

Mining nodes validate the transactions and receive mining reward (diminishes with time) plus the transaction fees.

The difficulty of mining is adjusted so that it takes approximately 10 minutes to find a solution.

The mining involves repeatedly hashing the header of the block and a random number with the SHA256 cryptographic algorithm until a solution matching a predetermined pattern emerges. Finding a solution (the so-called Proof-of-Work, or PoW) requires quadrillions of hashing operations per second across the entire bitcoin network. The first miner to find such a solution wins the round of competition and publishes that block into the blockchain.

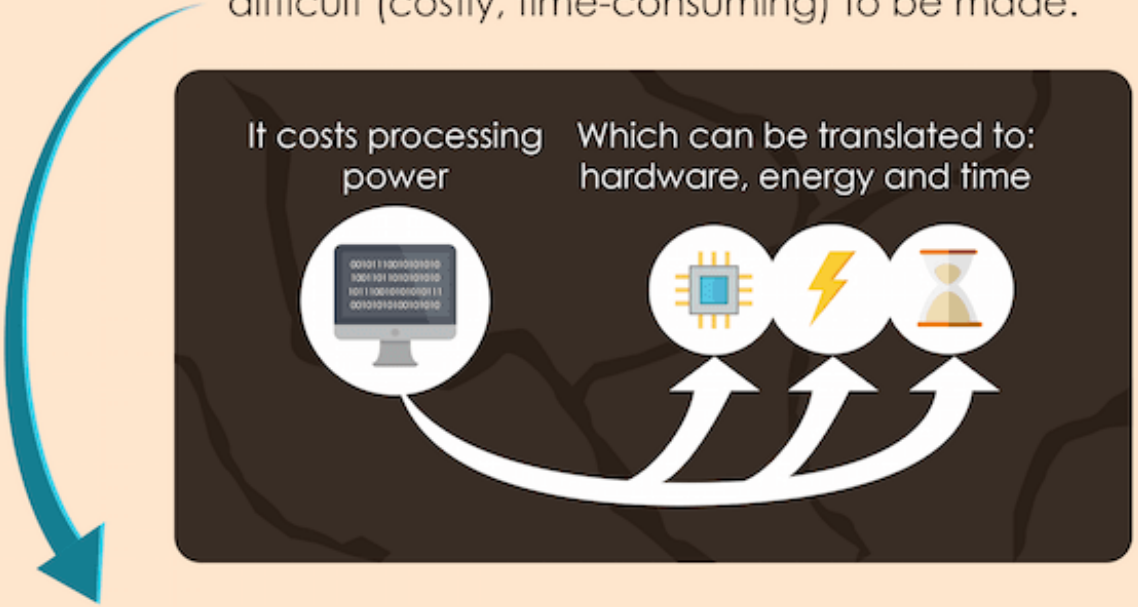
Proof of work

THE BITCOIN MINING SAGA - PART II

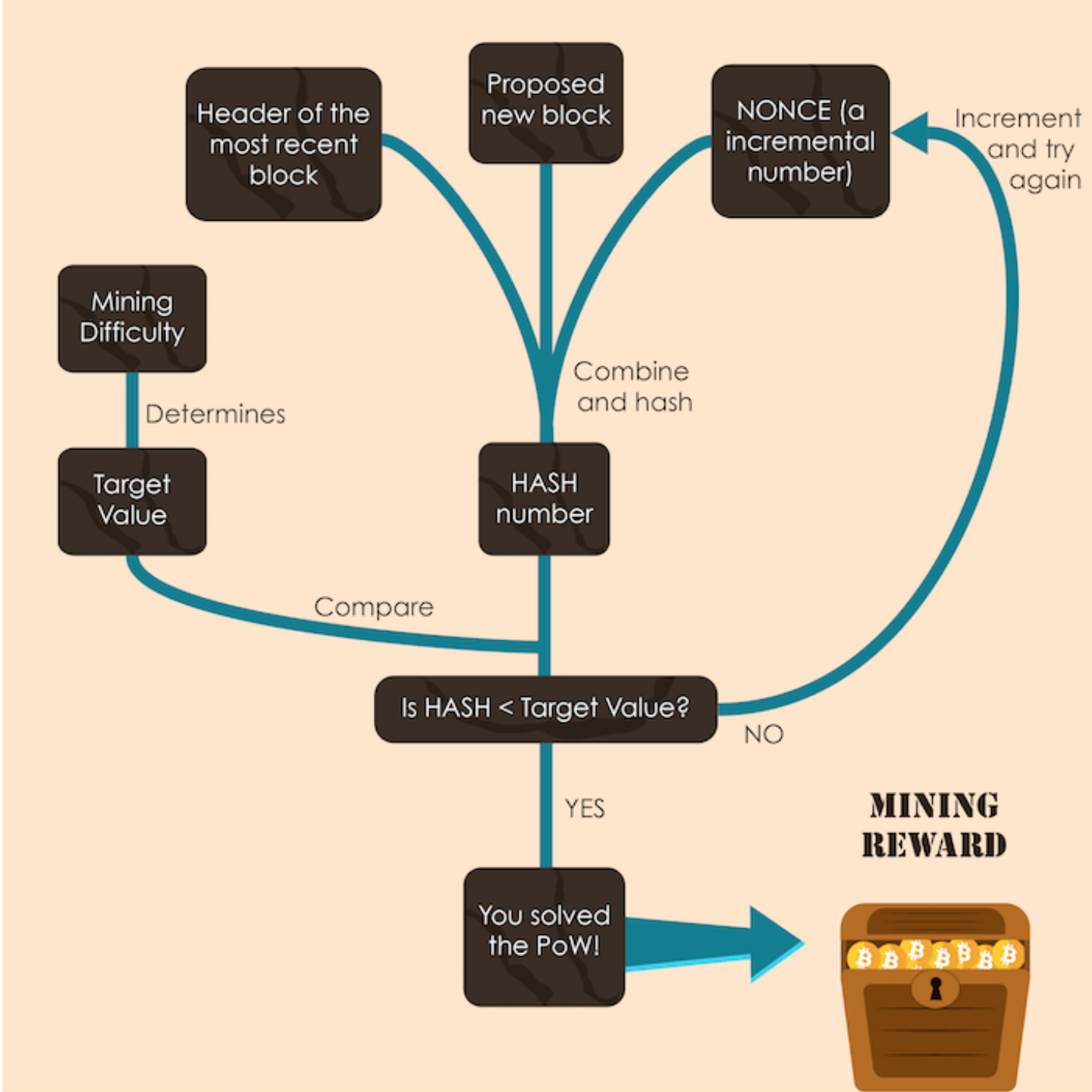
By Patrícia Estevão

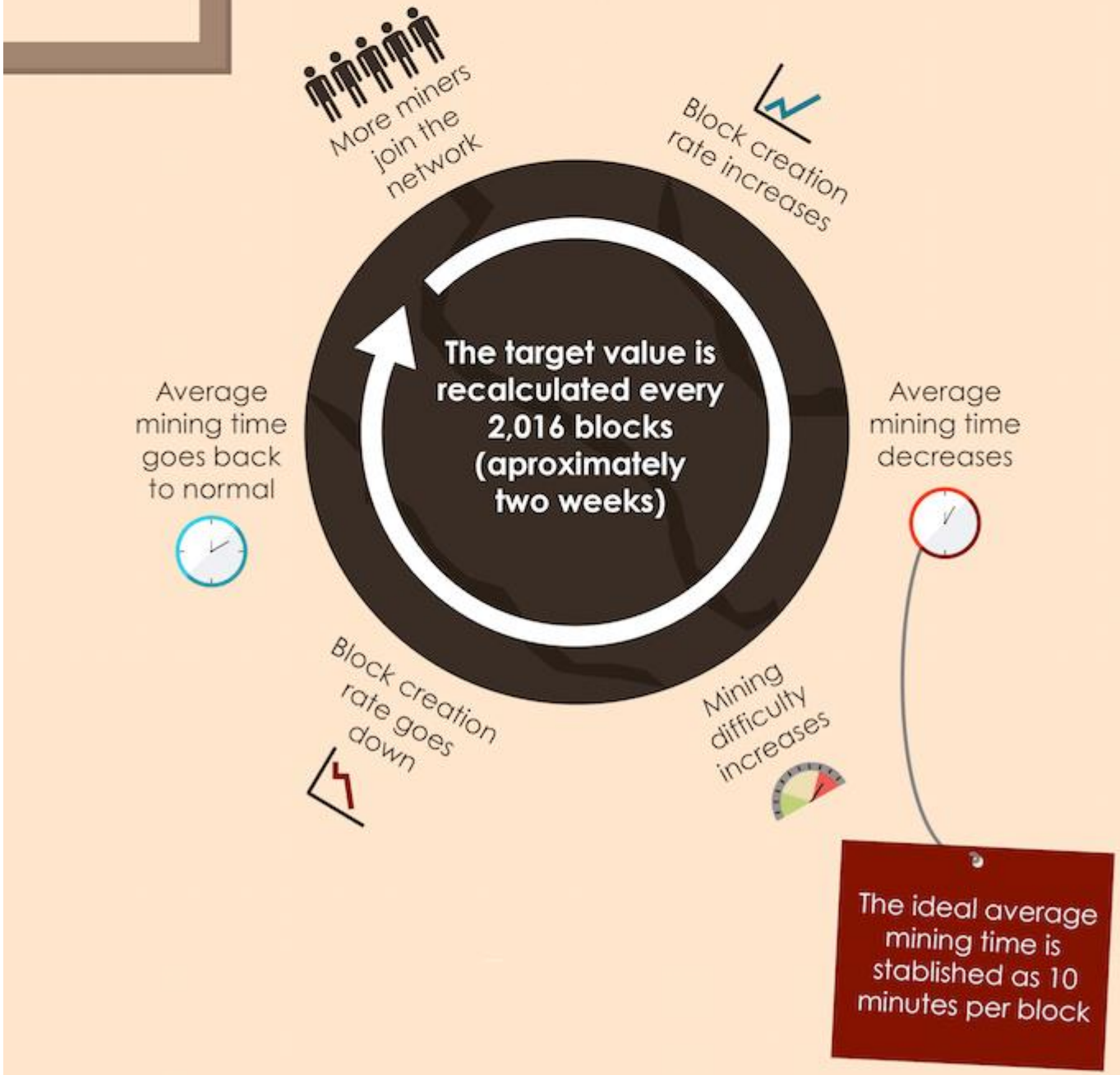
What is Proof of Work (PoW)?

It's a method to ensure that the information (the new block) was difficult (costly, time-consuming) to be made.

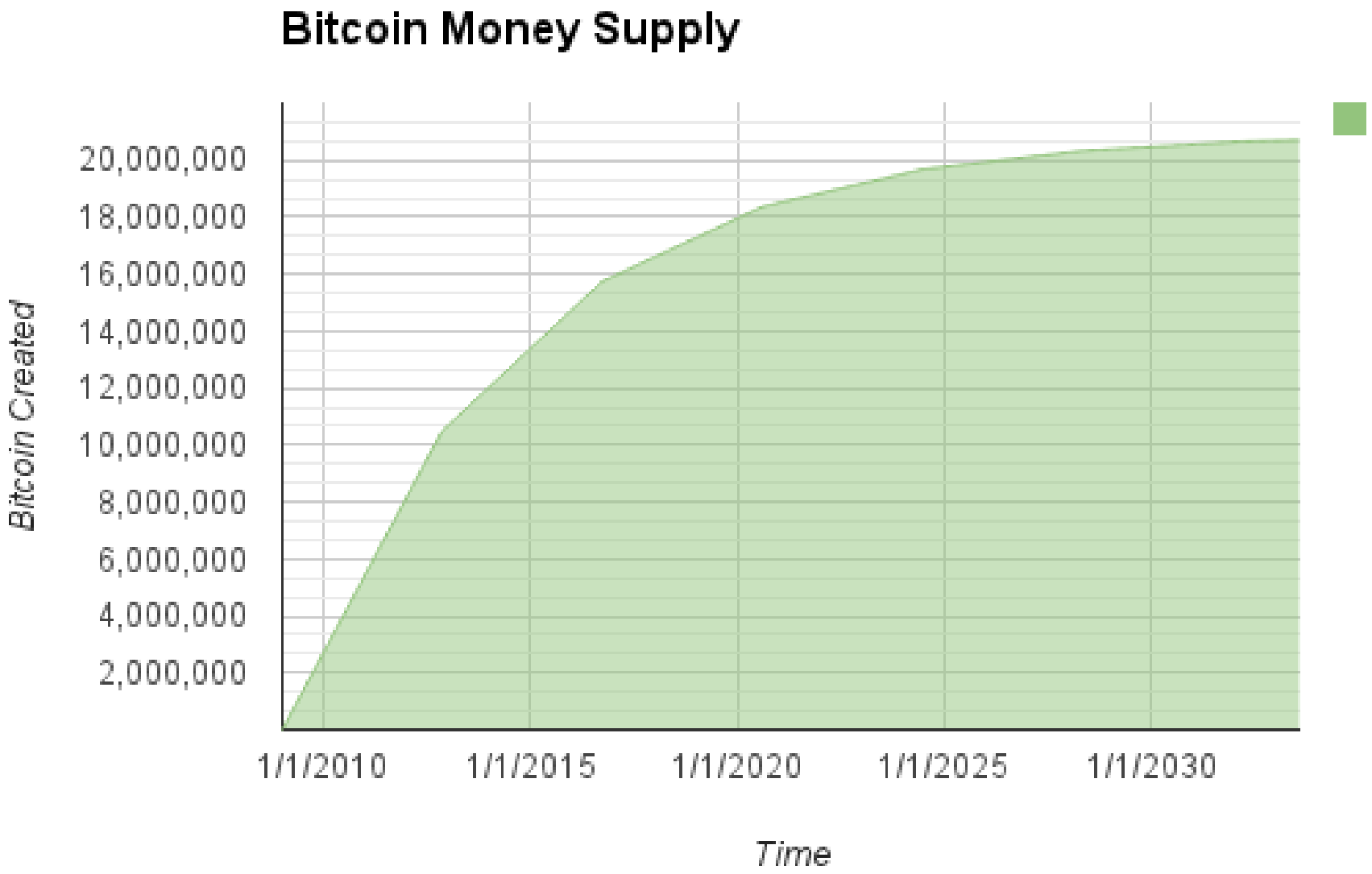


It's easy, on the other hand, for others to check if the requirements were met.

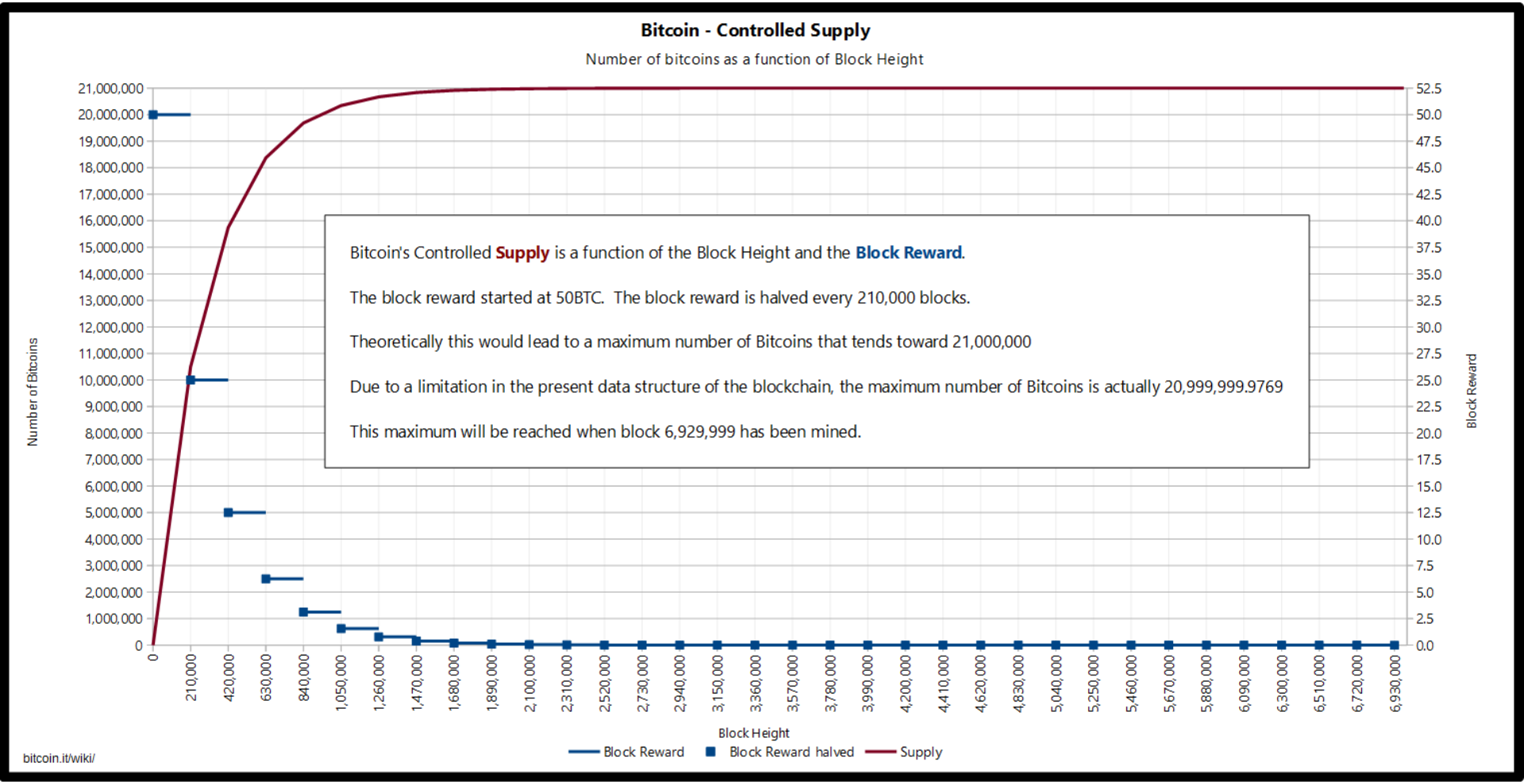

Header of the most recent block
Proposed new block
NONCE (a incremental number)
Increment and try again
Mining Difficulty
Determines
Combine and hash
Target Value
HASH number
Compare
Is HASH < Target Value?
NO
YES
MINING REWARD
You solved the PoW!


More miners join the network
Block creation rate increases
The target value is recalculated every 2,016 blocks (aproximately two weeks)
Average mining time goes back to normal
Average mining time decreases
Block creation rate goes down
Mining difficulty increases
The ideal average mining time is stablished as 10 minutes per block

Mining


Bitcoin Money Supply
20,000,000
18,000,000
16,000,000
14,000,000
12,000,000
10,000,000
8,000,000
6,000,000
4,000,000
2,000,000
Bitcoin Created
1/1/2010
1/1/2015
1/1/2020
1/1/2025
1/1/2030
Time

Mining



https://www.bitcoinblockhalf.com/

How the transaction becomes part of a block

Miners start the process of mining a new block of transactions as soon as they receive the previous block from the network.

New transactions are constantly flowing into the network, they get added to a temporary pool of unverified transactions maintained by each node.

Miners add unverified transactions (highest-fee first) from this pool to a new block, include a special transaction paying the block reward (currently 12.5 newly created bitcoin) plus the sum of transaction fees from all the transactions included in the block, and then attempt to mine the new block (candidate block).

The winning block becames part of the blockchain. The block containing Alice’s transaction is counted as one "confirmation" of that transaction.

The block that includes Alice’s transaction:

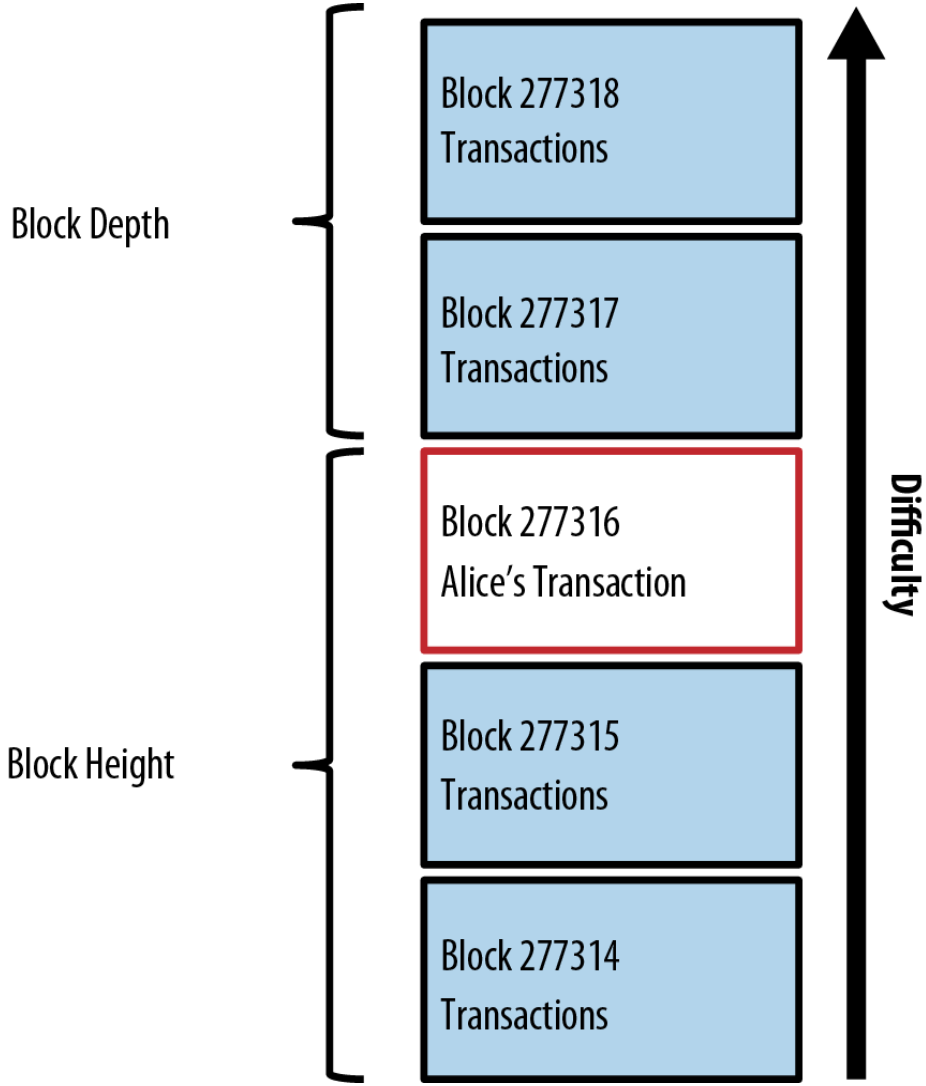
https://www.blockchain.com/btc/block/277316

How the transaction becomes part of a block

Later, a new block is mined by another miner. Because this new block is built on top of the block that contained Alice's transaction, it added even more computation to the blockchain, thereby strengthening the trust in those transactions.

Each block mined on top of the one containing the transaction counts as an additional confirmation for Alice's transaction. As the blocks pile on top of each other, it becomes exponentially harder to reverse the transaction, thereby making it more and more trusted by the network.

By convention, any block with more than six confirmations is considered irrevocable, because it would require an immense amount of computation to invalidate and recalculate six blocks.



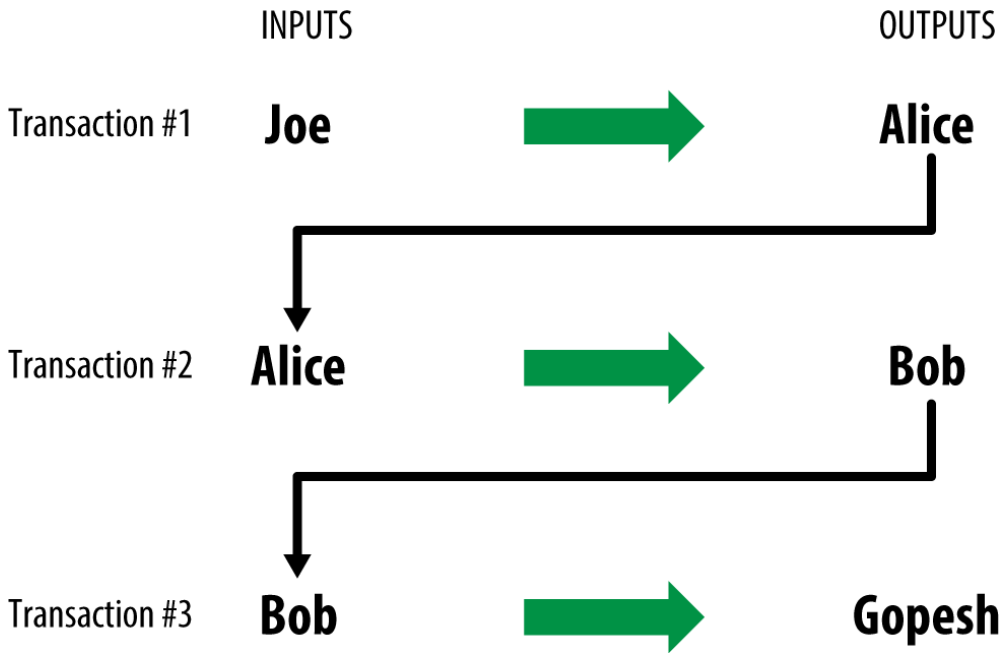
Valid and spendable transactions

As the transaction is embedded in the blockchain as part of a block, it is part of the distributed ledger of bitcoin and visible to all bitcoin applications. Each bitcoin client can independently verify the transaction as valid and spendable:

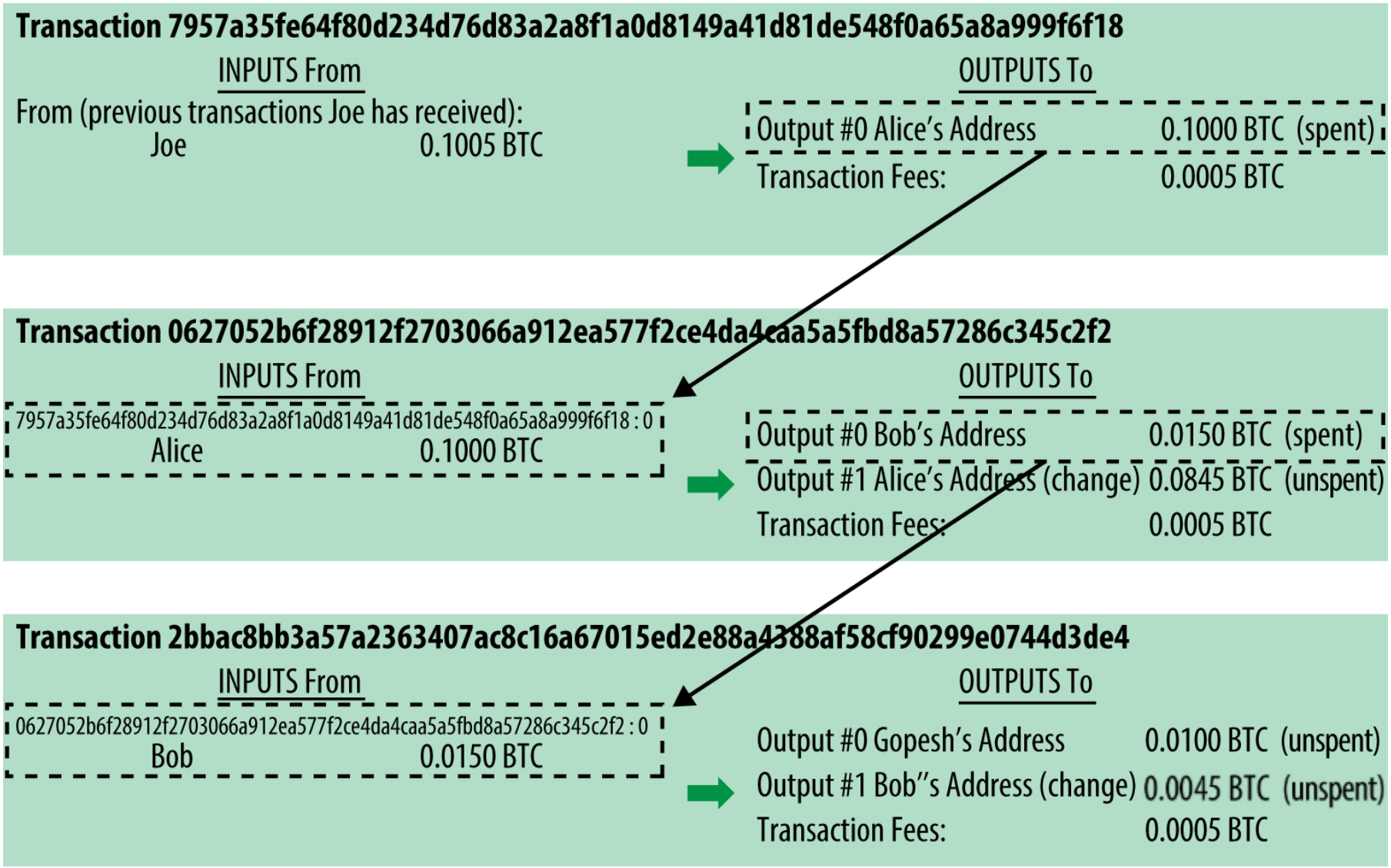
Full-node clients can track the source of the funds from the moment the bitcoin were first generated in a block, from transaction to transaction, until they reach the last transaction.

Lightweight clients can do a simplified payment verification: confirm that the transaction is in the blockchain and has several blocks mined after it.

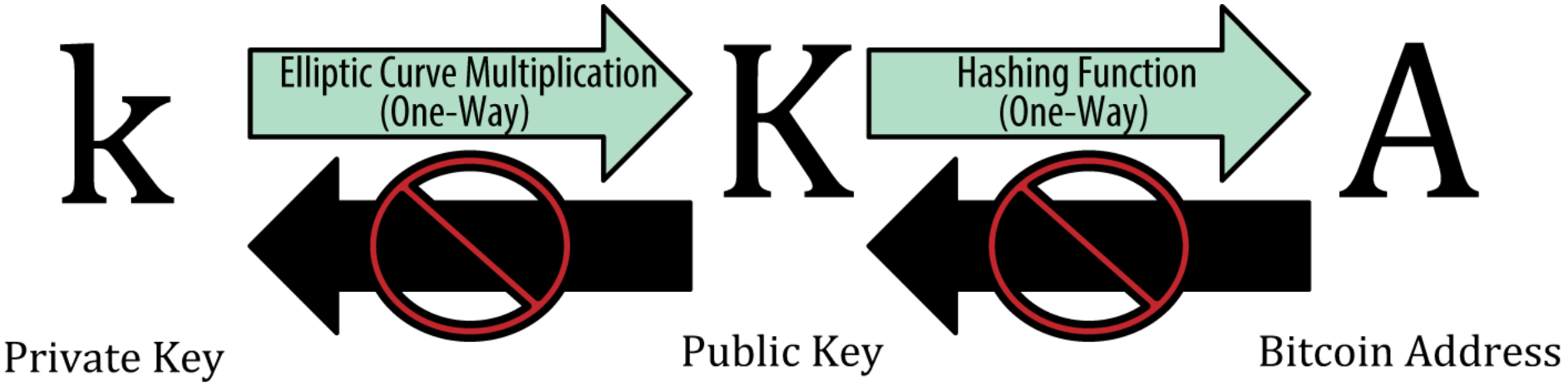
Bob can now spend the output from this and other spendable transactions:



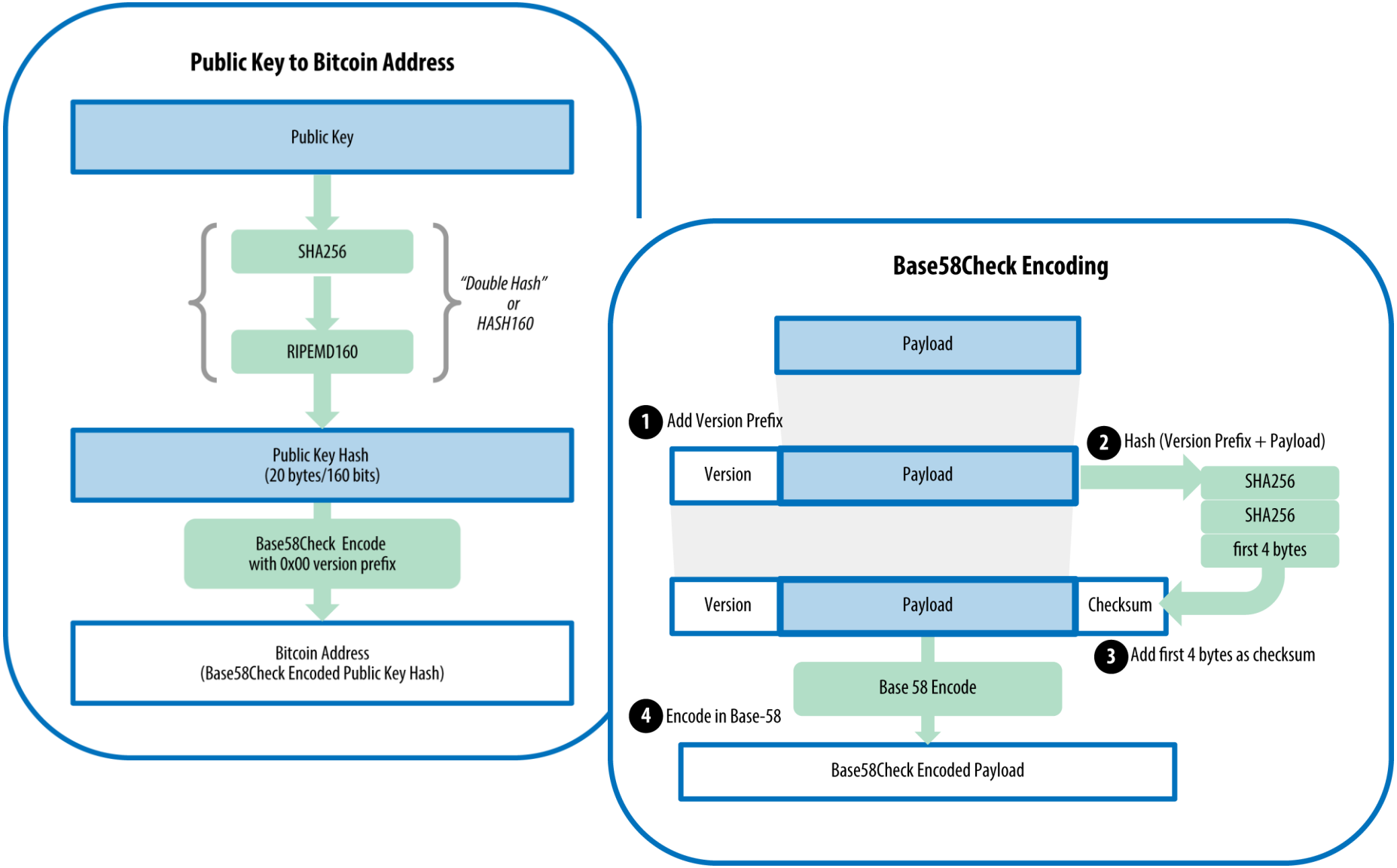
Transaction chain



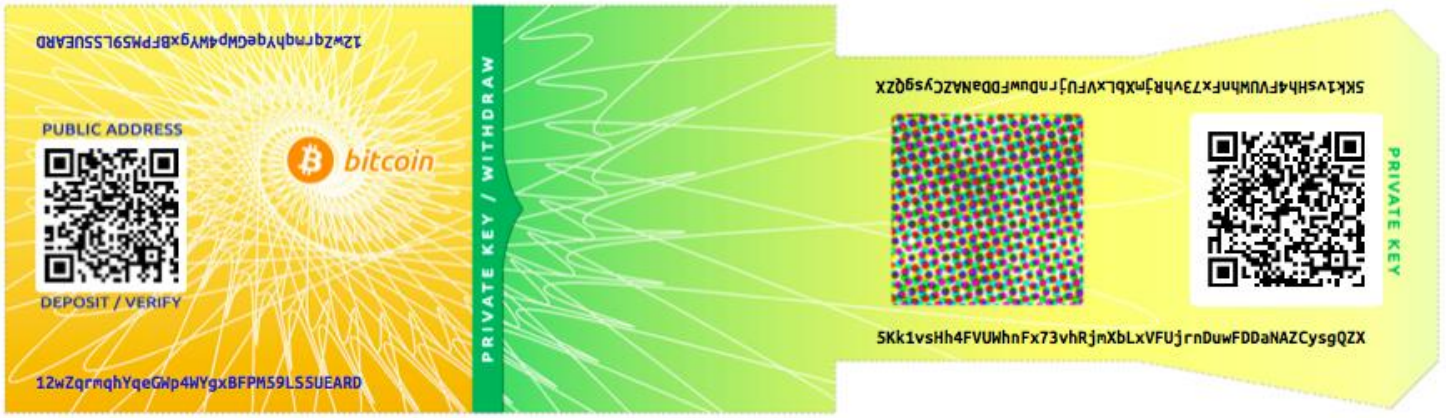
Key pair & Bitcoin address



Generate address



Private key backup ("paper wallet")



Transactions - behind the scenes

Alice's transaction:


```
{
  "version": 1,
  "locktime": 0,
  "vin": [
    {
      "txid": "7957a35fe64f80d234d76d83a2a8f1a0d8149a41d81de548f0a65a8a999f6f18",
      "vout": 0,
      "scriptSig" :
          "3045022100884d142d86652a3f47ba4746ec719bbfbd040a570b1deccbb6498c75c4ae24cb02204b9f039ff08df09cbe9f6addac960298cad530a8
          63ea8f53982c09db8f6e3813[ALL]
          0484ecc0d46f1918b30928fa0e4ed99f16a0fb4fde0735e7ade8416ab9fe423cc5412336376789d172787ec3457eee41c04f4938de5cc17b4a10fa3
          36a8d752adf",
      "sequence": 4294967295
    }
  ],
  "vout": [
    {
      "value": 0.01500000,
      "scriptPubKey": "OP_DUP OP_HASH160 ab68025513c3dbd2f7b92a94e0581f5d50f654e7 OP_EQUALVERIFY OP_CHECKSIG"
    },
    {
      "value": 0.08450000,
      "scriptPubKey": "OP_DUP OP_HASH160 7f9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a8 OP_EQUALVERIFY OP_CHECKSIG",
    }
  ]
}
```

UTXO: unspent transaction outputs

Transaction outputs are indivisible chunks of bitcoin currency, recorded on the blockchain, and recognized as valid by the entire network.

Bitcoin full nodes track all available and spendable outputs, known as unspent transaction outputs, or UTXO.

The UTXO set grows as new UTXO is created and shrinks when UTXO is consumed. Every transaction represents a change (state transition) in the UTXO set.

When we say that a user’s wallet has "received" bitcoin, what we mean is that the wallet has detected an UTXO that can be spent with one of the keys controlled by that wallet.

A user’s bitcoin "balance" is the sum of all UTXO that user’s wallet can spend.

Transaction outputs

Transaction outputs consist of two parts:

- An amount of bitcoin, denominated in satoshis, the smallest bitcoin unit
- A cryptographic puzzle that determines the conditions required to spend the output

"vout": [
  {
    "value": 0.01500000,
    "scriptPubKey": "OP_DUP OP_HASH160 ab68025513c3dbd2f7b92a94e0581f5d50f654e7
      OP_EQUALVERIFY OP_CHECKSIG"
  },
  {
    "value": 0.08450000,
    "scriptPubKey": "OP_DUP OP_HASH160 7f9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a8
      OP_EQUALVERIFY OP_CHECKSIG",
  }
]

Transaction inputs

Transaction inputs identify which UTXO will be consumed and provide proof of ownership.


```
"vin": [
  {
    "txid": "7957a35fe64f80d234d76d83a2a8f1a0d8149a41d81de548f0a65a8a999f6f18",
    "vout": 0,
    "scriptSig" :
      "3045022100884d142d86652a3f47ba4746ec719bbfbd040a570b1deccbb6498c75c4ae24cb02204b9f0
      39ff08df09cbe9f6addac960298cad530a863ea8f53982c09db8f6e3813[ALL]
      0484ecc0d46f1918b30928fa0e4ed99f16a0fb4fde0735e7ade8416ab9fe423cc5412336376789d172787
      ec3457eee41c04f4938de5cc17b4a10fa336a8d752adf",
    "sequence": 4294967295
  }
]
```


- transaction ID, referencing the transaction that contains the UTXO being spent
- output index, identifying which UTXO from that transaction is used (first one is zero)
- scriptSig, which satisfies the conditions placed on the UTXO, unlocking it for spending
- sequence number

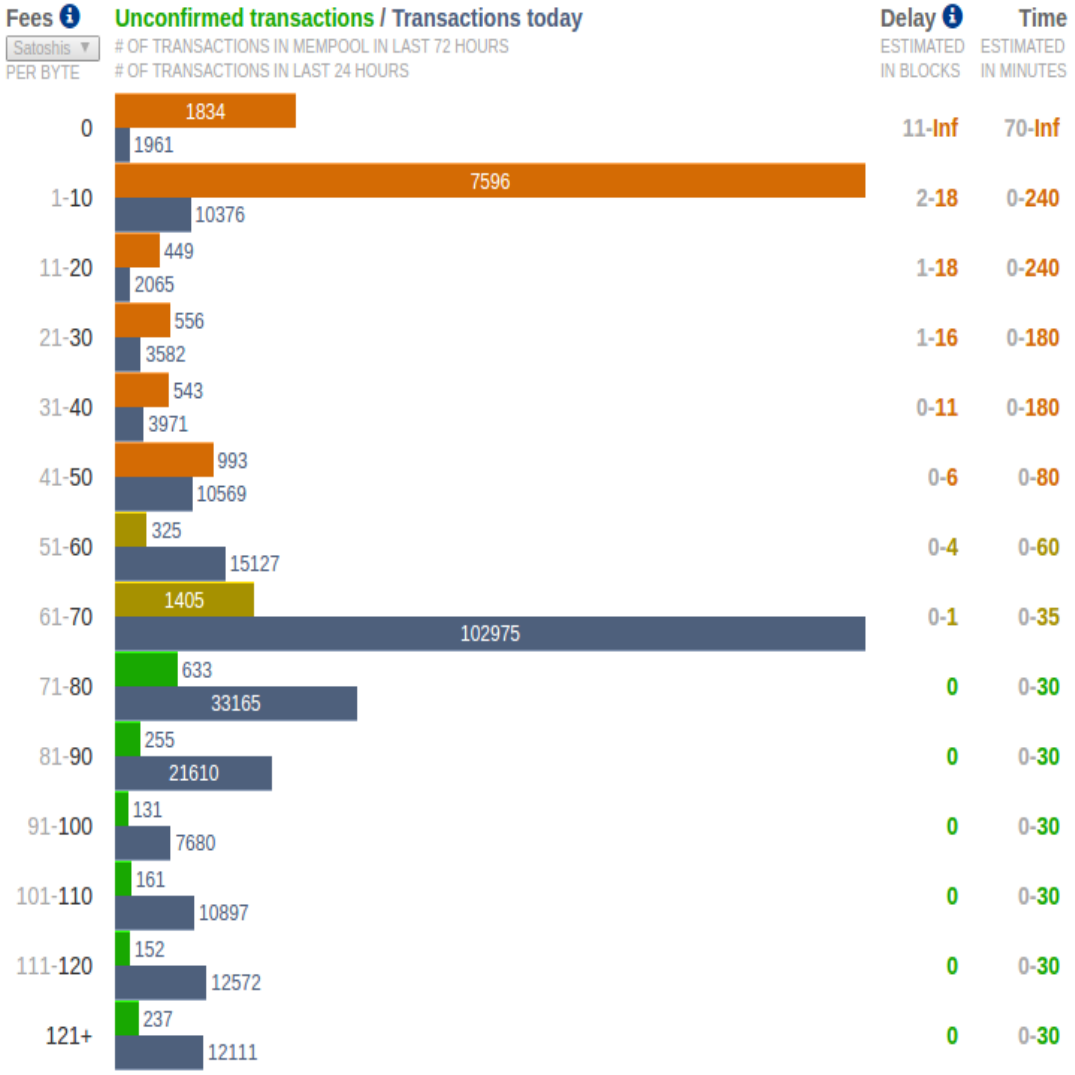
Fees ("keep the change")

Transaction fees serve as an incentive to include (mine) a transaction into the next block. Transaction fees are collected by the miner who mines the block that records the transaction on the blockchain.

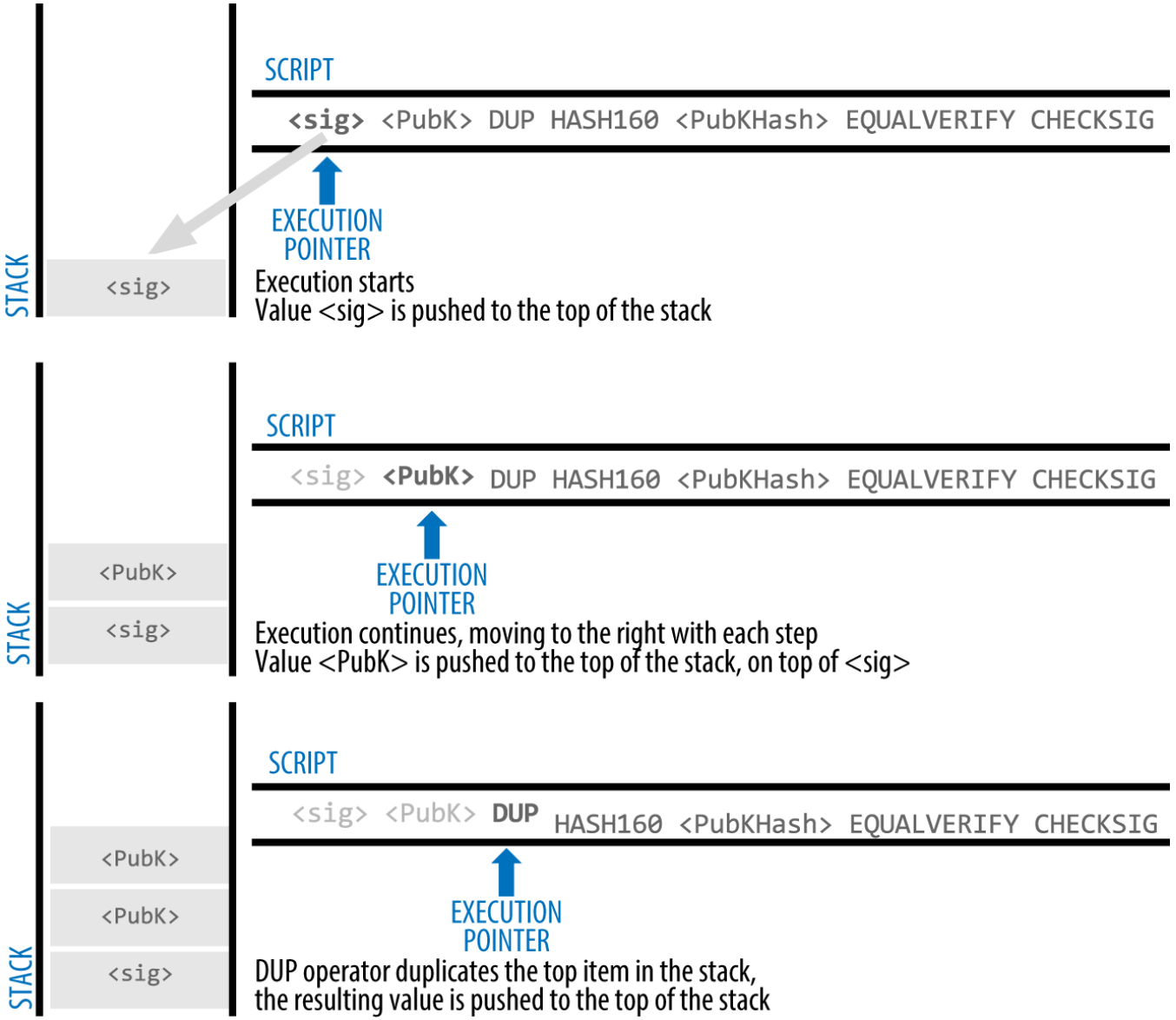
Also as a disincentive against abuse of the system by imposing a small cost on every transaction.

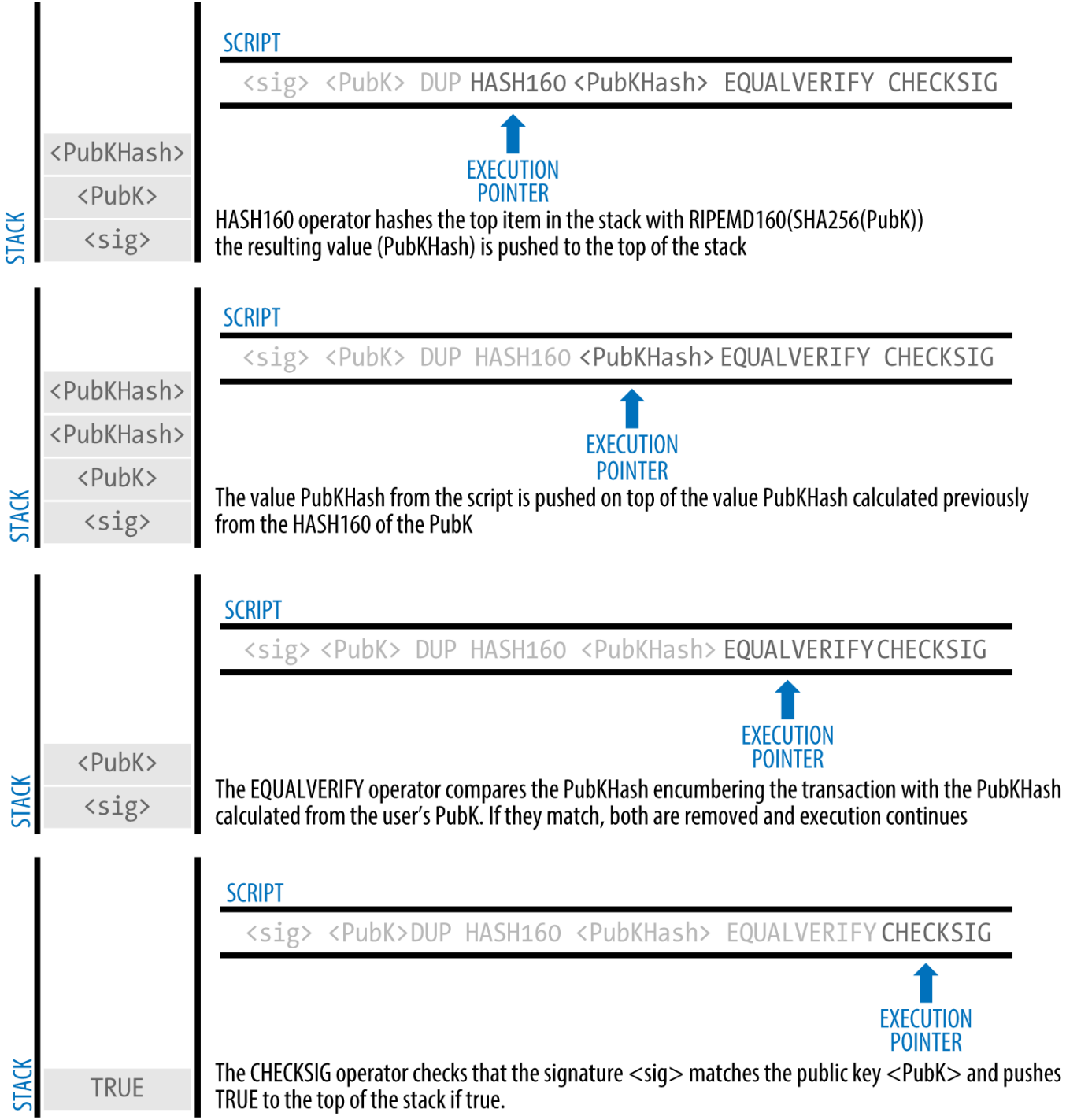
Transaction fees are calculated based on the size of the transaction in kilobytes, not the value of the transaction in bitcoin.

https://bitcoinfees.earn.com/



Script language




STACK
<PubKHash>
<PubK>
<sig>
SCRIPT
<sig> <PubK> DUP HASH160 <PubKHash> EQUALVERIFY CHECKSIG
EXECUTION POINTER
HASH160 operator hashes the top item in the stack with RIPEMD160(SHA256(PubK)) the resulting value (PubKHash) is pushed to the top of the stack
STACK
<PubKHash>
<PubKHash>
<PubK>
<sig>
SCRIPT
<sig> <PubK> DUP HASH160 <PubKHash> EQUALVERIFY CHECKSIG
EXECUTION POINTER
The value PubKHash from the script is pushed on top of the value PubKHash calculated previously from the HASH160 of the PubK
STACK
<PubK>
<sig>
SCRIPT
<sig> <PubK> DUP HASH160 <PubKHash> EQUALVERIFY CHECKSIG
EXECUTION POINTER
The EQUALVERIFY operator compares the PubKHash encumbering the transaction with the PubKHash calculated from the user's PubK. If they match, both are removed and execution continues
STACK
TRUE
SCRIPT
<sig> <PubK>DUP HASH160 <PubKHash> EQUALVERIFY CHECKSIG
EXECUTION POINTER
The CHECKSIG operator checks that the signature <sig> matches the public key <PubK> and pushes TRUE to the top of the stack if true.

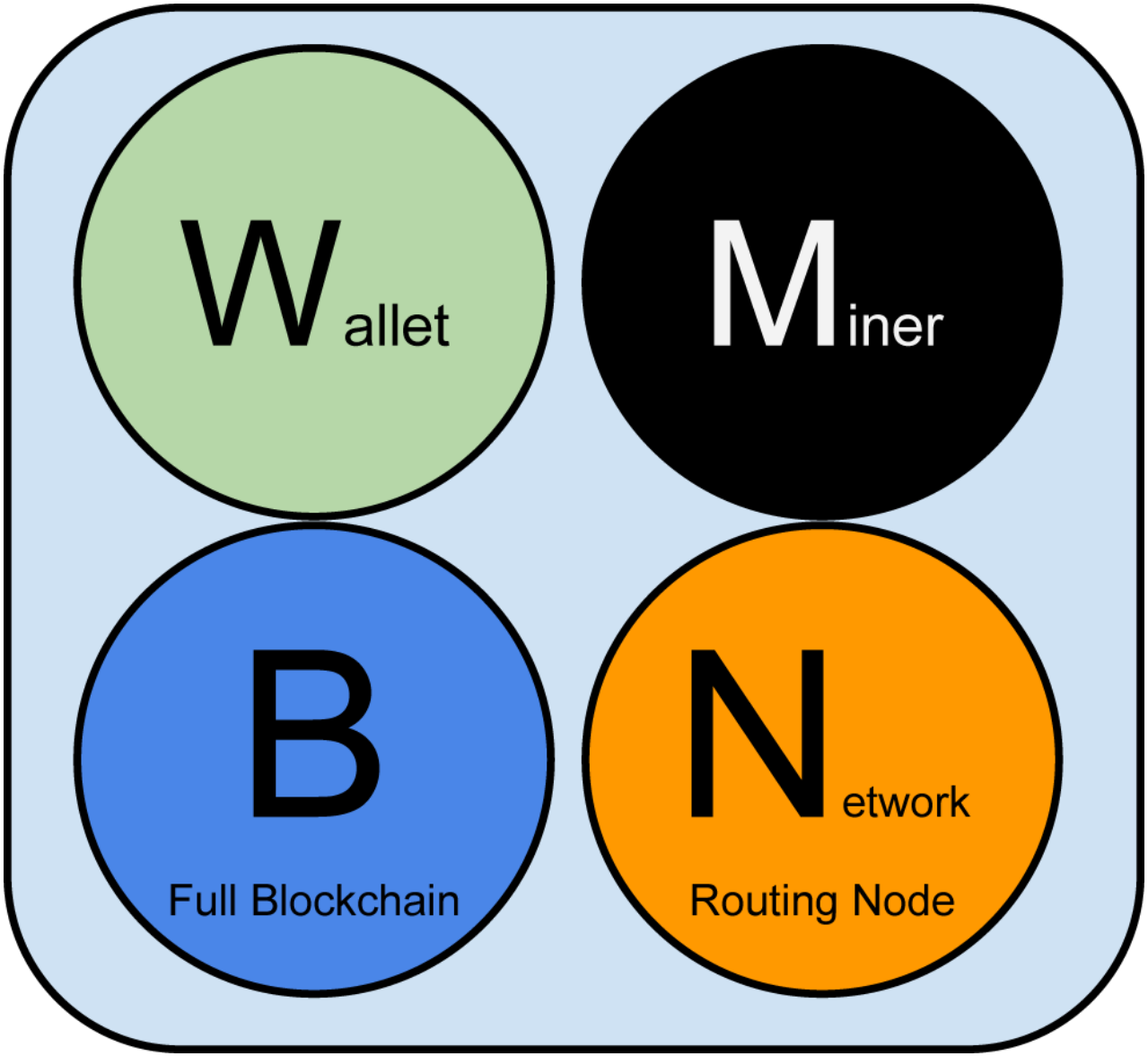
Transaction inputs

Transaction inputs identify which UTXO will be consumed and provide proof of ownership.

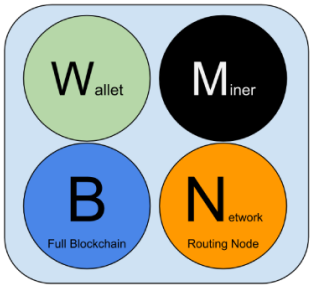

```
"vin": [
  {
    "txid": "7957a35fe64f80d234d76d83a2a8f1a0d8149a41d81de548f0a65a8a999f6f18",
    "vout": 0,
    "scriptSig" :
      "3045022100884d142d86652a3f47ba4746ec719bbfbd040a570b1deccbb6498c75c4ae24cb02204b9f0
      39ff08df09cbe9f6addac960298cad530a863ea8f53982c09db8f6e3813[ALL]
      0484ecc0d46f1918b30928fa0e4ed99f16a0fb4fde0735e7ade8416ab9fe423cc5412336376789d172787
      ec3457eee41c04f4938de5cc17b4a10fa336a8d752adf",
    "sequence": 4294967295
  }
]
```


- transaction ID, referencing the transaction that contains the UTXO being spent
- output index, identifying which UTXO from that transaction is used (first one is zero)
- scriptSig, which satisfies the conditions placed on the UTXO, unlocking it for spending
- sequence number (not used)

Bitcoin network node functions

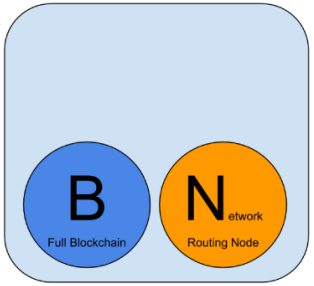


Bitcoin network node types



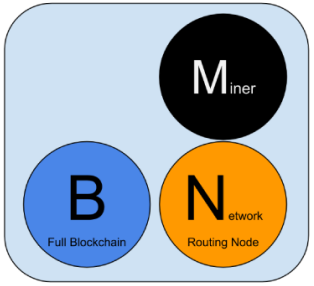
Reference Client (Bitcoin Core)

Contains a Wallet, Miner, full Blockchain database, and Network routing node on the bitcoin P2P network.



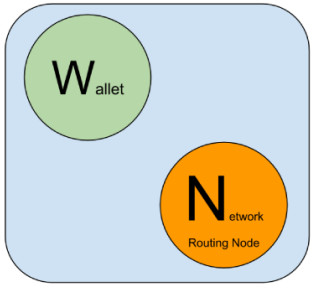
Full Block Chain Node

Contains a full Blockchain database, and Network routing node on the bitcoin P2P network.



Solo Miner

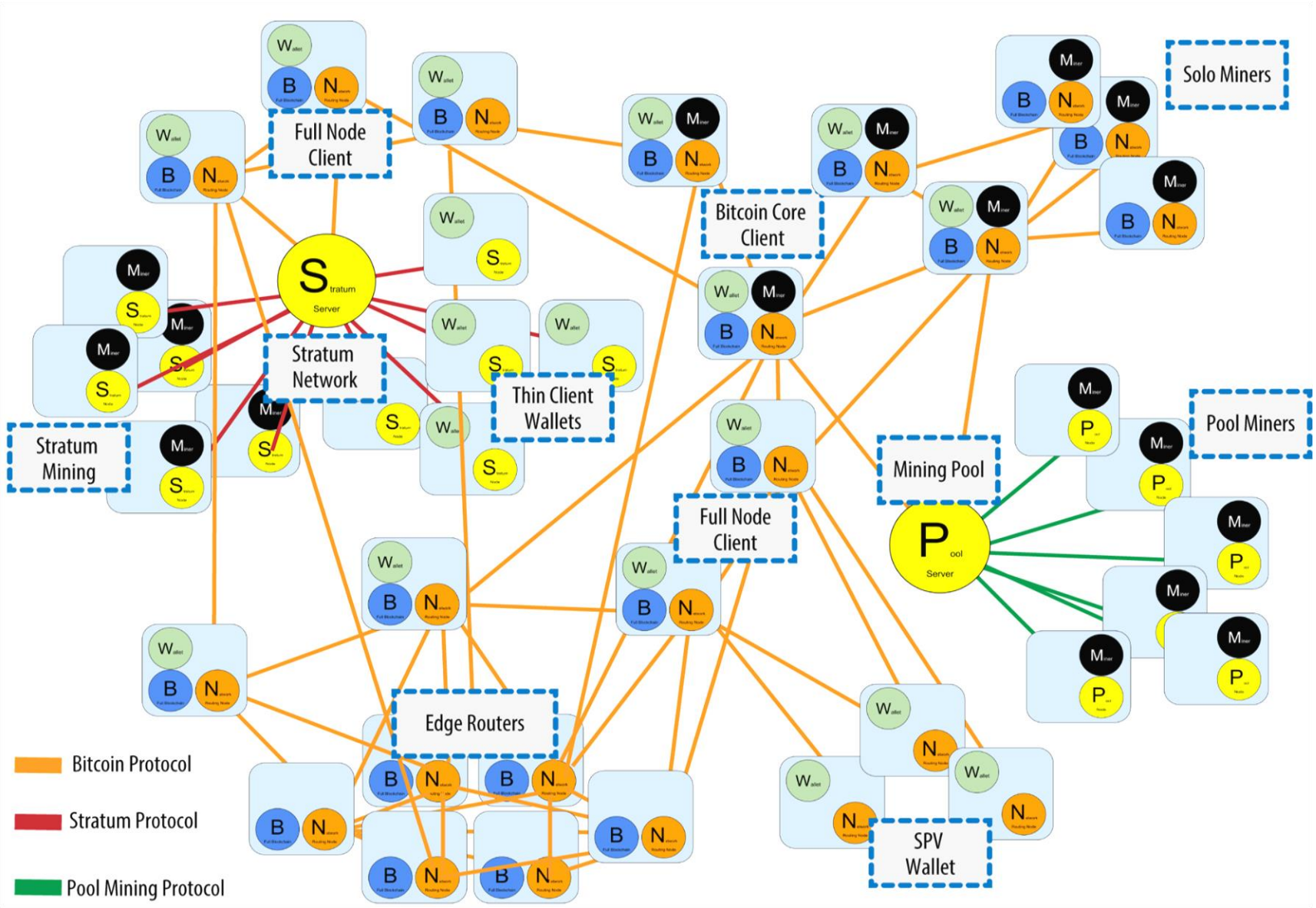
Contains a mining function with a full copy of the blockchain and a bitcoin P2P network routing node.



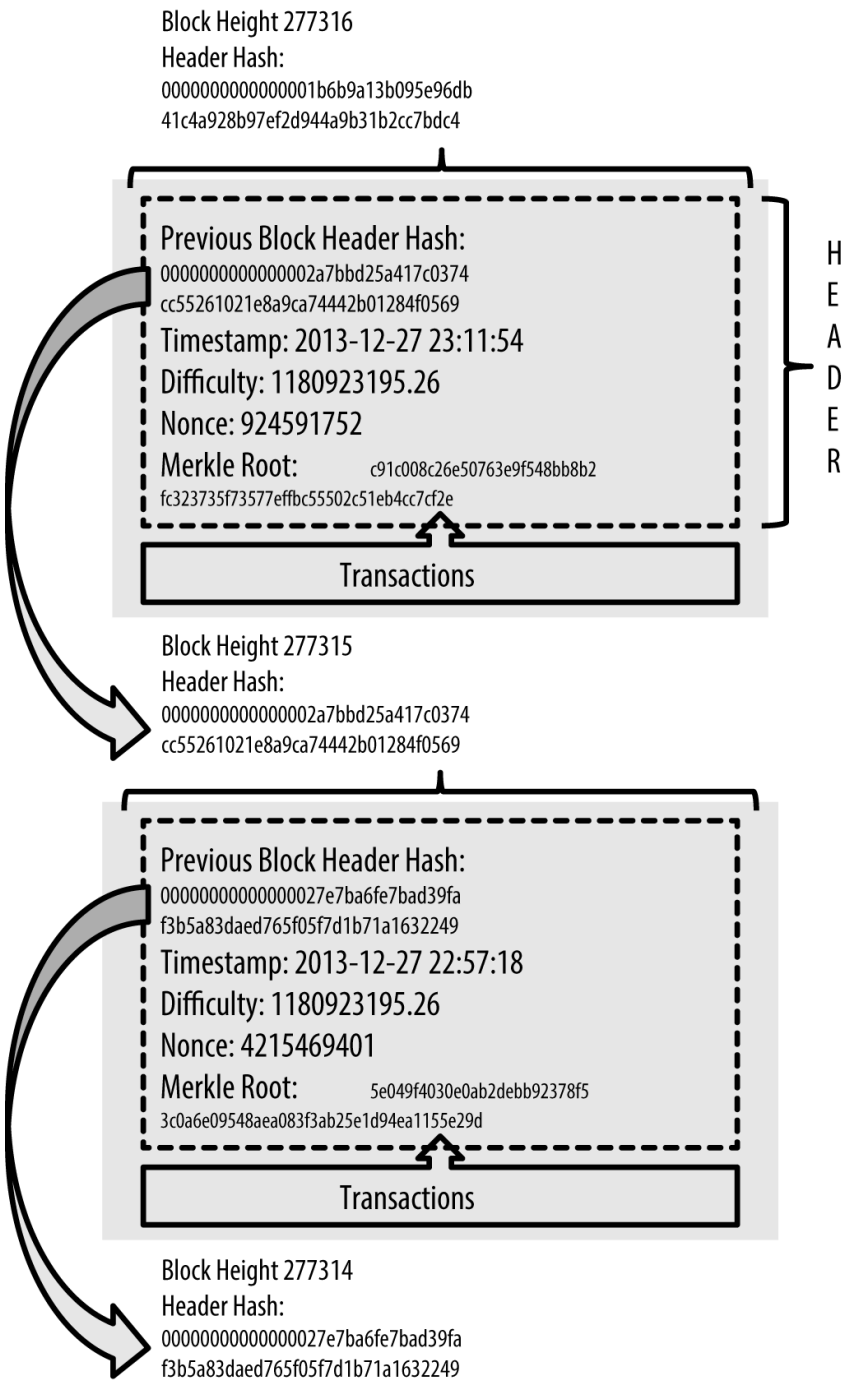
Lightweight (SPV) wallet

Contains a Wallet and a Network node on the bitcoin P2P protocol, without a blockchain.

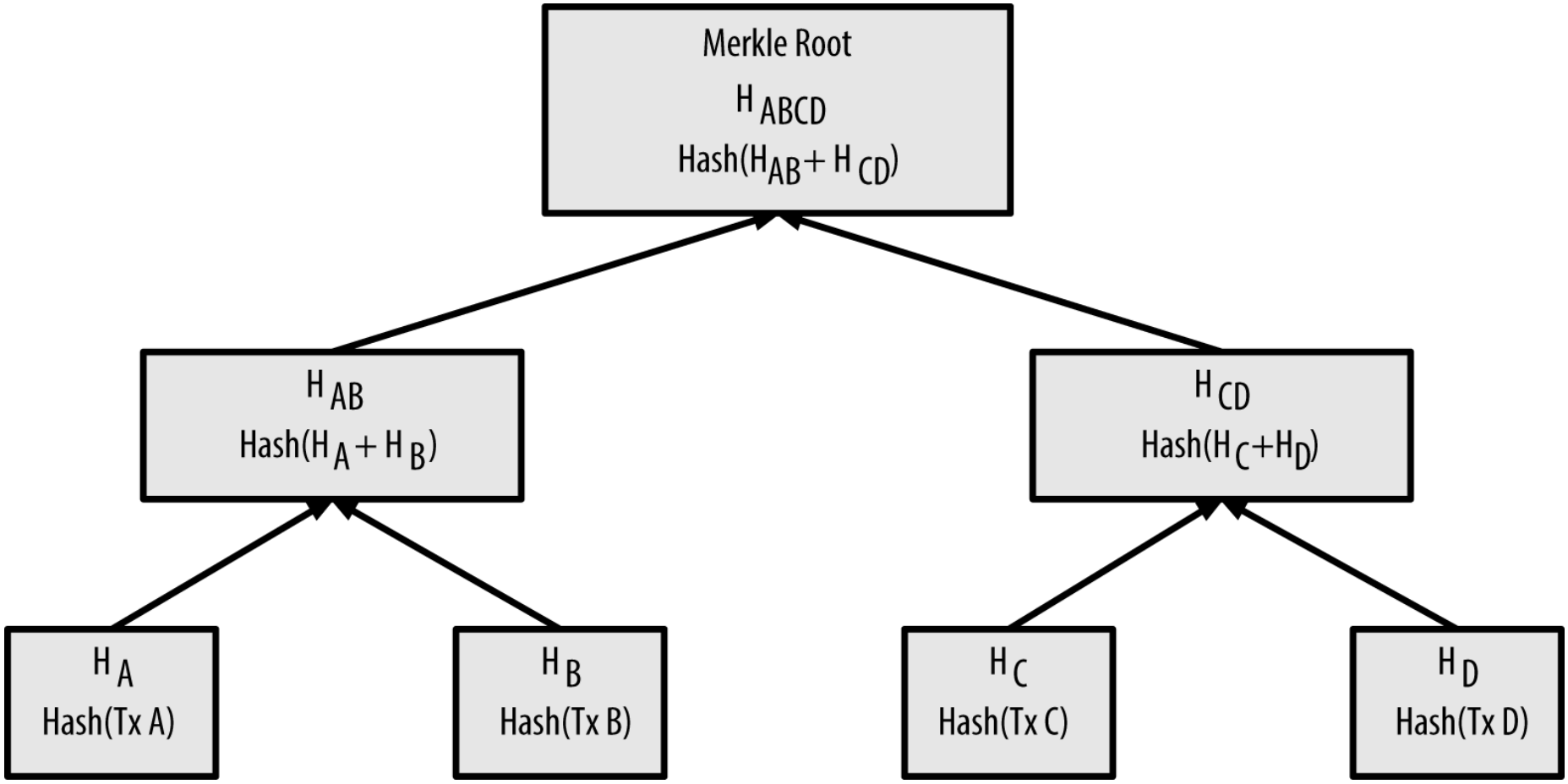
Bitcoin network



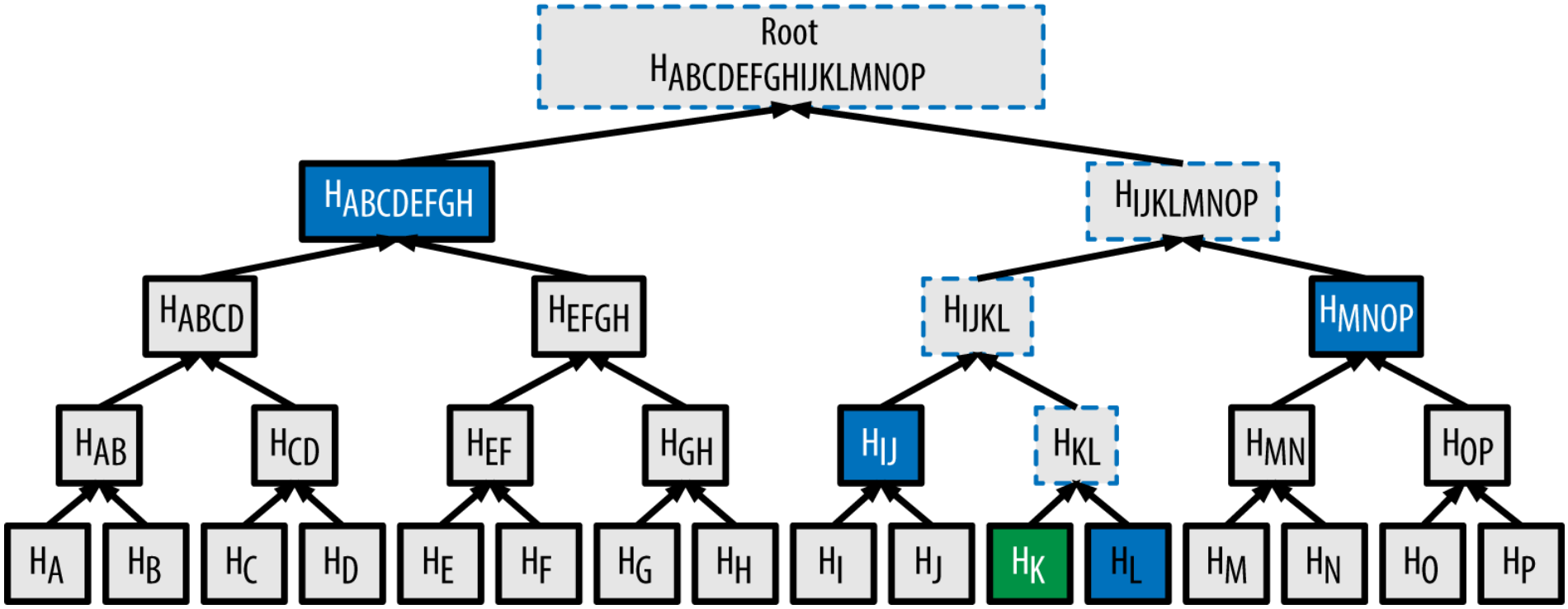
Blockchain


Block Height 277316
Header Hash:
000000000000000001b6b9a13b095e96db
41c4a928b97ef2d944a9b31b2cc7bdc4
Previous Block Header Hash:
000000000000000002a7bbd25a417c0374
cc55261021e8a9ca74442b01284f0569
Timestamp: 2013-12-27 23:11:54
Difficulty: 1180923195.26
Nonce: 924591752
Merkle Root: c91c008c26e50763e9f548bb8b2
fc323735f73577effbc55502c51eb4cc7cf2e
Transactions
HEADER
Block Height 277315
Header Hash:
000000000000000002a7bbd25a417c0374
cc55261021e8a9ca74442b01284f0569
Previous Block Header Hash:
000000000000000027e7ba6fe7bad39fa
f3b5a83daed765f05f7d1b71a1632249
Timestamp: 2013-12-27 22:57:18
Difficulty: 1180923195.26
Nonce: 4215469401
Merkle Root: 5e049f4030e0ab2debb92378f5
3c0a6e09548aea083f3ab25e1d94ea1155e29d
Transactions
Block Height 277314
Header Hash:
000000000000000027e7ba6fe7bad39fa
f3b5a83daed765f05f7d1b71a1632249

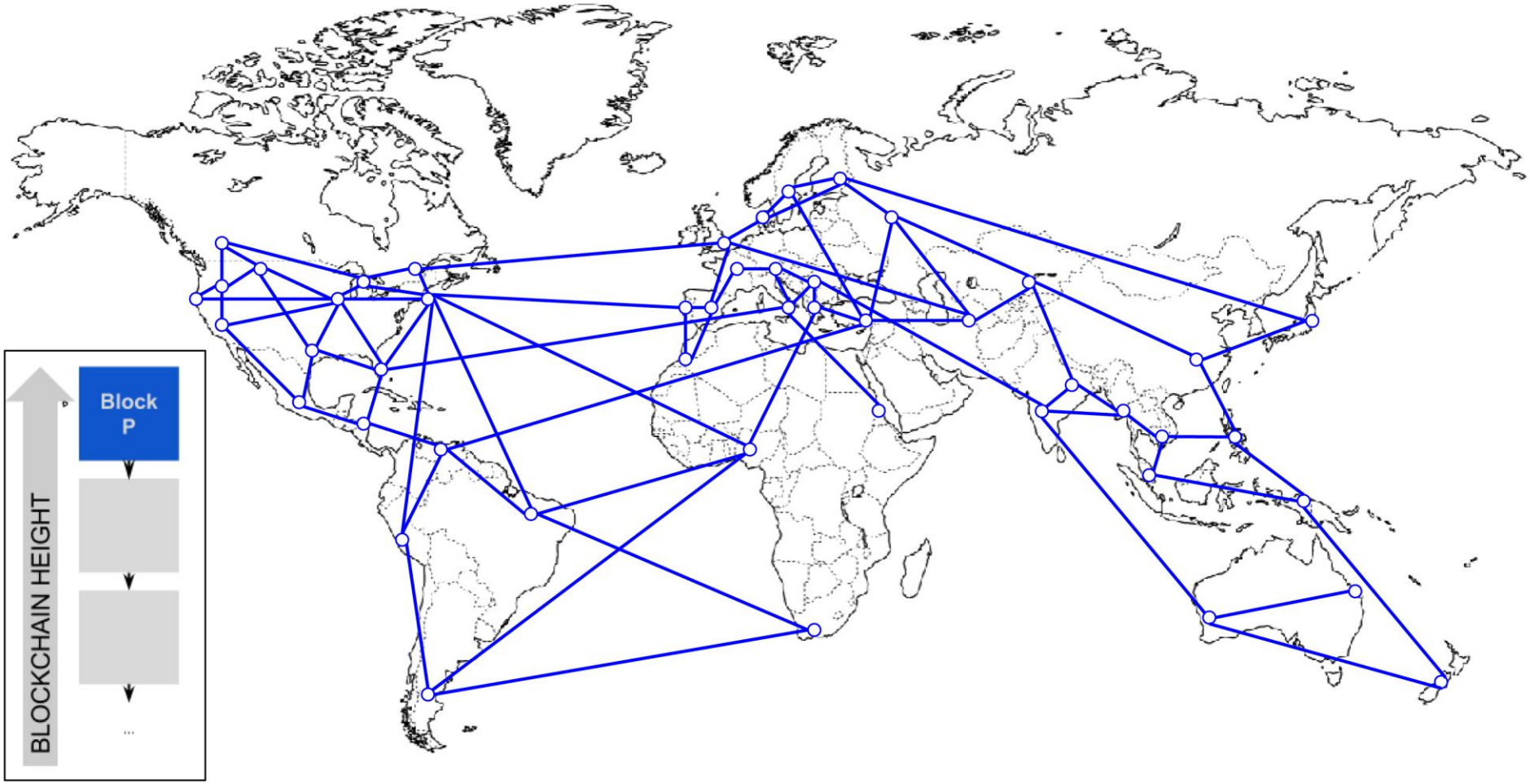
Merkle tree


Merkle Root
H ABCD
Hash(H AB + H CD)
H AB
Hash(H A + H B)
H CD
Hash(H C+H D)
H A
Hash(Tx A)
H B
Hash(Tx B)
H C
Hash(Tx C)
H D
Hash(Tx D)

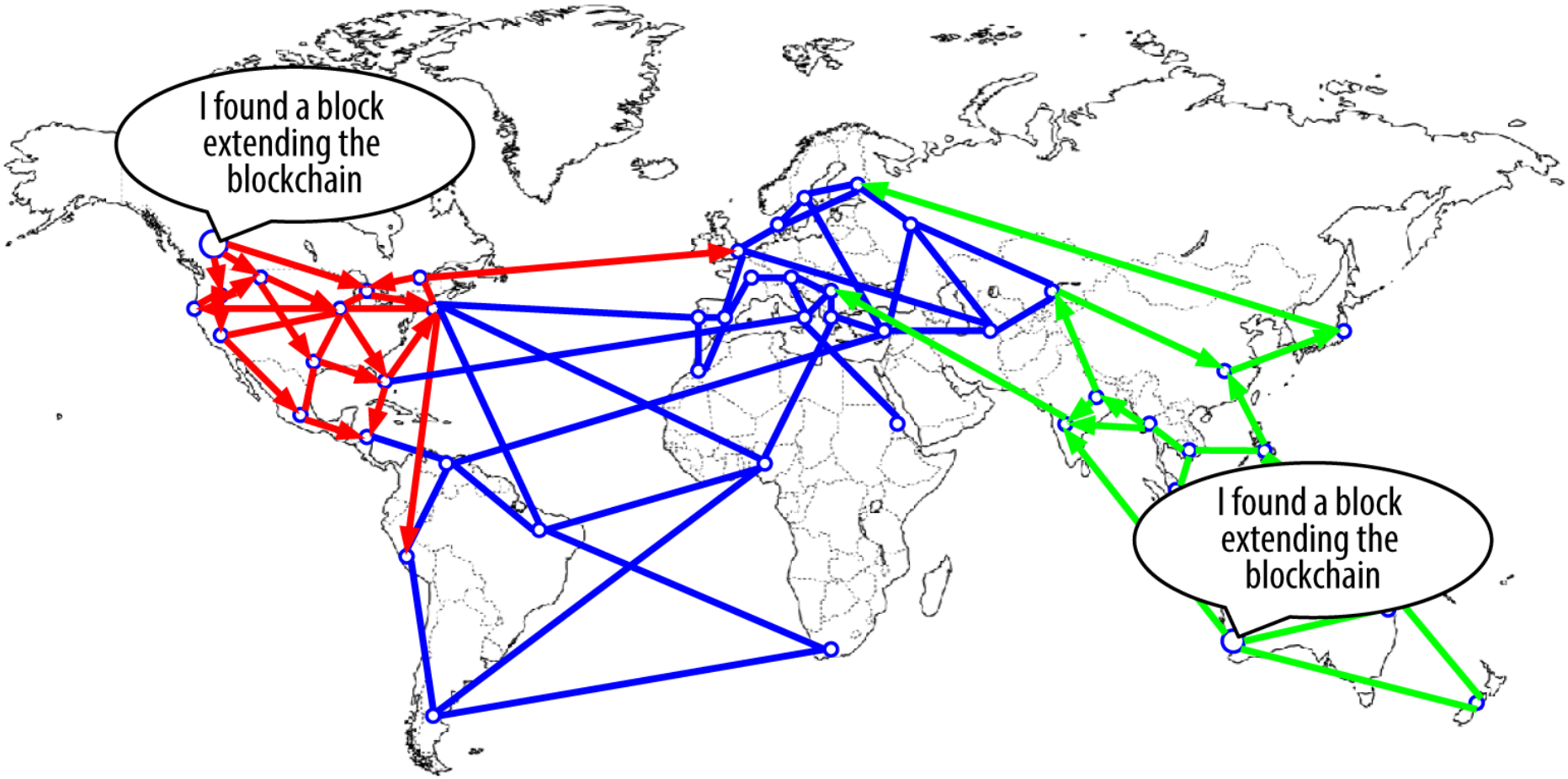
Merkle path


Root
HABCDEFGHIJKLMNOP
HABCDEFGH
HIJKLMNOP
HABCD
HEFGH
HIJKL
HMNOP
HAB
HCD
HEF
HGH
HIJ
HKL
HMN
HOP
HA
HB
HC
HD
HE
HF
HG
HH
HI
HJ
HK
HL
HM
HN
HO
HP

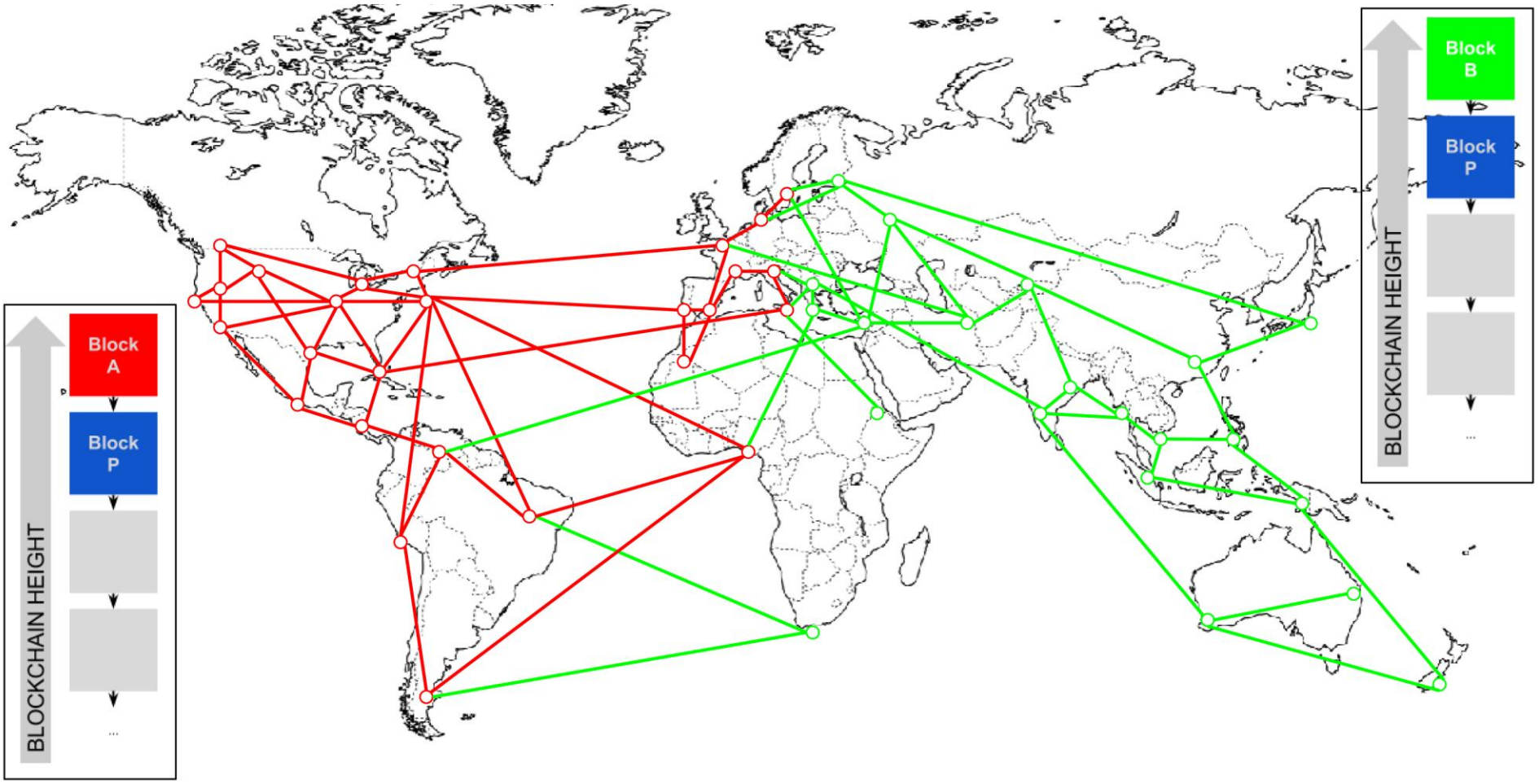
Consensus



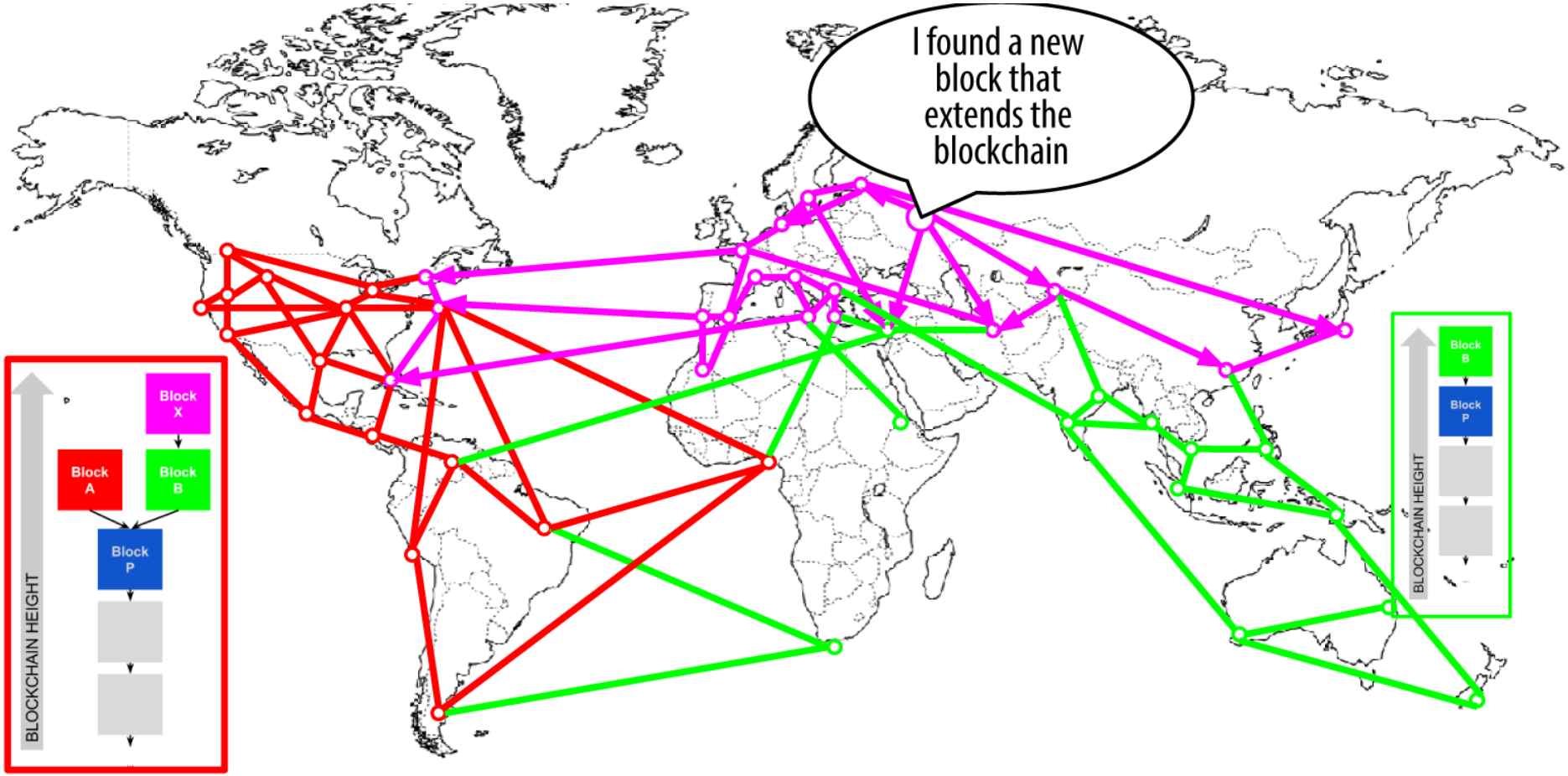
Consensus


I found a block extending the blockchain
I found a block extending the blockchain

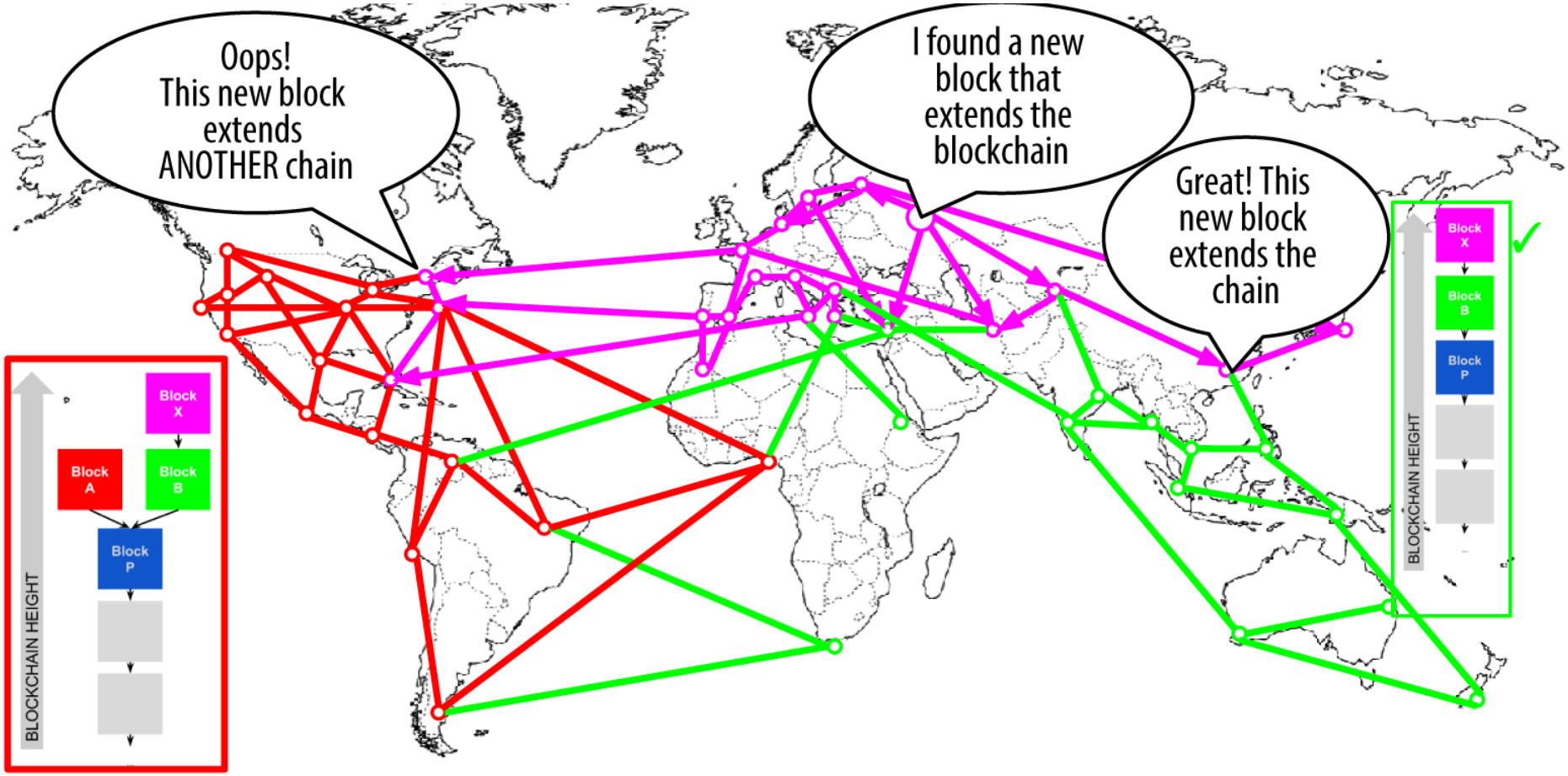
Consensus


Block
A
Block
P
...
BLOCKCHAIN HEIGHT
Block
B
Block
P
...
BLOCKCHAIN HEIGHT

Consensus


I found a new block that extends the blockchain
Block X
Block A
Block B
Block P
BLOCKCHAIN HEIGHT
Block B
Block P
BLOCKCHAIN HEIGHT

Consensus


Oops!
This new block
extends
ANOTHER chain
I found a new
block that
extends the
blockchain
Great! This
new block
extends the
chain
Block
X
Block
A
Block
B
Block
P
BLOCKCHAIN HEIGHT
Block
X
Block
B
Block
P
BLOCKCHAIN HEIGHT

Peer-to-peer network protocol

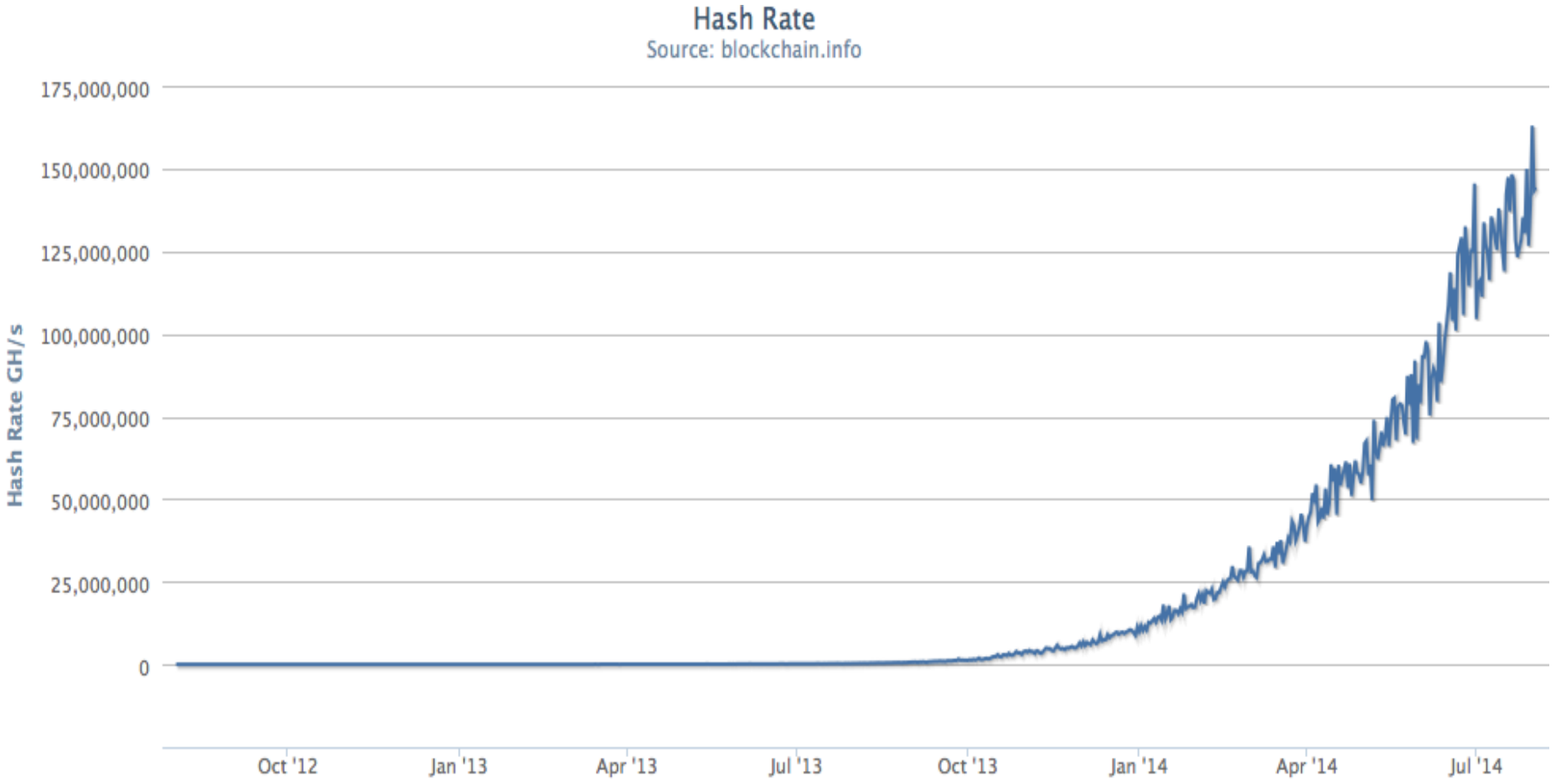
A peer-to-peer network is designed around the notion of equal peer nodes simultaneously functioning as both "clients" and "servers" to the other nodes on the network.

Peer-to-peer networks generally implement some form of virtual overlay network on top of the physical network topology, where the nodes in the overlay form a subset of the nodes in the physical network. Data is still exchanged directly over the underlying TCP/IP network, but at the application layer peers are able to communicate with each other directly, via the logical overlay links.

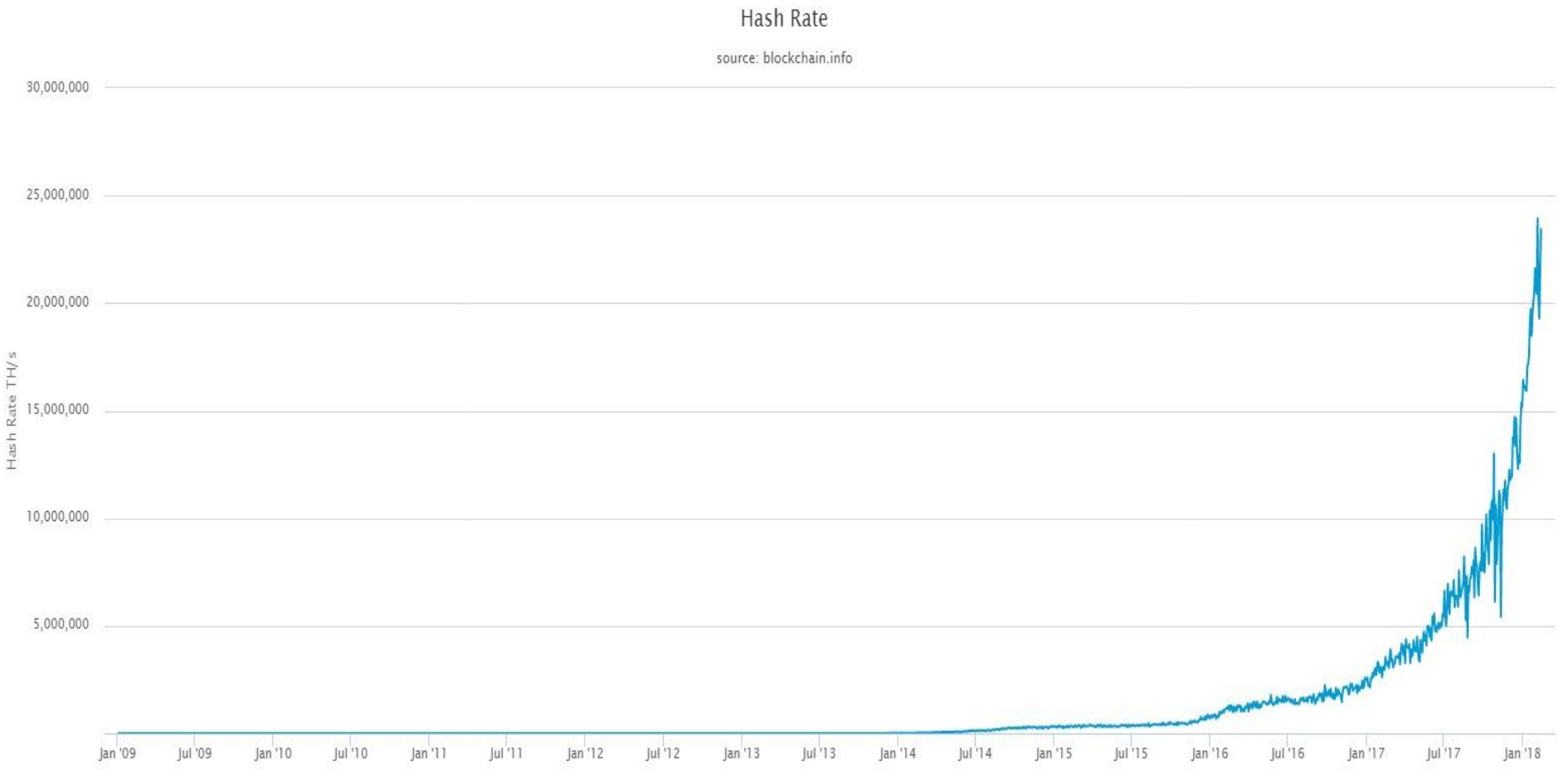
Bitcoin P2P network: https://bitcoin.org/en/p2p-network-guide

- Peer Discovery
- Connecting To Peers
- Initial Block Download
- Block Broadcasting
- Transaction Broadcasting
- Misbehaving Nodes
- Alerts

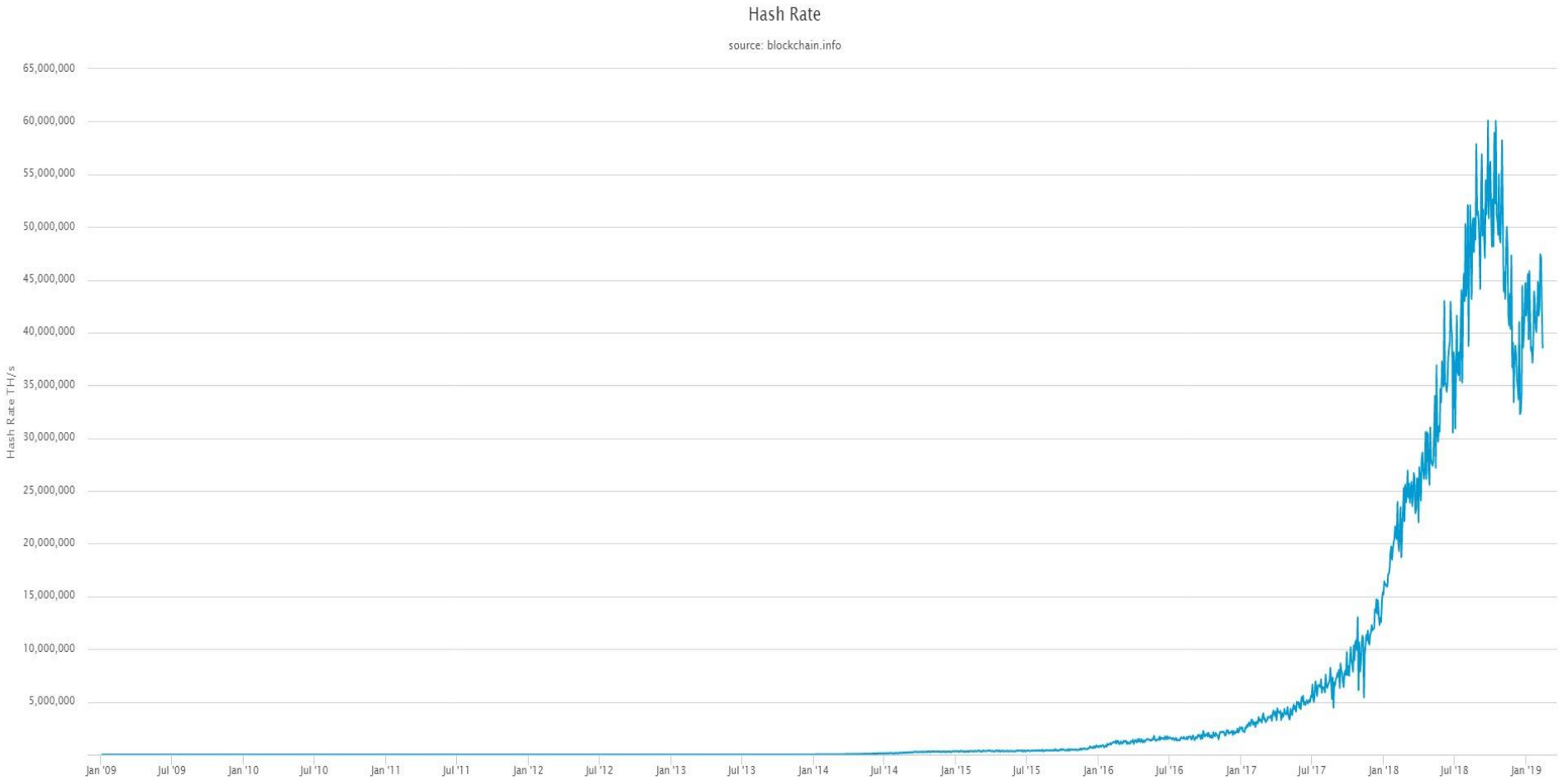
Miners' competition



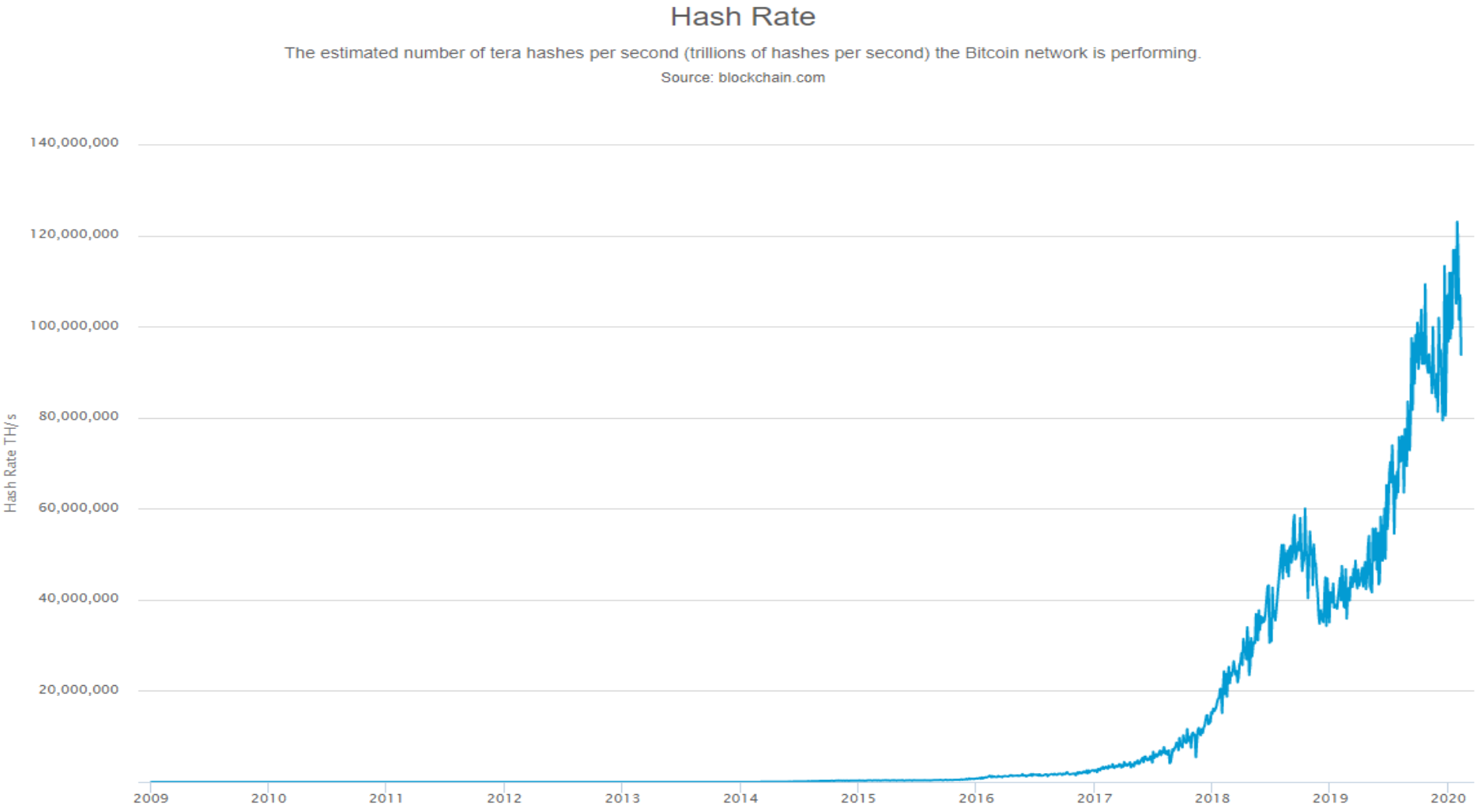
Miners' competition



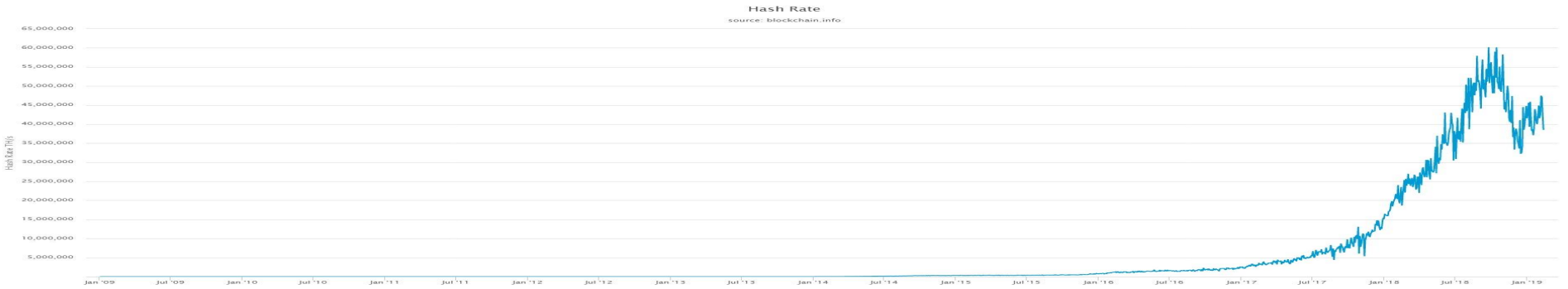
Miners' competition

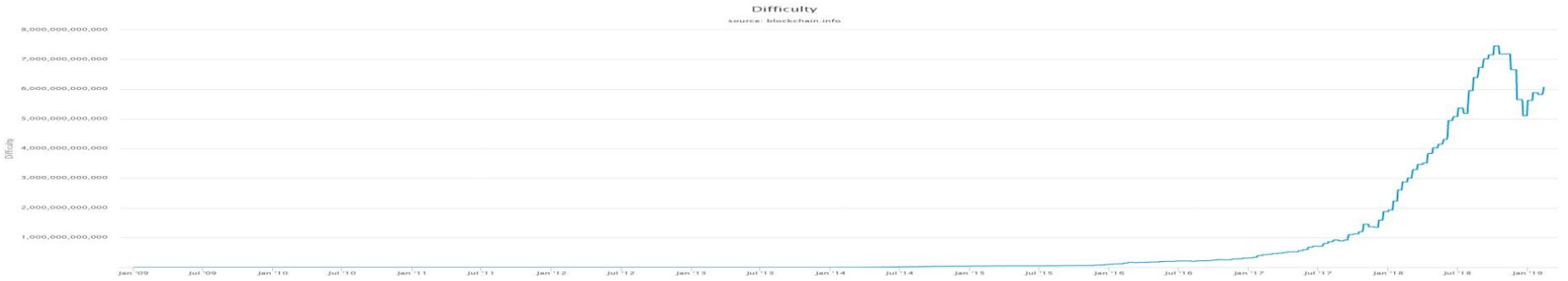


Miners' competition



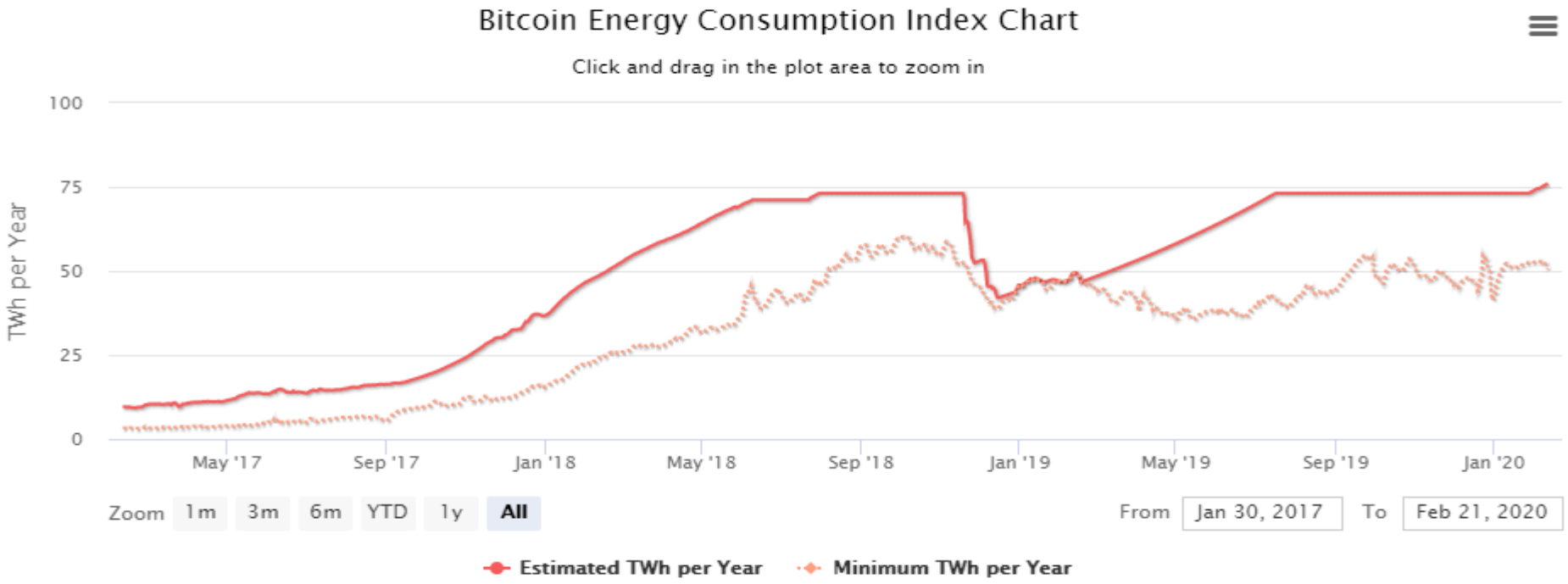
Hash rate vs. Price





Miners' daily revenue

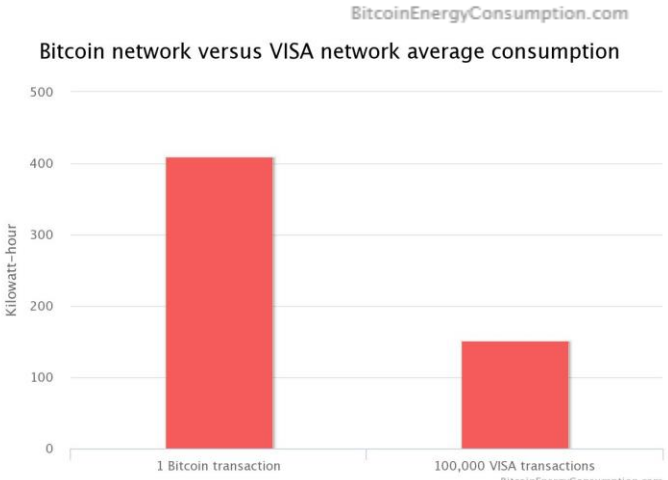
Energy Consumption



Hungary (2018): 45 TWh

as a percentage of the world's electricity consumption: 0.34%

https://digiconomist.net/bitcoin-energy-consumption



Other problems

A Survey on Security and Privacy Issues of Bitcoin (2017)

Mining

https://www.weusecoins.com/en/mining-guide/

Ki az a Satoshi Nakamoto

https://en.wikipedia.org/wiki/Satoshi_Nakamoto

Elfogadóhelyek

Pornó, drog, fegyver:
https://darknetmarkets.org/

Magyarországon:
https://coincolors.co/bitcoin-elfogadohelyek/

Life on Bitcoin:
https://www.kickstarter.com/projects/bitcoinlife/life-on-bitcoin-a-documentary-film

Balhék

Mt. Gox eltűnése:
http://www.coindesk.com/mt-gox-bitcoin-exchange-review/
https://www.mtgox.com/
http://en.wikipedia.org/wiki/Mt._Gox

BTC-e bezárása:
https://en.wikipedia.org/wiki/BTC-e

https://thechain.media/10-cryptocurrency-scandals-you-need-to-know-about

Blokk rejtjelezők

PPKE, ITK
Csapodi Márton

Advanced Encryption Standard

NIST (National Institute of Standards and Technology) kezdeményezésére

Cél: következő évtizedek titkosító algoritmusa a US kormányzat számára

1997. szeptember 12.: Request for candidate algorithms

Minimum feltételek: nyilvános, licensz mentes, szimmetrikus kulcsú, blokk kódoló, blokk méret: 128 bit, kulcs méret: 128/192/256 bit

Értékelési szempontok: biztonság, SW/HW számítási hatékonyság, memóriaszükséglet, rugalmasság (blokk és kulcsméret, számítási platform, konstrukciók: PRNG stb.), egyszerűség

Advanced Encryption Standard

1998. augusztus 20.: 15 First AES Candidates

nyilvános vizsgálat 1999. április 15-ig

1999. augusztus 9.: Round 1 Report (MARS, RC6, Rijndael, Serpent, Twofish)

nyilvános vizsgálat 2000. május 15-ig

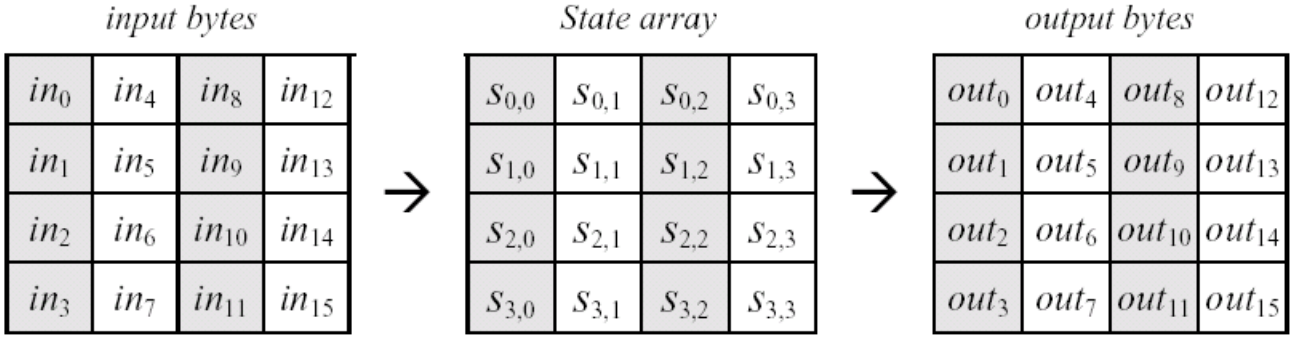
2000. október 2.: Round 2 Report (Rijndael)

FIPS-197: AES Standard, 2001. november 26. (AES-128, AES-192, AES-256)

Advanced Encryption Standard

128 bites blokk: 16 byte

4x4-es tömbbe rendezve:



in_0	in_4	in_8	in_{12}
in_1	in_5	in_9	in_{13}
in_2	in_6	in_{10}	in_{14}
in_3	in_7	in_{11}	in_{15}

→

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

→

out_0	out_4	out_8	out_{12}
out_1	out_5	out_9	out_{13}
out_2	out_6	out_{10}	out_{14}
out_3	out_7	out_{11}	out_{15}

a byte-okat polinomként kezeli:

$$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0$$

összeadás, szorzás: GF(2^8)-ben

Advanced Encryption Standard

AES-ben használt modulus:

$$m(x) = x^8 + x^4 + x^3 + x + 1$$

minden byte reprezentálható polinomként
példa (01010111)x(10000011) szorzásra:

$$\begin{aligned}(x^6 + x^4 + x^2 + x + 1)\,(x^7 + x + 1) \quad &= \quad x^{13} + x^{11} + x^9 + x^8 + x^7 + \\ &\qquad x^7 + x^5 + x^3 + x^2 + x + \\ &\qquad x^6 + x^4 + x^2 + x + 1 \\ &= \quad x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1\end{aligned}$$

$$\begin{aligned}x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1 \ \texttt{modulo}\ (x^8 + x^4 + x^3 + x + 1) \\ = \quad x^7 + x^6 + 1.\end{aligned}$$

azaz: (01010111)x(10000011)=(11000001) hatékonyan számítható

Advanced Encryption Standard

4 byte-ból word képezhető: $a(x) = a_3x^3 + a_2x^2 + a_1x + a_0$

az együtthatók GF(2^8) -beliek

összeadás: $a(x)+b(x) = (a_3 \oplus b_3)x^3 + (a_2 \oplus b_2)x^2 + (a_1 \oplus b_1)x + (a_0 \oplus b_0)$

szorzás modulo (x^4+1): $a(x) \otimes b(x)$

$$d(x) = d_3x^3 + d_2x^2 + d_1x + d_0$$

$$d_0 = (a_0 \bullet b_0) \oplus (a_3 \bullet b_1) \oplus (a_2 \bullet b_2) \oplus (a_1 \bullet b_3)$$
$$d_1 = (a_1 \bullet b_0) \oplus (a_0 \bullet b_1) \oplus (a_3 \bullet b_2) \oplus (a_2 \bullet b_3)$$
$$d_2 = (a_2 \bullet b_0) \oplus (a_1 \bullet b_1) \oplus (a_0 \bullet b_2) \oplus (a_3 \bullet b_3)$$
$$d_3 = (a_3 \bullet b_0) \oplus (a_2 \bullet b_1) \oplus (a_1 \bullet b_2) \oplus (a_0 \bullet b_3)$$

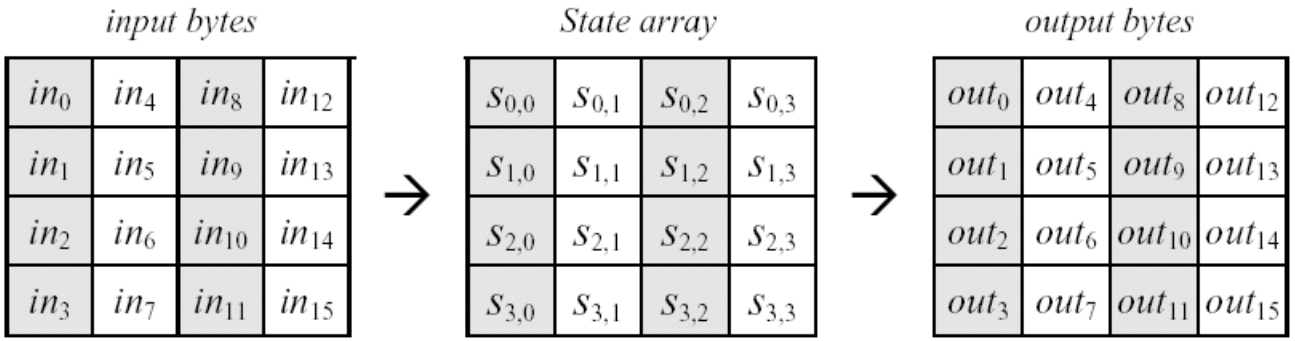
mátrix alakban:

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Advanced Encryption Standard

4x4-es tömb oszlopai: egy 32 bites word

oszlopok száma: Nb=4



kulcsok: 128/192/256 bit, vagyis 4 soros tömbbe rendezve: Nk=4, 6 vagy 8

round: helyettesítés
sor eltolás
oszlop keverés
kulcs hozzáadás

	Key Length (Nk words)	Block Size (Nb words)	Number of Rounds (Nr)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Advanced Encryption Standard

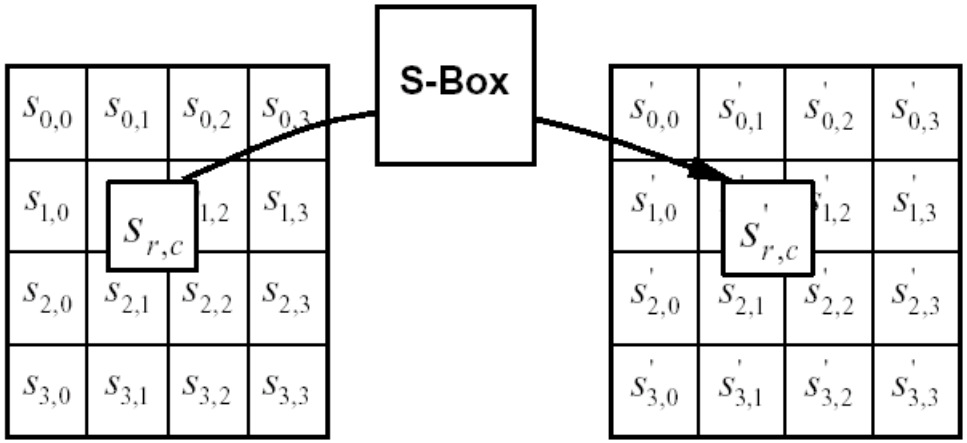
```
Cipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
   byte  state[4,Nb]

   state = in

   AddRoundKey(state, w[0, Nb-1])                  // See Sec. 5.1.4

   for round = 1 step 1 to Nr-1
      SubBytes(state)                              // See Sec. 5.1.1
      ShiftRows(state)                             // See Sec. 5.1.2
      MixColumns(state)                            // See Sec. 5.1.3
      AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
   end for

   SubBytes(state)
   ShiftRows(state)
   AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])

   out = state
end
```

```
InvCipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
   byte  state[4,Nb]

   state = in

   AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1]) // See Sec. 5.1.4

   for round = Nr-1 step -1 downto 1
      InvShiftRows(state)                          // See Sec. 5.3.1
      InvSubBytes(state)                           // See Sec. 5.3.2
      AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
      InvMixColumns(state)                         // See Sec. 5.3.3
   end for

   InvShiftRows(state)
   InvSubBytes(state)
   AddRoundKey(state, w[0, Nb-1])

   out = state
end
```

Advanced Encryption Standard

helyettesítés (SubBytes): inverz GF(2^8)-ban, majd leképezés:

$$\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$



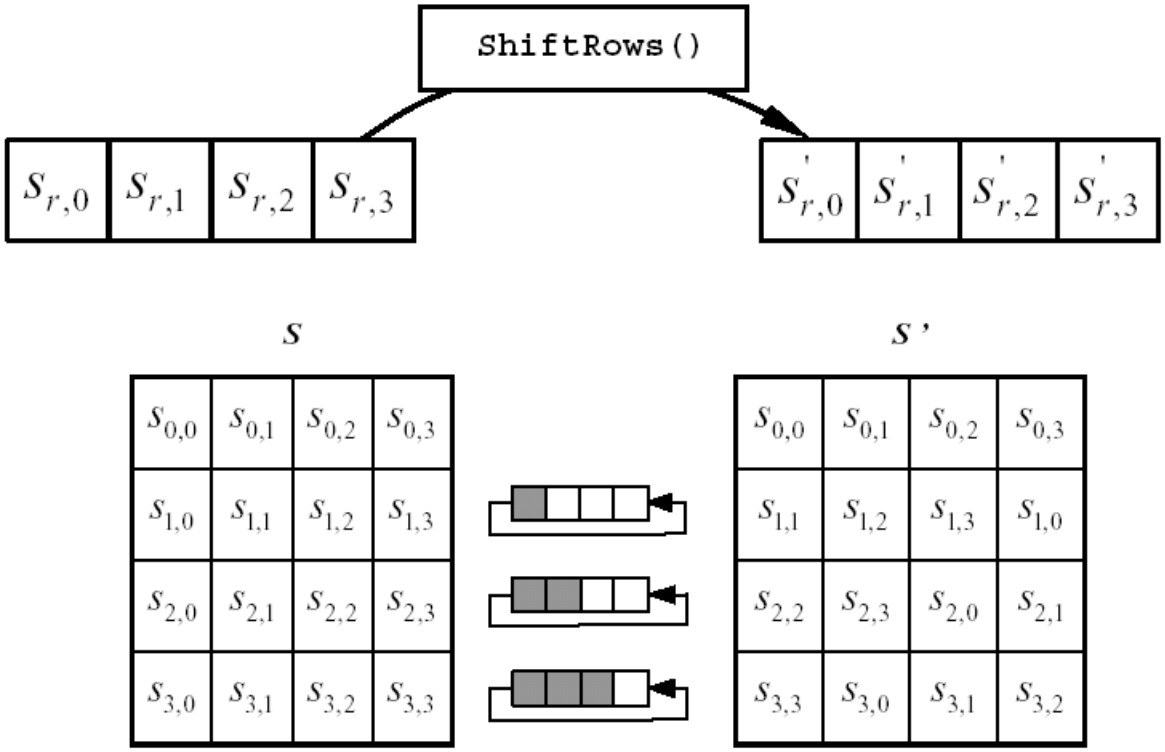
Advanced Encryption Standard

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
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									
e	e1	f8	98	11	69	d9	8e									
f	8c	a1	89	0d	bf	e6	42									

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

Advanced Encryption Standard

sor eltolás (ShiftRows):



Advanced Encryption Standard

oszlop keverés (MixColumns):

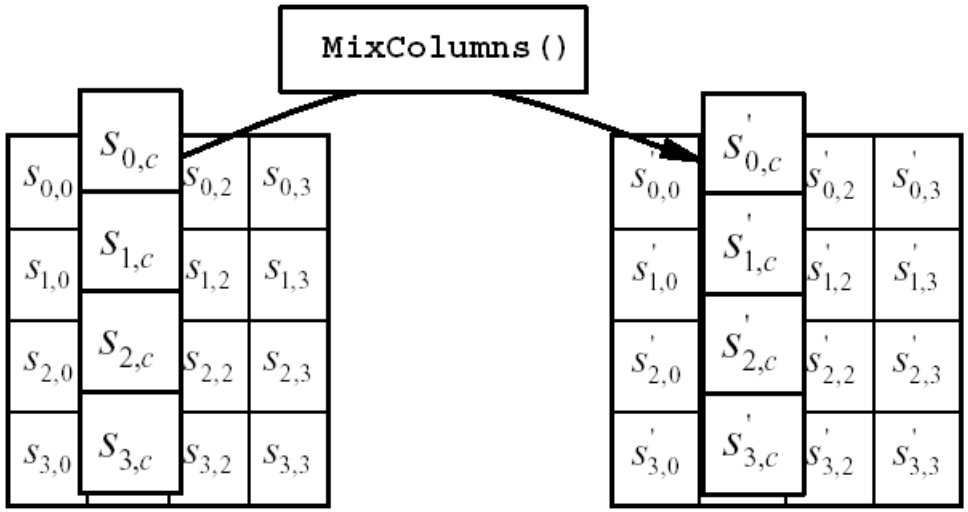
az oszlop polinom szorzása

$$s'(x) = a(x) \otimes s(x)$$

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$$

$$a^{-1}(x) = \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\}$$

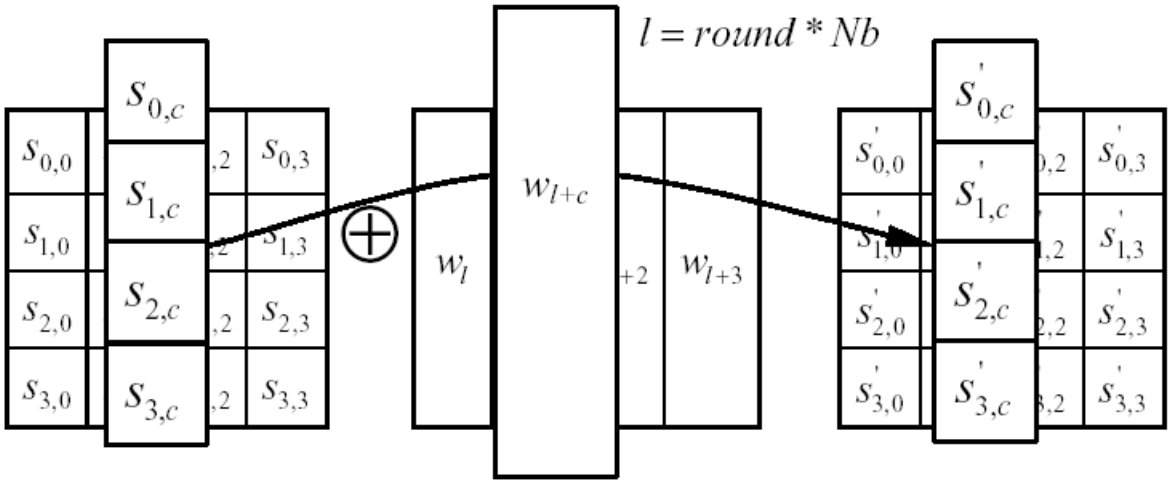
$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$



Advanced Encryption Standard

kulcs hozzáadás (AddRoundKey):

$$[s'_{0,c}, s'_{1,c}, s'_{2,c}, s'_{3,c}] = [s_{0,c}, s_{1,c}, s_{2,c}, s_{3,c}] \oplus [w_{round*Nb+c}]$$



Nb(Nr+1) 4 byte-os kulcsot képez az Nk x 4 byte-os kulcsból

Advanced Encryption Standard

kulcs kiterjesztés (Key Expansion):

```
KeyExpansion(byte key[4*Nk], word w[Nb*(Nr+1)], Nk)
begin
   word  temp

   i = 0

   while (i < Nk)
      w[i] = word(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3])
      i = i+1
   end while

   i = Nk

   while (i < Nb * (Nr+1)]
      temp = w[i-1]
      if (i mod Nk = 0)
         temp = SubWord(RotWord(temp)) xor Rcon[i/Nk]
      else if (Nk > 6 and i mod Nk = 4)
         temp = SubWord(temp)
      end if
      w[i] = w[i-Nk] xor temp
      i = i + 1
   end while
end
```

Advanced Encryption Standard

Cipher Key = 2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c

w_0 = 2b7e1516 w_1 = 28aed2a6 w_2 = abf71588 w_3 = 09cf4f3c

i (dec)	temp	After RotWord()	After SubWord()	Rcon[i/Nk]	After XOR with Rcon	w[i-Nk]	w[i] = temp XOR w[i-Nk]
4	09cf4f3c	cf4f3c09	8a84eb01	01000000	8b84eb01	2b7e1516	a0fafe17
5	a0fafe17					28aed2a6	88542cb1
6	88542cb1					abf71588	23a33939
7	23a33939					09cf4f3c	2a6c7605
8	2a6c7605	6c76052a	50386be5	02000000	52386be5	a0fafe17	f2c295f2
9	f2c295f2					88542cb1	7a96b943
10	7a96b943					23a33939	5935807a
11	5935807a					2a6c7605	7359f67f
12	7359f67f	59f67f73	cb42d28f	04000000	cf42d28f	f2c295f2	3d80477d
13	3d80477d					7a96b943	4716fe3e
14	4716fe3e					5935807a	1e237e44

Advanced Encryption Standard

Input = 32 43 f6 a8 88 5a 30 8d 31 31 98 a2 e0 37 07 34

Cipher Key = 2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c

Round Number	Start of Round	After SubBytes	After ShiftRows	After MixColumns	Round Key Value	
input	32 88 31 e0 43 5a 31 37 f6 30 98 07 a8 8d a2 34				2b 28 ab 09 7e ae f7 cf 15 d2 15 4f 16 a6 88 3c	⊕ =
1	19 a0 9a e9 3d f4 c6 f8 e3 e2 8d 48 be 2b 2a 08	d4 e0 b8 1e 27 bf b4 41 11 98 5d 52 ae f1 e5 30	d4 e0 b8 1e bf b4 41 27 5d 52 11 98 30 ae f1 e5	04 e0 48 28 66 cb f8 06 81 19 d3 26 e5 9a 7a 4c	a0 88 23 2a fa 54 a3 6c fe 2c 39 76 17 b1 39 05	⊕ =
2	a4 68 6b 02 9c 9f 5b 6a 7f 35 ea 50 f2 2b 43 49	49 45 7f 77 de db 39 02 d2 96 87 53 89 f1 1a 3b	49 45 7f 77 db 39 02 de 87 53 d2 96 3b 89 f1 1a	58 1b db 1b 4d 4b e7 6b ca 5a ca b0 f1 ac a8 e5	f2 7a 59 73 c2 96 35 59 95 b9 80 f6 f2 43 7a 7f	⊕ =

Blokk rejtjelezők

PPKE, ITK
Csapodi Márton

Definíciók

n: nyílt és rejtjelezett szöveg blokkmérete, általában nincs expanzió

K: k bites kulcs, véletlenszerűen kiválasztva

a tényleges kulcsméret lehet <k

blokk rejtjelező: nyílt szöveg és kulcs leképezése rejtjelezett szövegre, mely minden kulcs esetén invertálható

rejtjelezés: $C=E_K(P)$

dekódolás: $P=D_K(C)$

Az összes lehetséges P->C bijekció száma: $2^n!$

valódi véletlen rejtjelező: minden lehetséges bijekciót megvalósít, óriási kulcsméretet eredményez, ha n nem "túl kicsi"

praktikusan: véletlennek tűnő rejtjelező (véletlen kulcs esetén, vagy a kulcs ismerete nélkül)

Támadások

A blokk rejtjelező a bizalmasságot védi

passzív támadás <-> aktív támadás
A passzív támadás fenyegeti a bizalmasságot

Feltételezés:

- minden rejtjelezett szöveg megismerhető
- csak a kulcs ismeretlen (Kerckhoff feltétel)

További osztályozás: mit ismer még a rejtjelezett szövegen kívül:

- ciphertext-only
- known-plaintext
- (adaptive) chosen-plaintext
- (adaptive) chosen-ciphertext

Támadások

tervezési elv: chosen-plaintext támadás ellen legyen védett

rejtjelező biztonsága: a leghatékonyabb támadás komplexitásával arányos

komplexitás szempontjai:

- data complexity
- storage complexity
- processing complexity

kimerítő keresés:

data/storage: max 2^n

processing: max 2^k

Jó rejtjelező: mindegyik komplexitás közel a határhoz

Kimerítő kulcskeresés

Ismert nyílt szöveg (known-plaintext) vagy redundáns nyílt szöveg esetén nagyságrendileg 2^{k-1} próbálkozással a kulcs meghatározható

Példa (redundáns nyílt szöveg):

8 ASCII karakterből + karakterenként paritás bitből álló blokk (64 bit)

k=56 (DES)

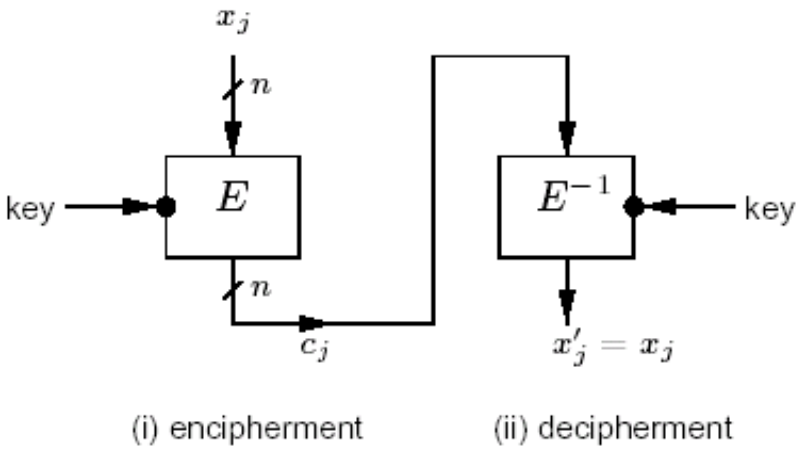
Rossz kulccsal is lehet korrekt paritású nyílt szöveget kapni, de további rejtjelszöveg blokkok dekódolása kiszűri a rossz kulcsot néhány blokk után

Működési módok / ECB

ECB: Electronic Codebook

jellemzők:

- azonos nyílt szöveg (és kulcs)
  -> azonos rejtjelezett szöveg
- blokkok függetlenek egymástól
- hiba esetén csak egy blokk hibás



korlátozásokkal alkalmazható

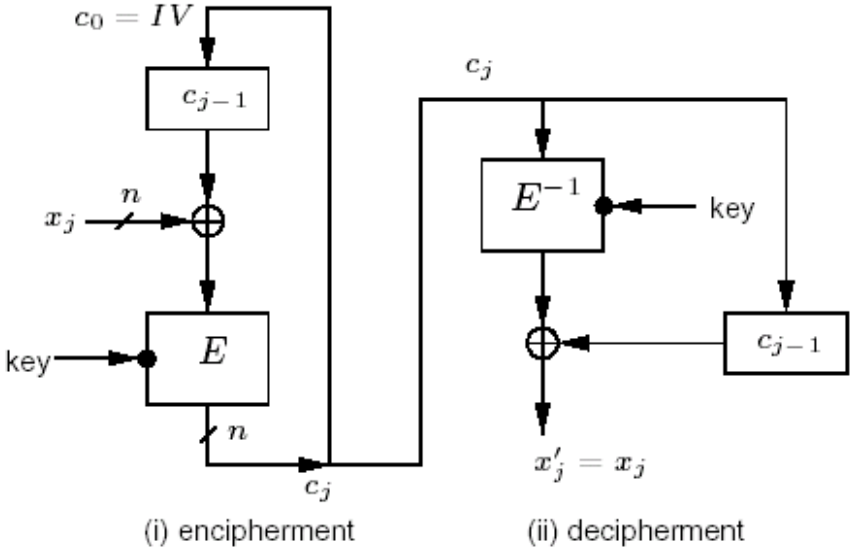
véletlen "padding" segít

Működési módok / CBC

CBC: Cipher-block Chaining

jellemzők:

- azonos nyílt szöveg (és kulcs)
  -> eltérő rejtjelezett szöveg
  véletlen IV-vel
- dekódoláshoz kell az előző rejtjelszöveg
- hiba esetén csak két dekódolt blokk hibás



IV lehet nyílt, de integritása szükséges (1. blokk biztonságos dekódolása miatt)

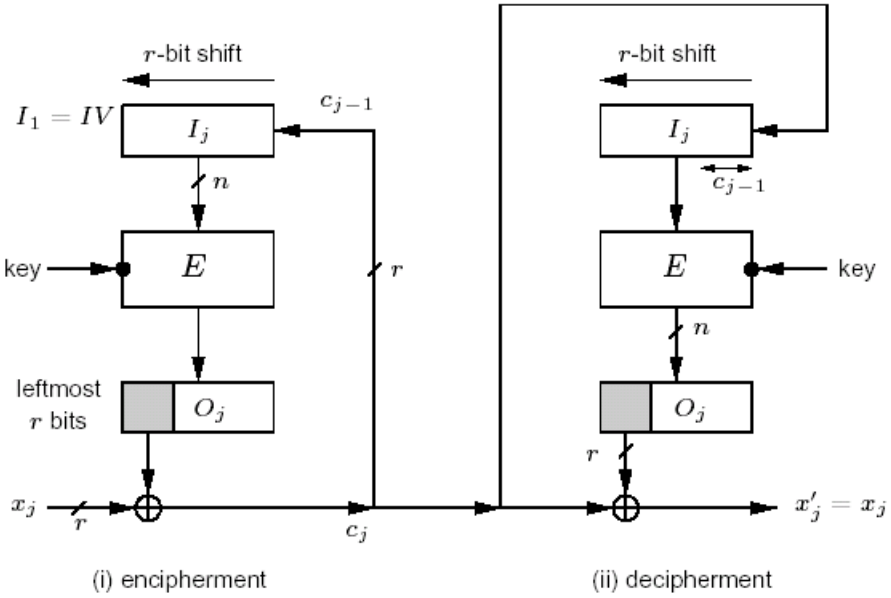
Működési módok / CFB

CFB: Cipher Feedback

jellemzők:

- azonos nyílt szöveg (és kulcs)
  -> eltérő rejtjelezett szöveg
  véletlen IV-vel
- dekódoláshoz kell
  az előző néhány rejtjelszöveg
- hiba esetén csak néhány dekódolt blokk hibás
- alacsonyabb hatásfok r<n miatt

IV lehet nyílt

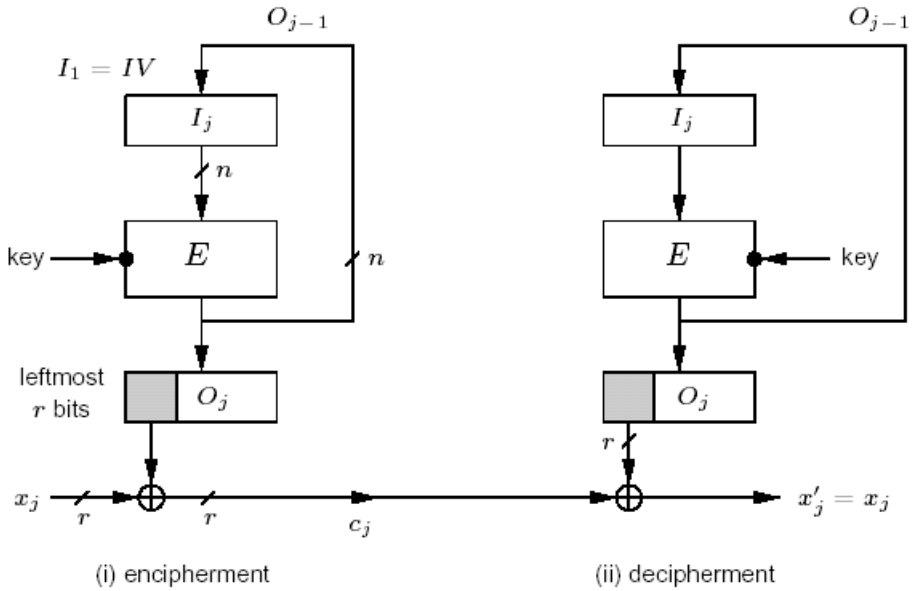


Működési módok / OFB

OFB: Output Feedback

jellemzők:

- azonos nyílt szöveg
  -> eltérő rejtjelezett szöveg
  véletlen IV-vel
- a dekódoló kulcsfolyam független
  a rejtjelszövegtől
- bithiba esetén csak egy dekódolt blokk hibás, de egy rejtjelszöveg bit kiesése esetén kiesik a szinkronból



IV lehet nyílt, de változtatni kell a kulcs újbóli használatakor

Visszacsatolás helyett lehet egy számláló is (CTR mode)

Működési módok

Megjegyzések:

1.

OFB, CTR: Szinkron adatfolyam rejtjelező kulcsgenerátora

CFB: Aszinkron adatfolyam rejtjelező

2.

CFB és OFB (CTR) módok esetén csak rejtjelezés (E(P)) számítása szükséges, erre érdemes optimalizálni (pl. AES)

Jellegzetes blokk rejtjelezők

szorzat rejtjelező (product cipher): helyettesítés, áthelyezés, lineáris leképezés, aritmetikai művelet, moduláris szorzás egymás utáni végrehajtása

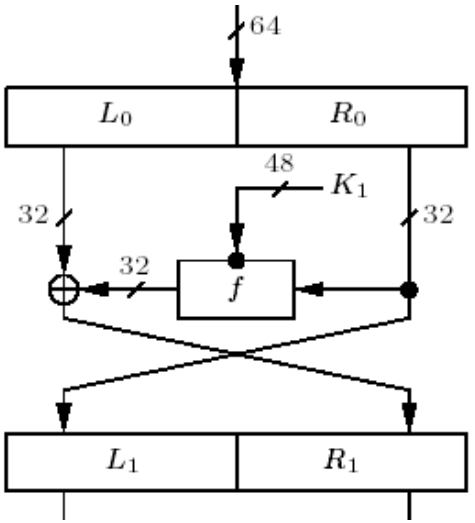
iterált rejtjelező: egy belső invertálható funkció ismétlése, minden lépésben a kulcsból generált alkulcs alkalmazásával

helyettesítéses-permutációs hálózat (substitution-permutation network): iterált szorzat rejtjelező egyszerű helyettesítésekből és permutációkból

Feistel rejtjelező: iterált rejtjelező, mely azáltal lesz invertálható, hogy minden lépésben tetszőleges f-re:

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$$



DES

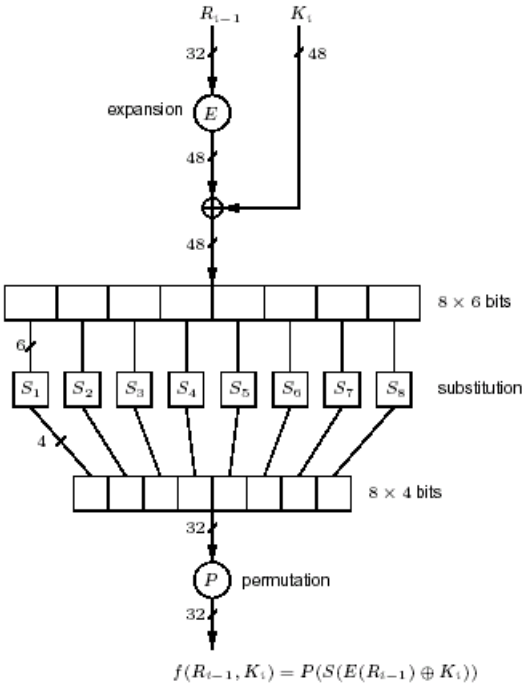
Feistel rejtjelező

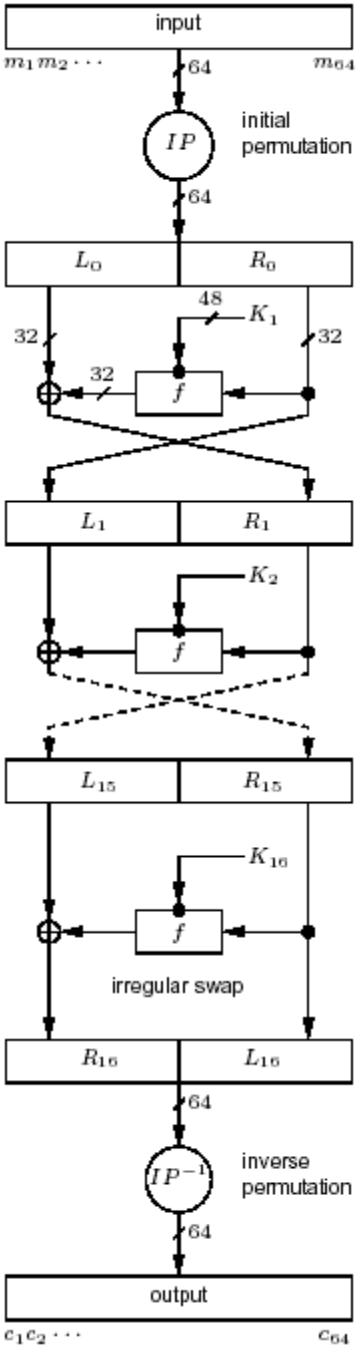
blokkméret: n=64

kulcsméret: k=64 bit, de a tényleges méret: 56 bit

8 paritás bit miatt

16 iteratív lépés





DES

Tulajdonságok:

minden rejtjelszöveg bit függ minden nyílt szöveg bittől: egy bit megváltoztatása minden rejtjel bitet 1/2 valószínűséggel változtat meg

nem jósolható, hogy egy rejtjelszöveg bit változtatása milyen hatással lesz a dekódolt szövegre

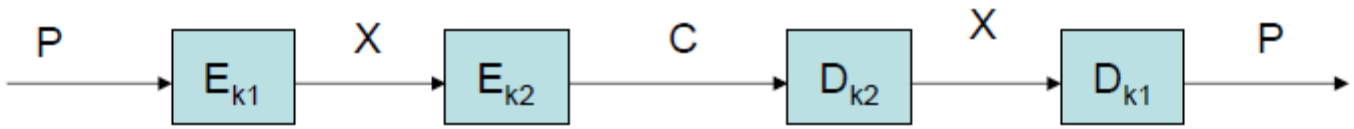
Nem csoporttulajdonságú: kompozícióra nem zárt, tehát két különböző kulccsal történő rejtjelezés egymás után nem váltható ki egy harmadik kulccsal történő rejtjelezésre

Emiatt van lehetőség többszörös rejtjelezésre (3DES):

- DES: nem biztonságos
- double DES: nincs értelme (meet-in-the-middle támadás, 2^{57} művelet)
- triple DES: kettő vagy három kulccsal (itt is van meet-in-the-middle támadás), de lassú

Double DES: Meet in The Middle Attack

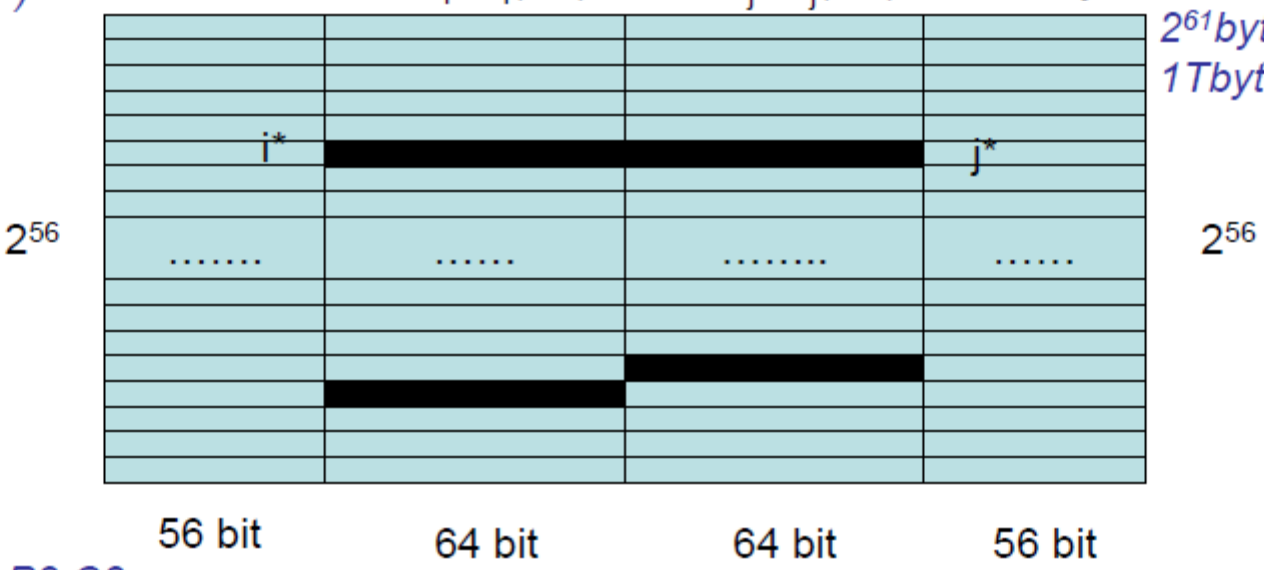
Known Plaintext Attack, two pairs: P1;C1 & P2;C2



$X=E_{k1}(P)$ $X=D_{k2}(C)$

First Pair: P1;C1
To find keys (i;j*)*
• Xi= Xj**

Total Memory
$256x2^{56}=2^{64}$bit
2^{61}byte~10^{18}byte
1Tbyte x 10^6run

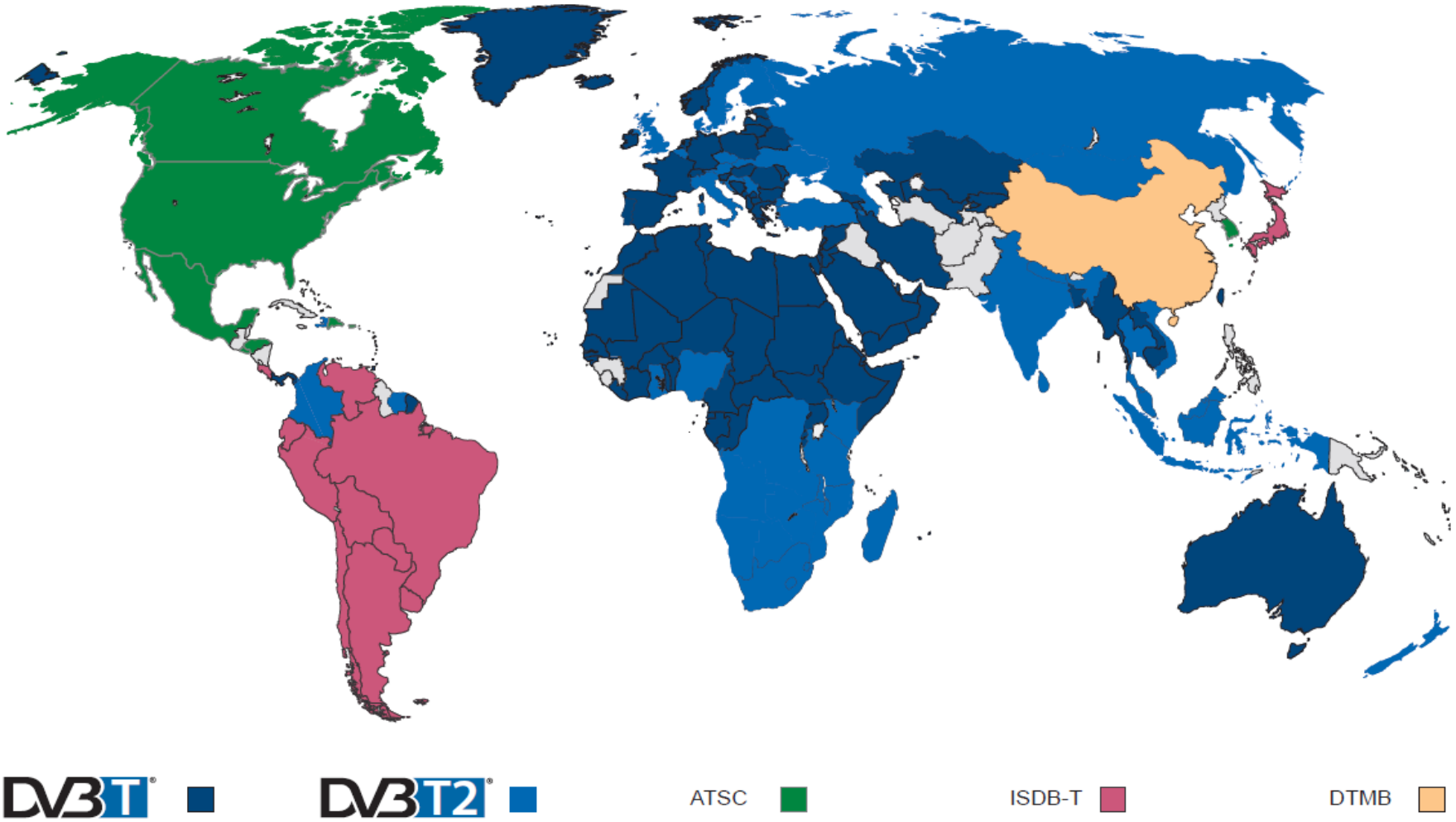


Second Pair: P2;C2
To check keys (i;j*): P2=Di*[Dj*(C2)]; C2=Ej*[Ei*(P2)]; Average Run # = 2x 2^{56}/2= 2^{56}*

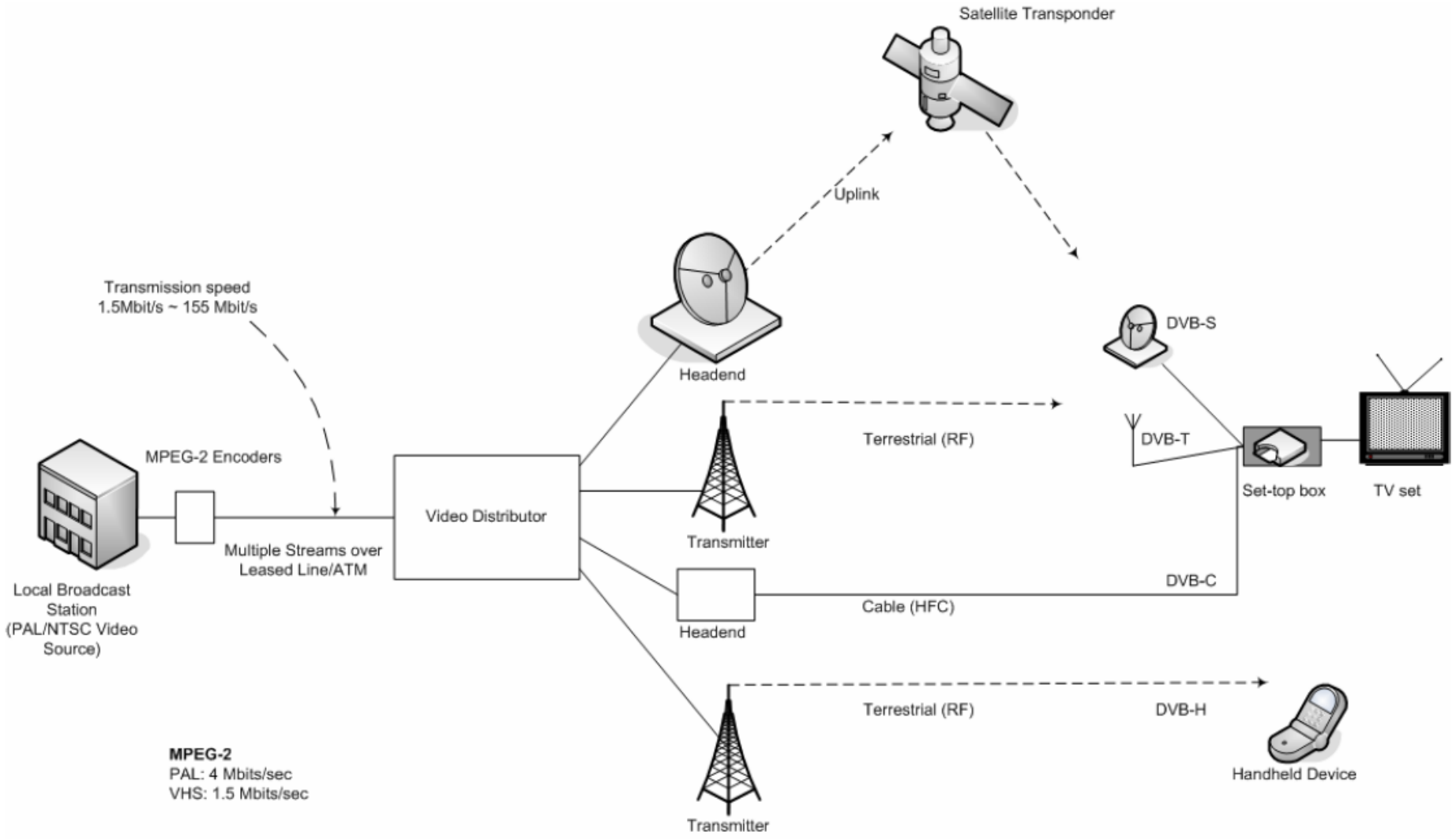
DVB Conditional Access

PPKE, ITK
Csapodi Márton

DVB – Digital Video Broadcasting



DVB – Digital Video Broadcasting



DVB Magyarországon

digitális átállás: az analóg műsorterjesztés helyébe a digitális műsorterjesztés lép

~~2011. 2012.~~ december 31.: analóg lekapcsolás végdátuma, megszűnik az analóg földfelszíni sugárzású televíziós műsorszórás, és helyébe a digitális földfelszíni sugárzás (DVB-T) lép

5/2013. (III. 11.) NMHH rendelet:
két ütemben, 2013. július 31-én és 2013. október 31-én

hagyományos szoba- vagy tetőantenna mellé:

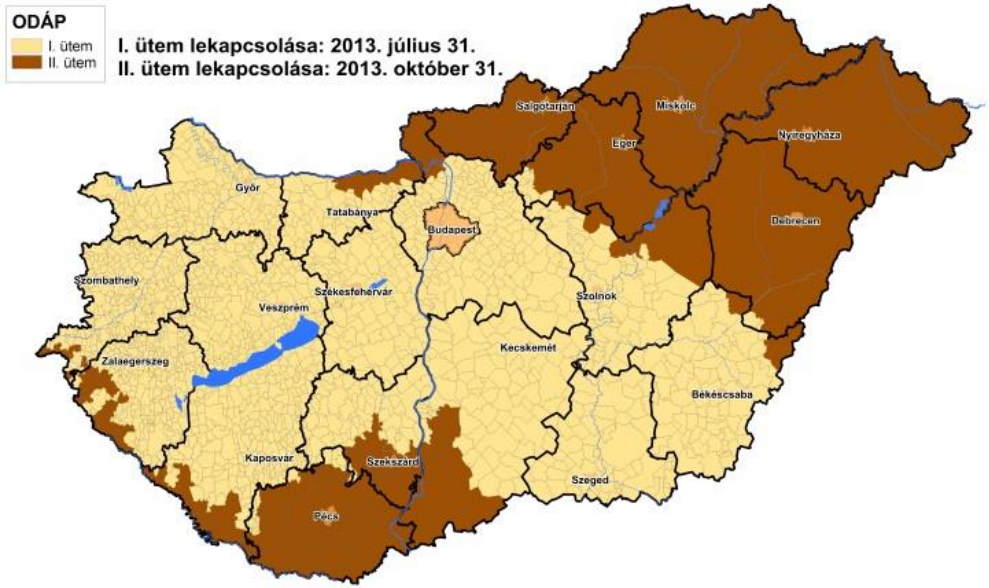
- set-top-box + hagyományos TV
- új TV beépített vevőegységgel

2013 februári előfizetők (szabad hozzáférésű nélkül):

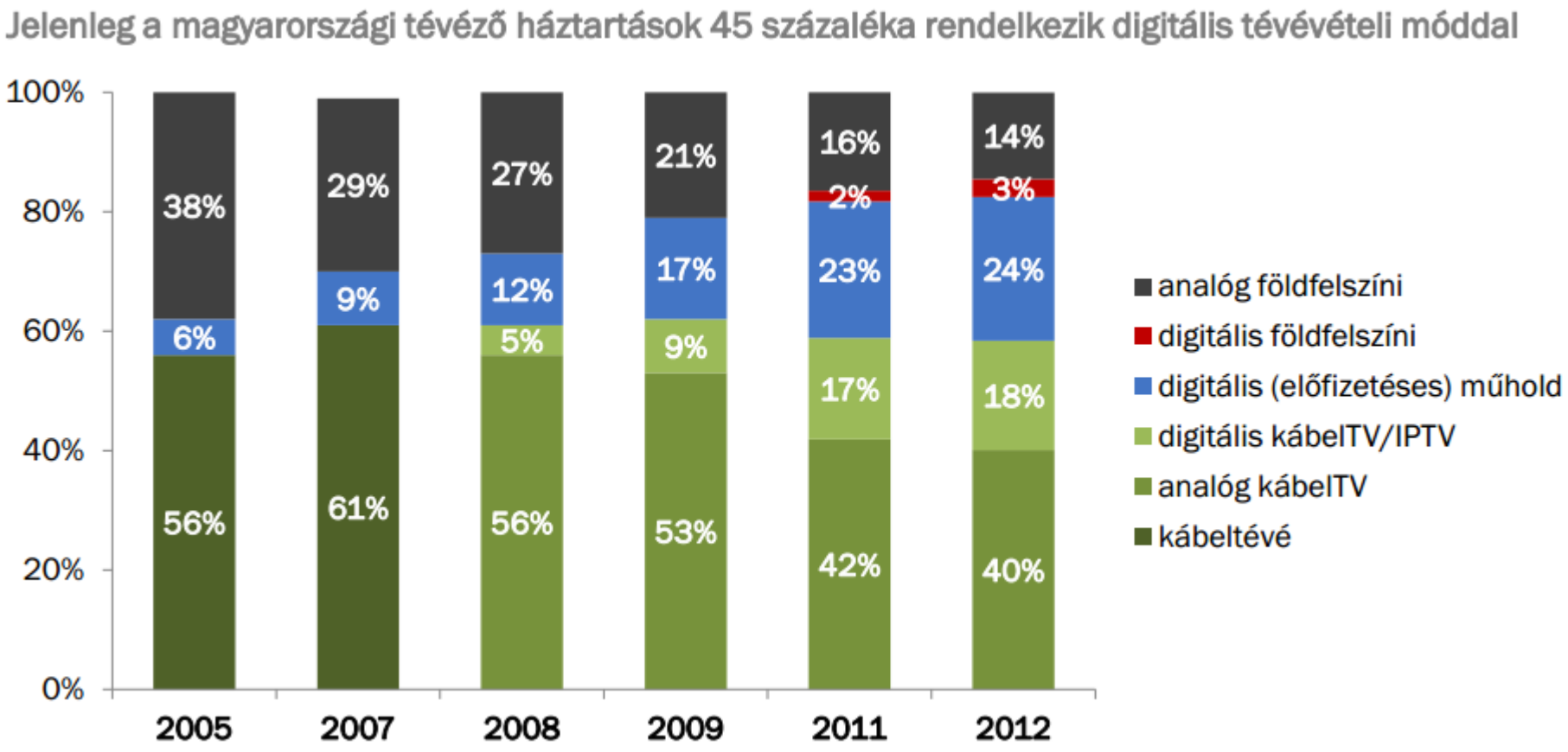
- DVB-C: 900e
- DVB-S: 900e
- DVB-T: 85e
- analóg vezetékes: 1.050e

szabad hozzáférésű analóg:

- 500e (becslés)

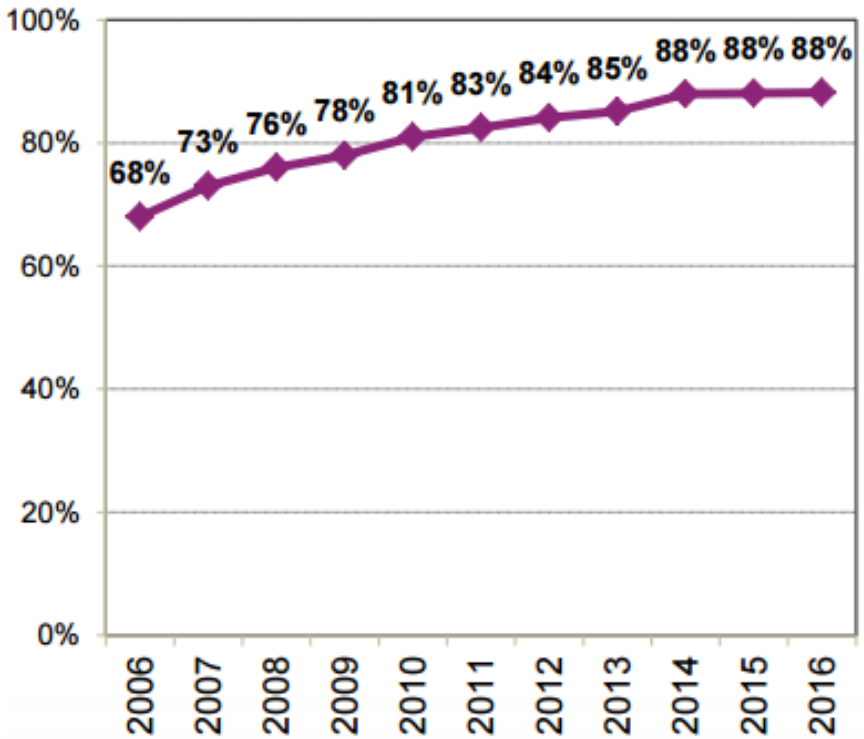


DVB Magyarországon (múlt)



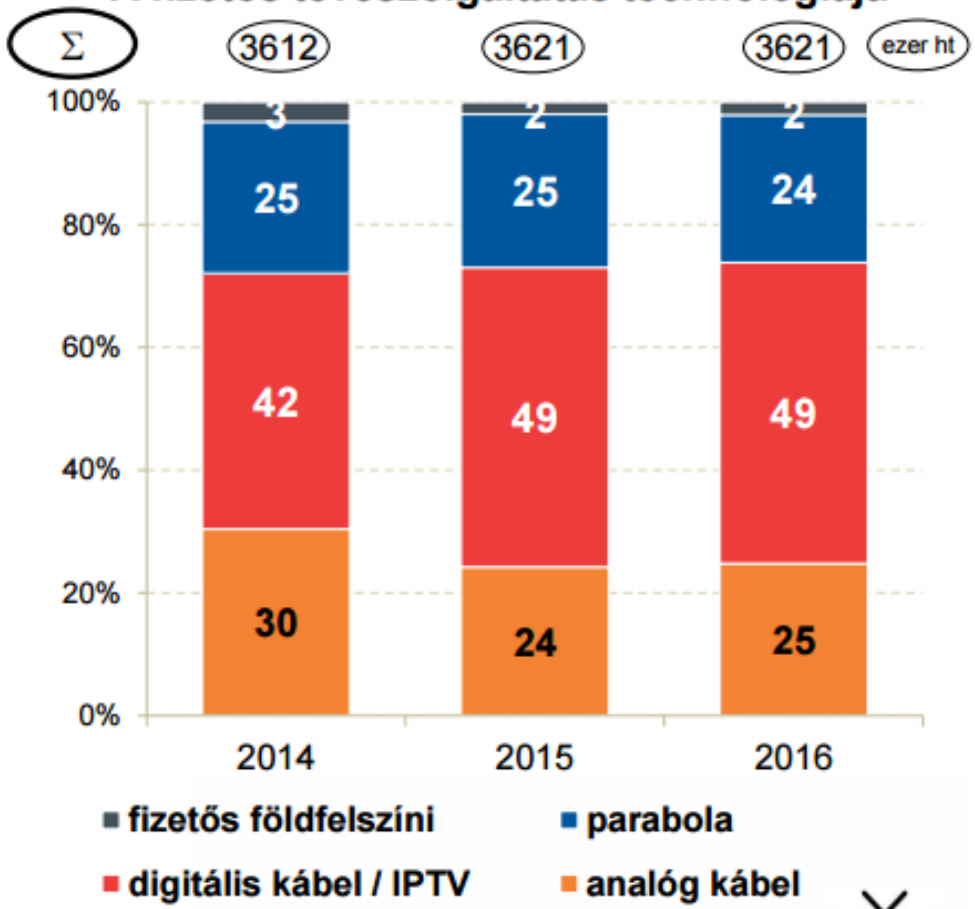
DVB Magyarországon (jelen)

A háztartások ellátottsága fizetős tévével



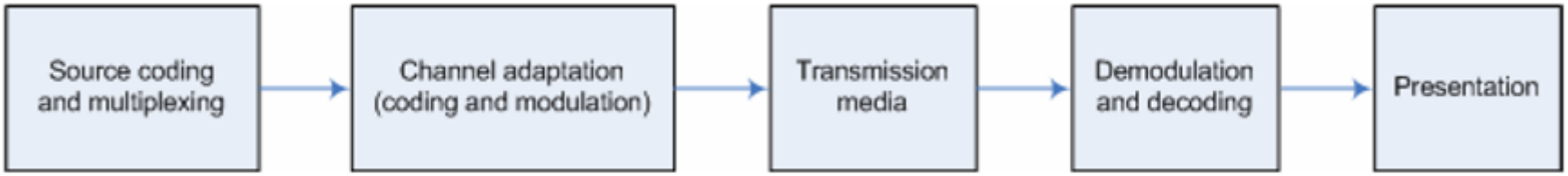
Bázis: Összes háztartás (N=4,106 millió; n=2020)

A fizetős tévészolgáltatás technológiája



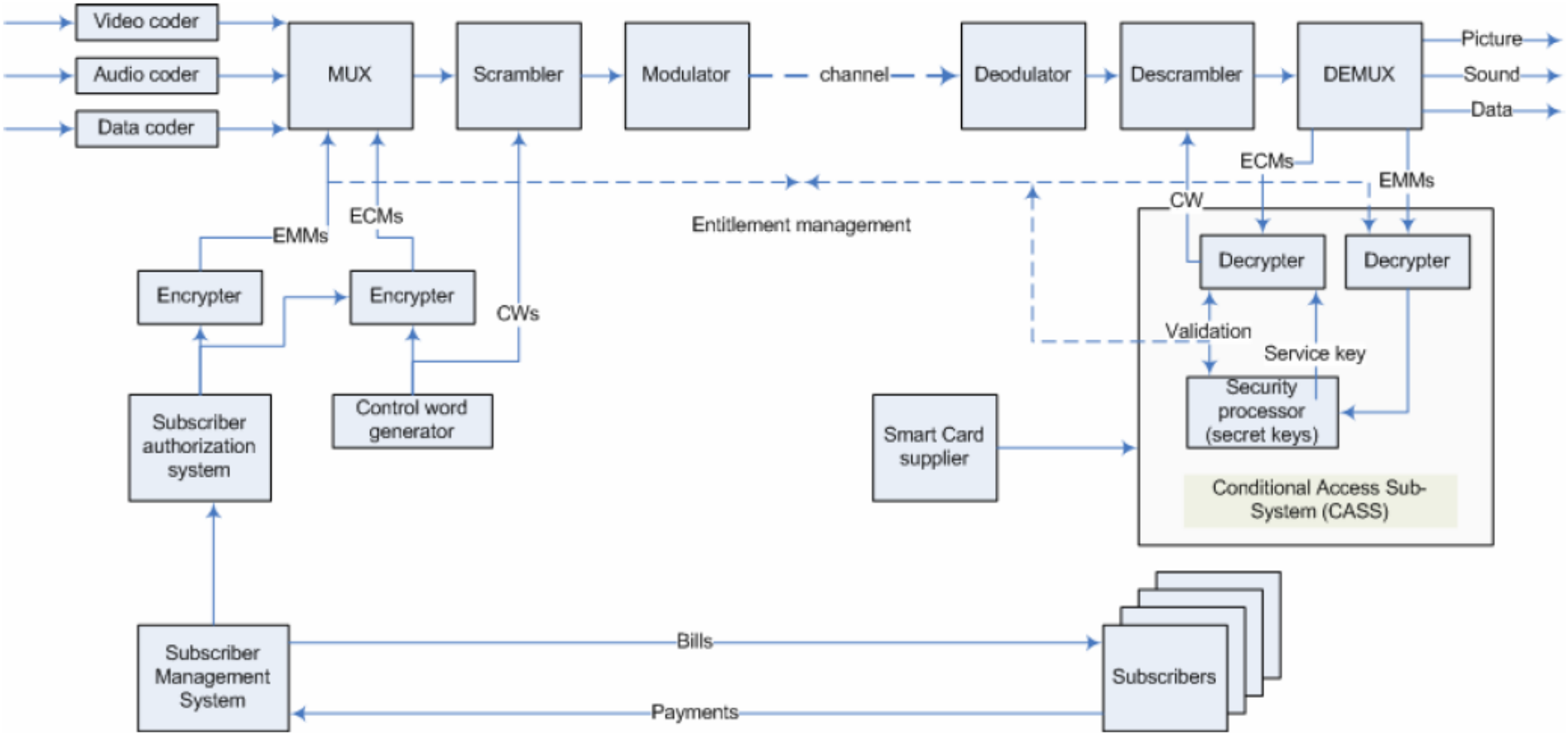
Bázis: Fizetős tévével rendelkező háztartások
(N=3,621 millió HT; n=1776)

DVB átvitel



source coding & multiplexing: MPEG Transport Stream létrehozása
MPEG-2 vagy MPEG-4 (pl. Magyaro.)
channel adaptation: az átvivő közegtől függően
transmission media: kábel, műhold, földi sugárzás, mobil
demodulation & decoding: MPEG-TS visszanyerése
presentation: függ a készülék, kábel képességeitől

DVB hozzáférés védelem



DVB hozzáférés védelem

DVB-CA (Conditional Access): fizetős adások hozzáférés védelmi mechanizmusa

DVB-CSA (Common Scrambling Algorithm): az adás rejtjelezésének algoritmusa

CW (Control Word): vezérlőszó, gyakran változik, ezzel történik az MPEG-TS titkosítása

ECM (Entitlement Control Message): jogosultság vezérlő üzenet, a szolgáltatási kulccsal titkosított CW

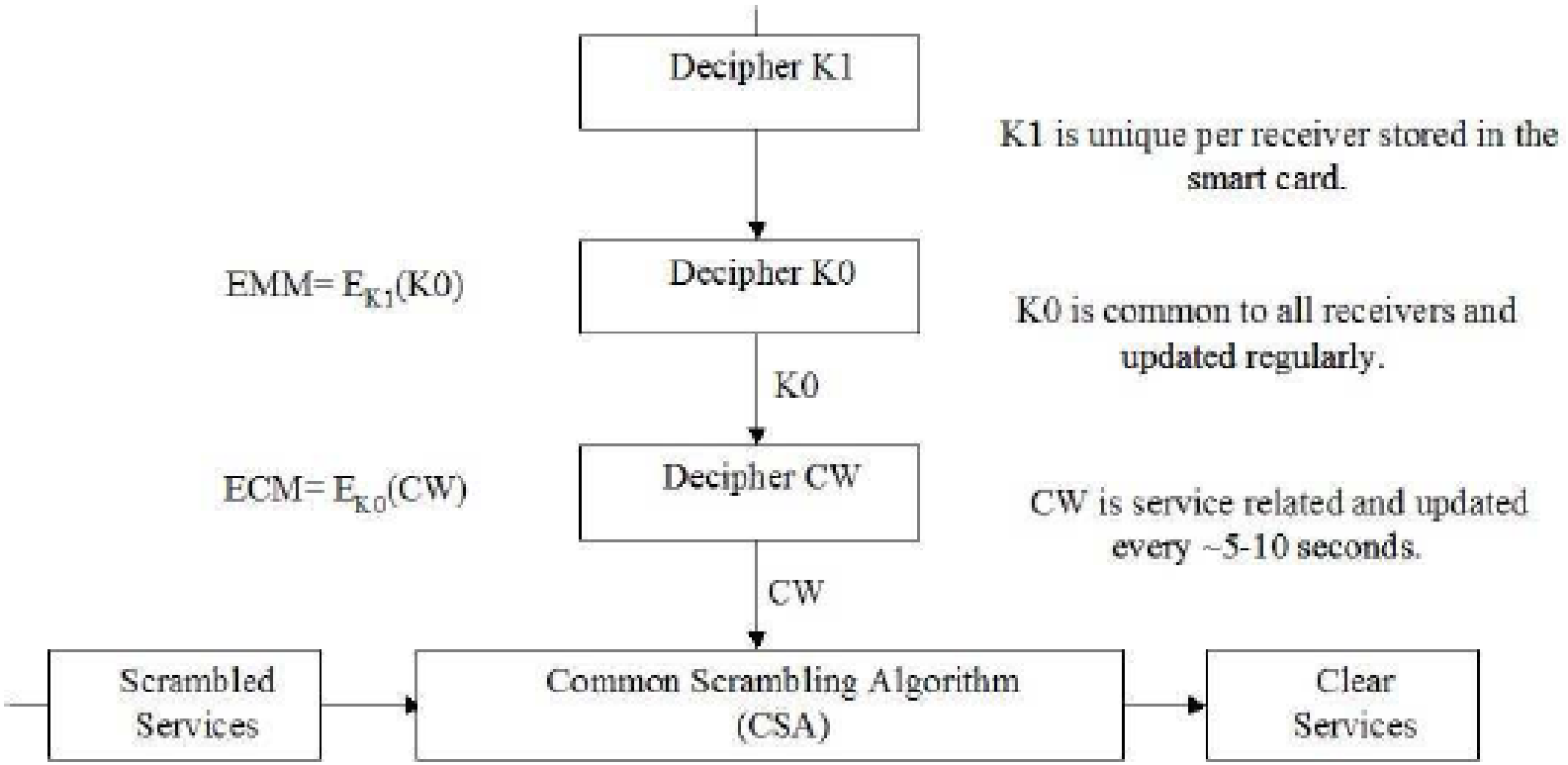
szolgáltatási kulcs: K0, ritkán változik, adott szolgáltatás tekintetében egységes mindenkinek

EMM (Entitlement Management Message): jogosultság kezelő üzenet, a felhasználói kulccsal titkosított szolgáltatási kulcs

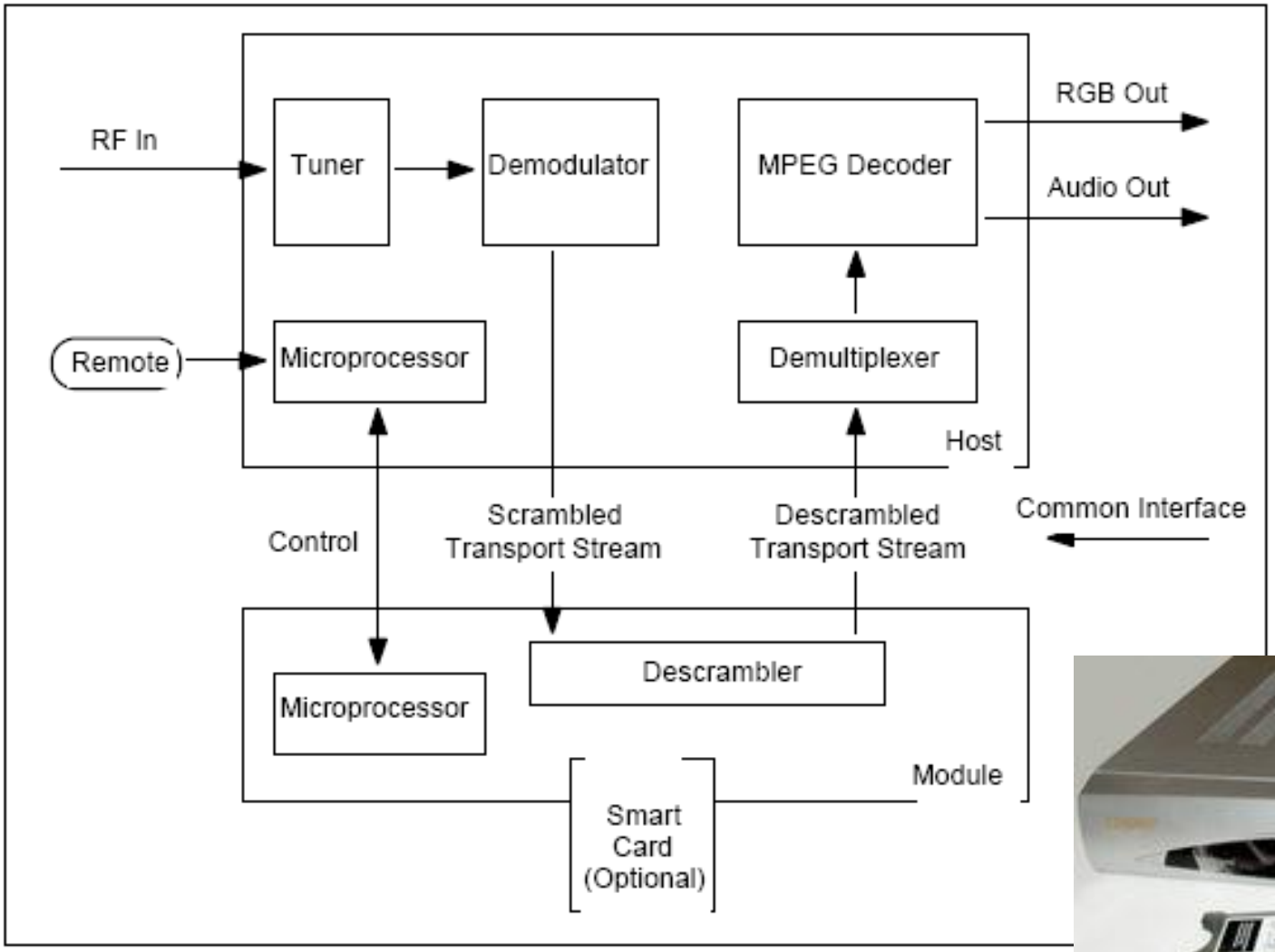
felhasználói kulcs: K1, állandó, készülékenként eltérő (valójában a felhasználók egy csoportja azonos kulccsal rendelkezik)

ECM és EMM az MPEG-TS-be beépül

DVB – Common Scrambling Algorithm



DVB – Common Interface



DVB – Common Interface

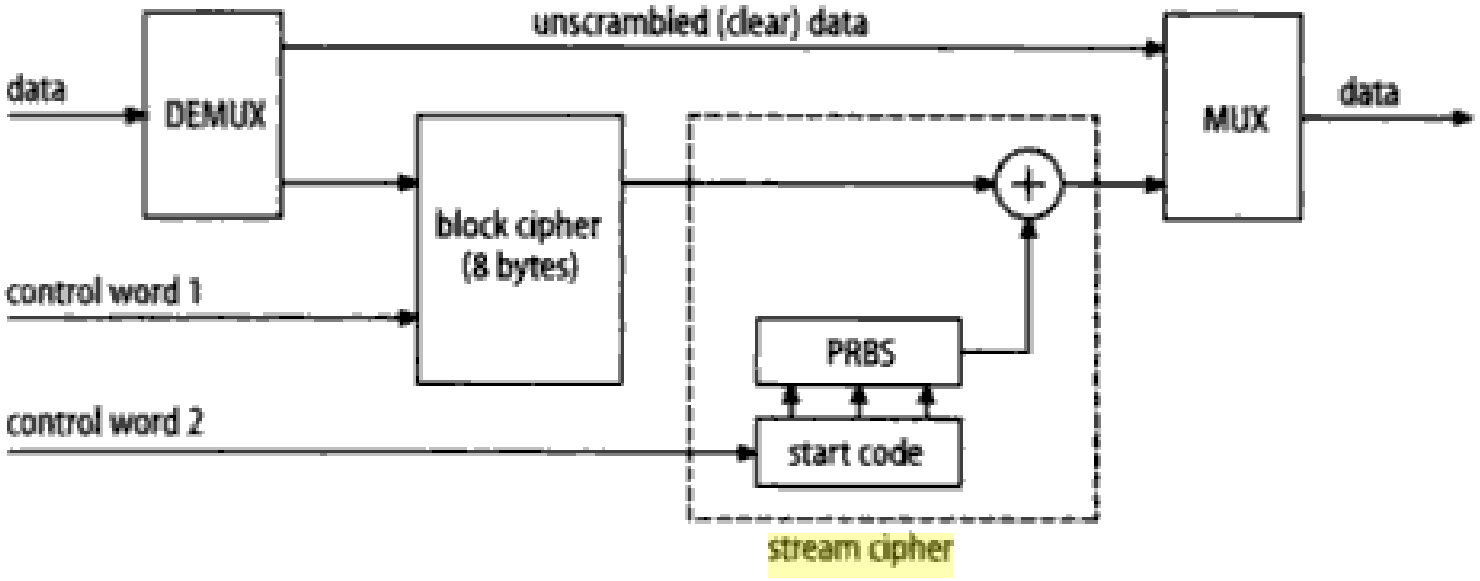
A set-top-box-ról, TV készülékről leválasztásra kerülnek a hozzáférés védelmi funkciók.

Célja: könnyebb szolgáltatóváltás

A Conditional Access alrendszer a PCMCIA szabványnak felel meg és tartalmazhat egy intelligens kártyát a felhasználói kulcs biztonságos tárolására.

A szolgáltatói kulcs és a vezérlőszó meghatározása, valamint a rejtjelezés dekódolása itt történik.

DVB – Common Scrambling Algorithm



ETSI szabvány, csak gyártók számára hozzáférhető

2002-ben megjelent egy szoftver implementációja a DVB-CA-nak

Rövid idő múlva visszafejtették

DVB biztonsági problémák

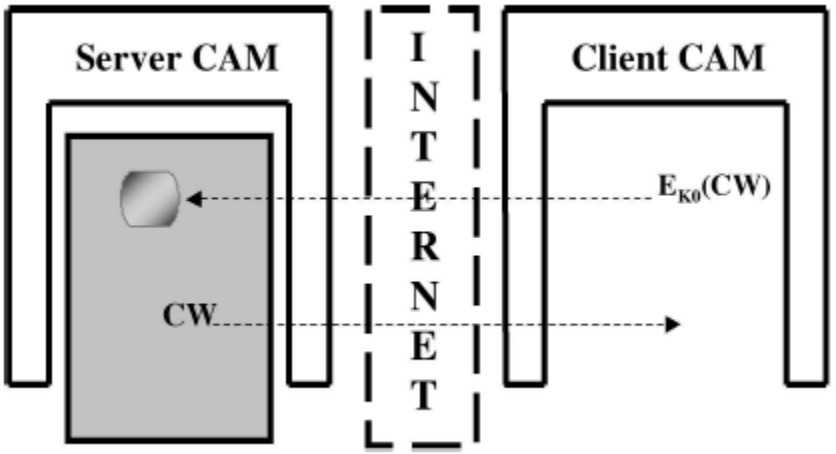
Jobb, mint az analóg hozzáférésvédelem, ahol elég volt egyféle kalózkártya

Itt a kulcsok folyamatos cseréjén és a különböző felhasználói kulcsokon alapul a védelem.

Problémák:

- mindenkihez eljut az adás
- nincs visszafelé irány, nem hitelesíthető a set-top-box, nem tudni, hányan használják ugyanazt a kulcsot
- a set-top-box és a CA modul lehet PC alapú, programok telepíthetők rá

DVB – Card Sharing Attack



www.cardsharingserver.com

http://tvboom.net/en

http://blastoblogja.blogspot.hu/

Kártyamegosztás ellenszere

Control Word encryption:

EMM dekódolása és a K0 (szolgáltatási kulcs) tárolása a smart card-on történik, de a CW elhagyja a kártyát, mert az MPEG-TS dekódolás nem a smart card-on, hanem a Conditional Access Module-ban (CAM) történik. Ekkor a CW megszerezhető.
Ezt akadályozza meg, ha a kártya és a CAM között egy plusz titkosítást alkalmazunk.

DVB CSA3:

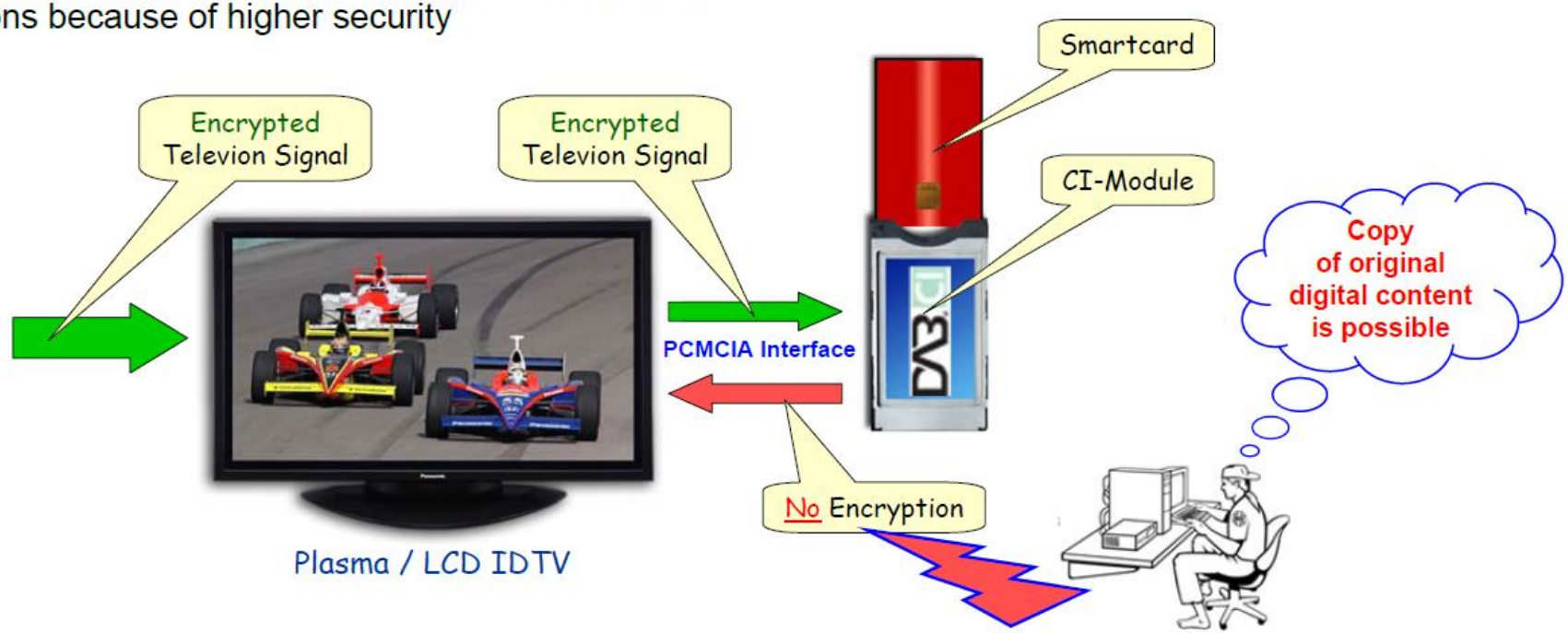
Új ETSI szabvány. Nem publikus a működése. „eXtended emulation Resistant Cipher”: hardverben gyors, szoftverben lassú a számítás.

Common Interface Plus (CI+):

Másolásvédelemre is alkalmazható. UPC ilyen CAM-ot ad az ügyfeleinek. Ha a TV is támogatja a szabványt, a CAM és a TV kölcsönösen hitelesítik egymást eszköz tanúsítványok alapján. A CAM és a TV között titkosított adatáramlás van.

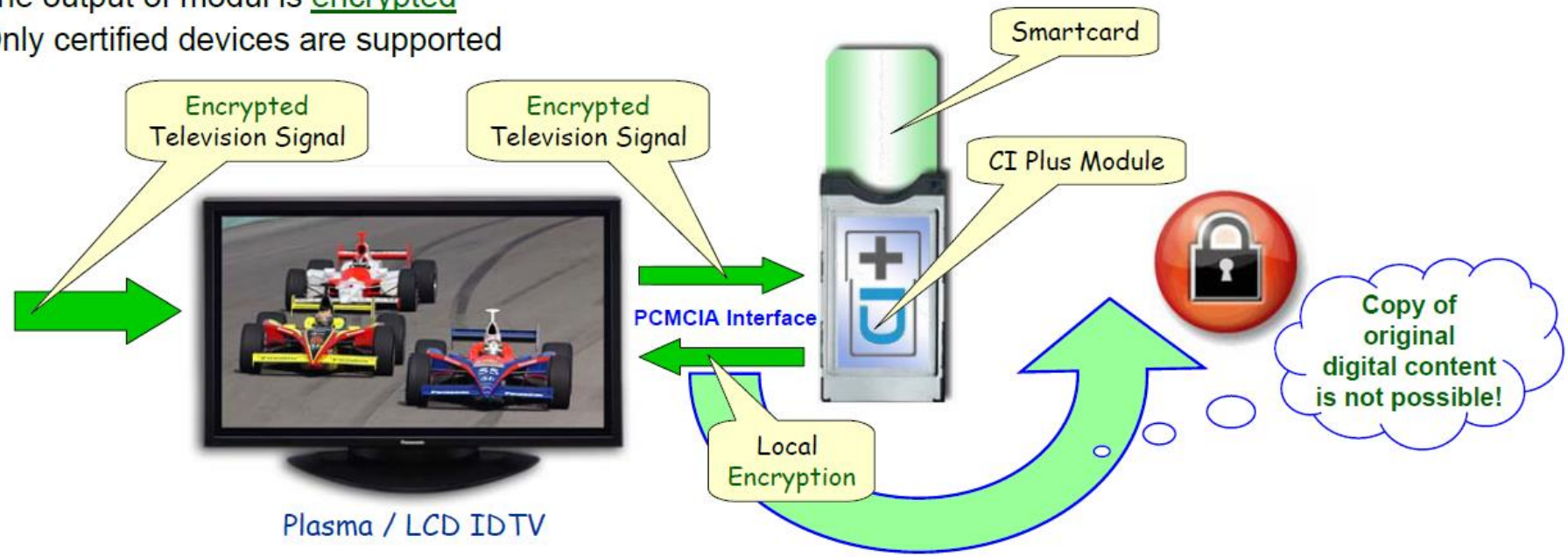
DVB CI - First Generation Standard v1

- CI-Module used with smartcard containing key-informationen
- CI-Module remove the encryption of protected content
- The output of CI-Module is unencrypted
- Due to this, most content providers prefer integrated solutions because of higher security

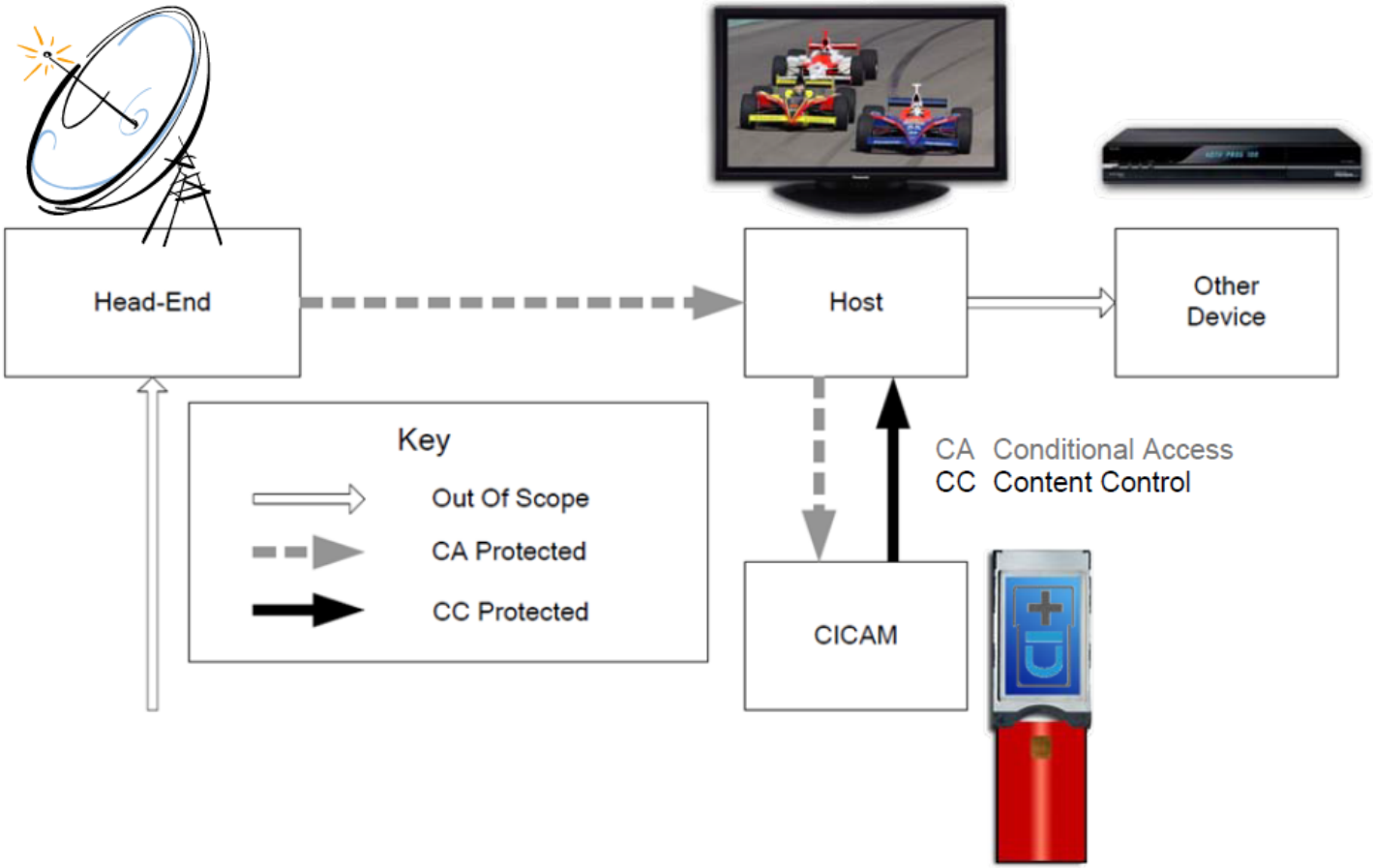


CI Plus - Protection of Content

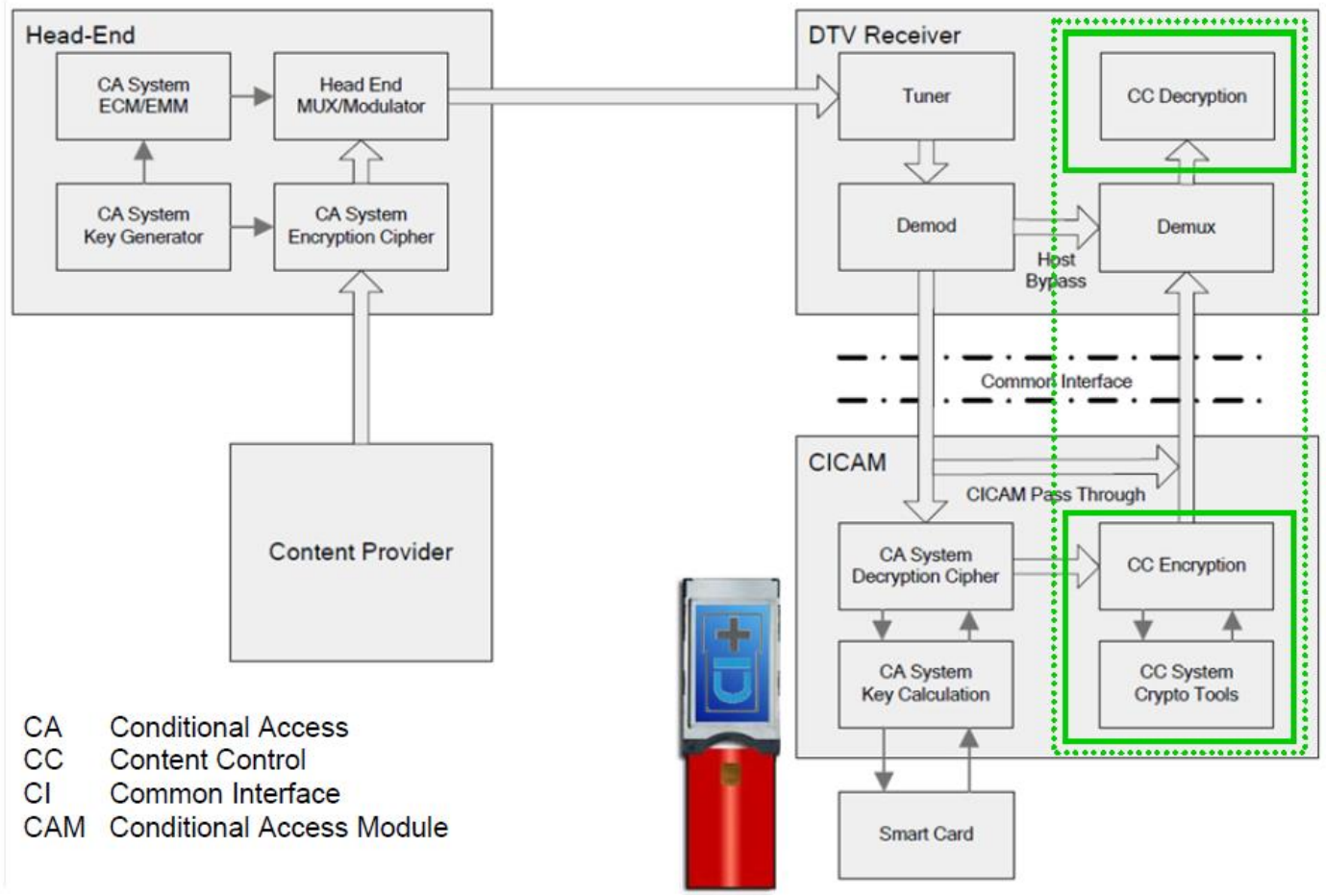
- Based on existing DVB-CI Standard
- Main requirement: achieving the same level of security as embedded solutions
- CI Plus Modul and Receiver
  - Calculation & Usage of a secure key for content protection
  - Secure, authentificated channel for critical system messages
- The output of modul is encrypted
- Only certified devices are supported



CI Plus - Scope of Protection



CI Plus - System Overview



CA Conditional Access
CC Content Control
CI Common Interface
CAM Conditional Access Module