Important Terms

computers interconnected by **communication network**
= **computer network** (of the first type)

computers providing switching in **communication network**
= **computer network** (of the second type)

distributed system
spatial
control and implementation structure

open system ≠ **public** system ≠ **open source** system

service integrated system

digital system

Threats and corresponding protection goals

threats: example: medical information system protection goals:

1) unauthorized access to information — confidentiality

computer company receives medical files

2) unauthorized modification of information — integrity ≅ partial correctness

undetected change of medication

≥ total correctness

3) unauthorized withholding of information or resources — availability for authorized users

detected failure of system

no classification, but pragmatically useful

example: unauthorized modification of a program

1) cannot be detected, but can be prevented; cannot be reversed

2)+3) cannot be prevented, but can be detected; can be reversed

Threats and corresponding protection goals

threats: | example: medical information system | protection goals:

1) unauthorized access to information — confidentiality

computer company receives medical files

2) unauthorized modification of information — integrity

undetected change of medication

≥ total correctness ≅ partial correctness

3) unauthorized withholding of information or resources — availability for authorized users

detected failure of system

no classification, but pragmatically useful

example: unauthorized modification of a program

1)	cannot be detected, but can be prevented;	cannot be reversed
2)+3)	cannot be prevented, but can be detected;	can be reversed

Definitions of the protection goals

confidentiality

Only authorized users get the information.

integrity

Information are correct, complete, and current
or this is detectably not the case.

availability

Information and resources are accessible where and
when the authorized user needs them.

- subsume: data, programs, hardware structure
- it has to be clear, who is authorized to do what in which situation
- it can only refer to the inside of a system

Protection against whom ?

Laws and forces of nature

- components are growing old
- excess voltage (lightning, EMP)
- voltage loss
- flooding (storm tide, break of water pipe, heavy rain)
- change of temperature ...

} fault tolerance

Human beings

- outsider
- user of the system
- operator of the system
- service and maintenance
- producer of the system
- designer of the system
- producer of the tools to design and produce
- designer of the tools to design and produce
- producer of the tools to design and produce the tools to design and produce
- designer ... includes user, operator, service and maintenance ... of the system used

} Trojan horse
- universal
- transitive

Considered maximal strength of the attacker

attacker model

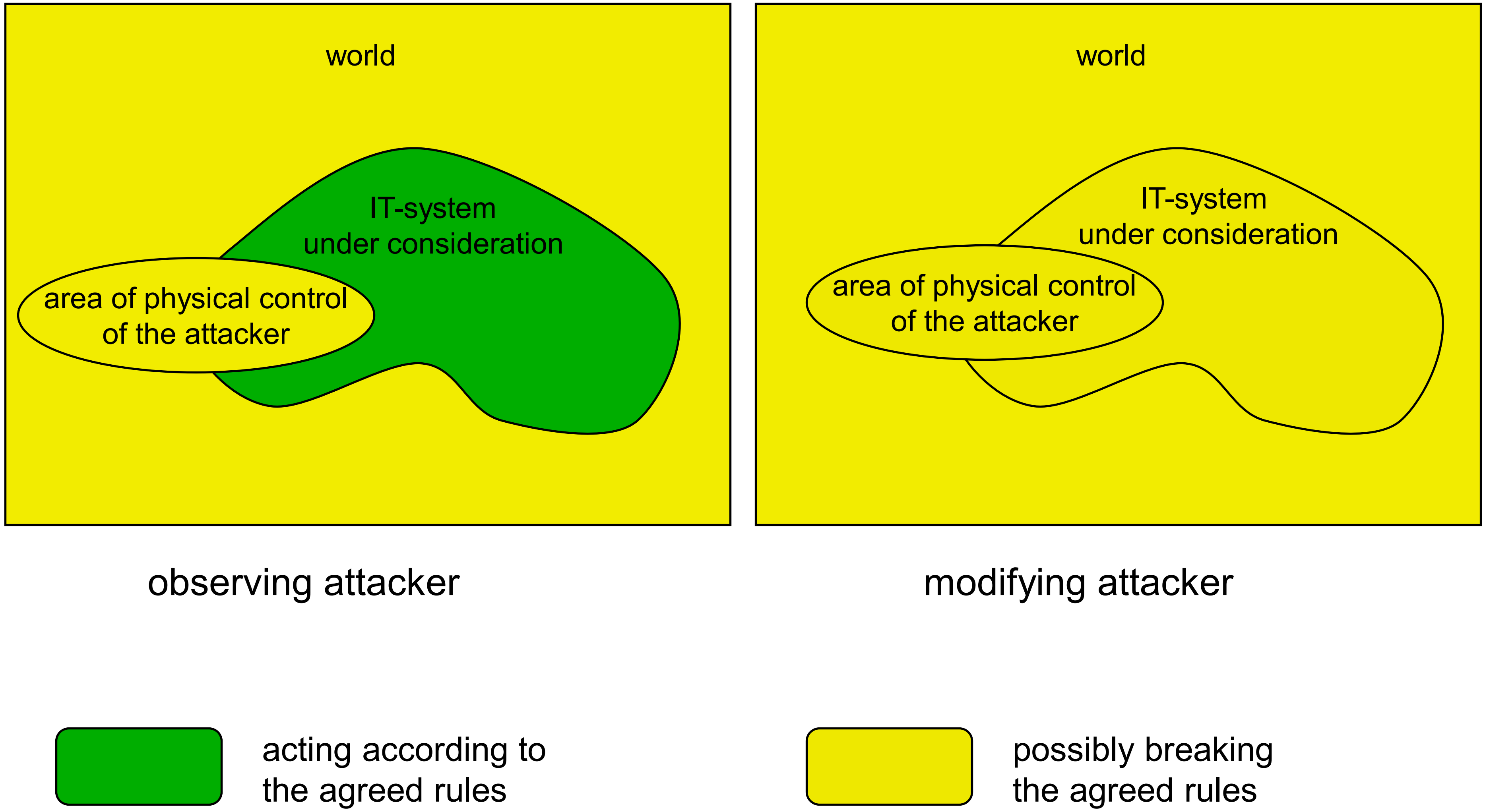
It‘s not possible to protect against an omnipotent attacker.

- roles of the attacker (outsider, user, operator, service and maintenance, producer, designer …), *also combined*
- area of physical control of the attacker
- behavior of the attacker
  - passive / active
  - observing / modifying (with regard to the agreed rules)
- stupid / intelligent
  - computing capacity:
    - not restricted: computationally unrestricted
    - restricted: computationally restricted

money

time

Observing vs. modifying attacker



Strength of the attacker (model)

Attacker (model) *A* is stronger than attacker (model) *B*, iff *A* is stronger than *B* in at least one respect and not weaker in any other respect.

Stronger means:

- set of roles of $A \supset$ set of roles of B,
- area of physical control of $A \supset$ area of physical control of B,
- behavior of the attacker
  - active is stronger than passive
  - modifying is stronger than observing
- intelligent is stronger than stupid
  - computing capacity: not restricted is stronger than restricted
- more money means stronger
- more time means stronger

Defines partial order of attacker (models).

Realistic protection goals/attacker models: Technical solution possible?

Security in computer networks

confidentiality

	• message content is confidential	**end-to-end encryption**
• **place**	• sender / recipient anonymous	**mechanisms to protect traffic data**

integrity

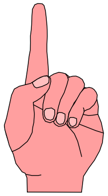
	• detect forgery	**authentication system(s)**
• **time**	• recipient can prove transmission	**sign messages**
• **time**	• sender can prove transmission	**receipt**
	• ensure payment for service	**during service by digital payment systems**

availability

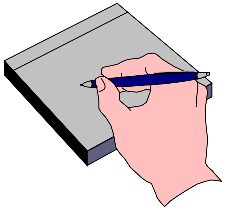
• enable communication	**diverse networks;** **fair sharing of resources**

Multilateral security

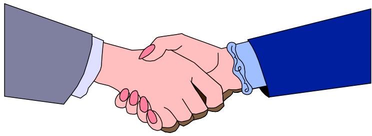
- Each party has its particular protection goals.



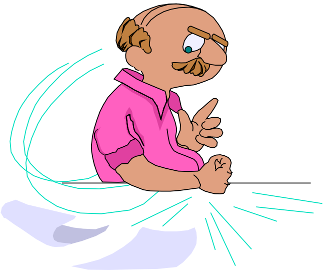
- Each party can formulate its protection goals.



- Security conflicts are recognized and compromises negotiated.



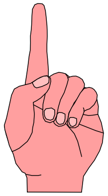
- Each party can enforce its protection goals within the agreed compromise.



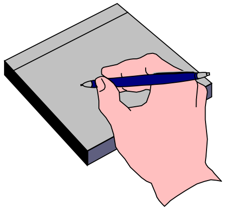
Security with minimal assumptions about others

Multilateral security (2nd version)

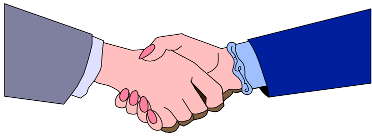
- Each party has its particular goals.



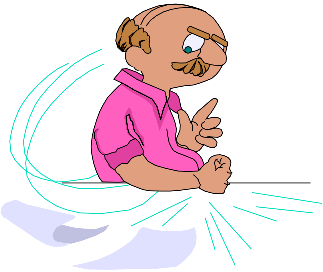
- Each party can formulate its protection goals.



- Security conflicts are recognized and compromises negotiated.



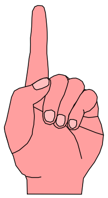
- Each party can enforce its protection goals within the agreed compromise.



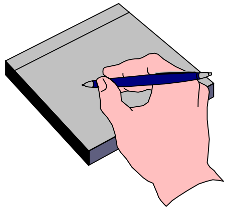
Security with minimal assumptions about others

Multilateral security (3rd version)

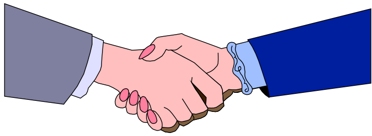
- Each party has its particular goals.



- Each party can formulate its protection goals.



- Security conflicts are recognized and compromises negotiated.



- Each party can enforce its protection goals within the agreed compromise. As far as limitations of this cannot be avoided, they equally apply to all parties.

Security with minimal assumptions about others

Protection Goals: Sorting

	Content	Circumstances
Prevent the unintended	Confidentiality Hiding	Anonymity Unobservability
Achieve the intended	Integrity	Accountability
	Availability	Reachability Legal Enforceability

Protection Goals: Definitions

Confidentiality ensures that nobody apart from the communicants can discover the content of the communication.

Hiding ensures the confidentiality of the transfer of confidential user data. This means that nobody apart from the communicants can discover the existence of confidential communication.

Anonymity ensures that a user can use a resource or service without disclosing his/her identity. Not even the communicants can discover the identity of each other.

Unobservability ensures that a user can use a resource or service without others being able to observe that the resource or service is being used. Parties not involved in the communication can observe neither the sending nor the receiving of messages.

Unlinkability ensures that an attacker cannot sufficiently distinguish whether two or more items of interest (subjects, messages, actions, …) are related or not.

Integrity ensures that modifications of communicated content (including the sender’s name, if one is provided) are detected by the recipient(s).

Accountability ensures that sender and recipients of information cannot successfully deny having sent or received the information. This means that communication takes place in a provable way.

Availability ensures that communicated messages are available when the user wants to use them.

Reachability ensures that a peer entity (user, machine, etc.) either can or cannot be contacted depending on user interests.

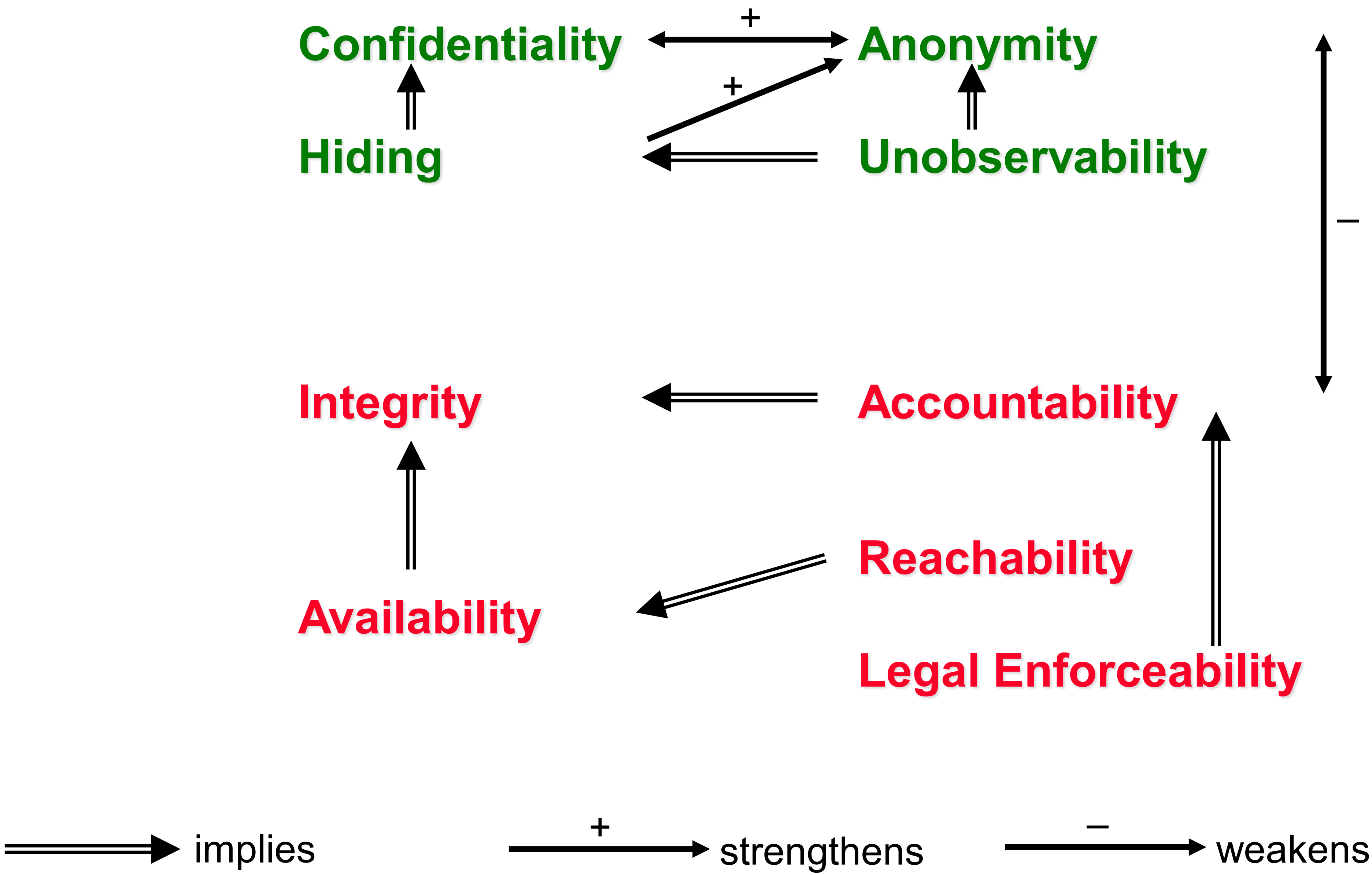
Legal enforceability ensures that a user can be held liable to fulfill his/her legal responsibilities within a reasonable period of time.

Additional Data Protection Goals: Definitions (Rost/Pfitzmann 2009)

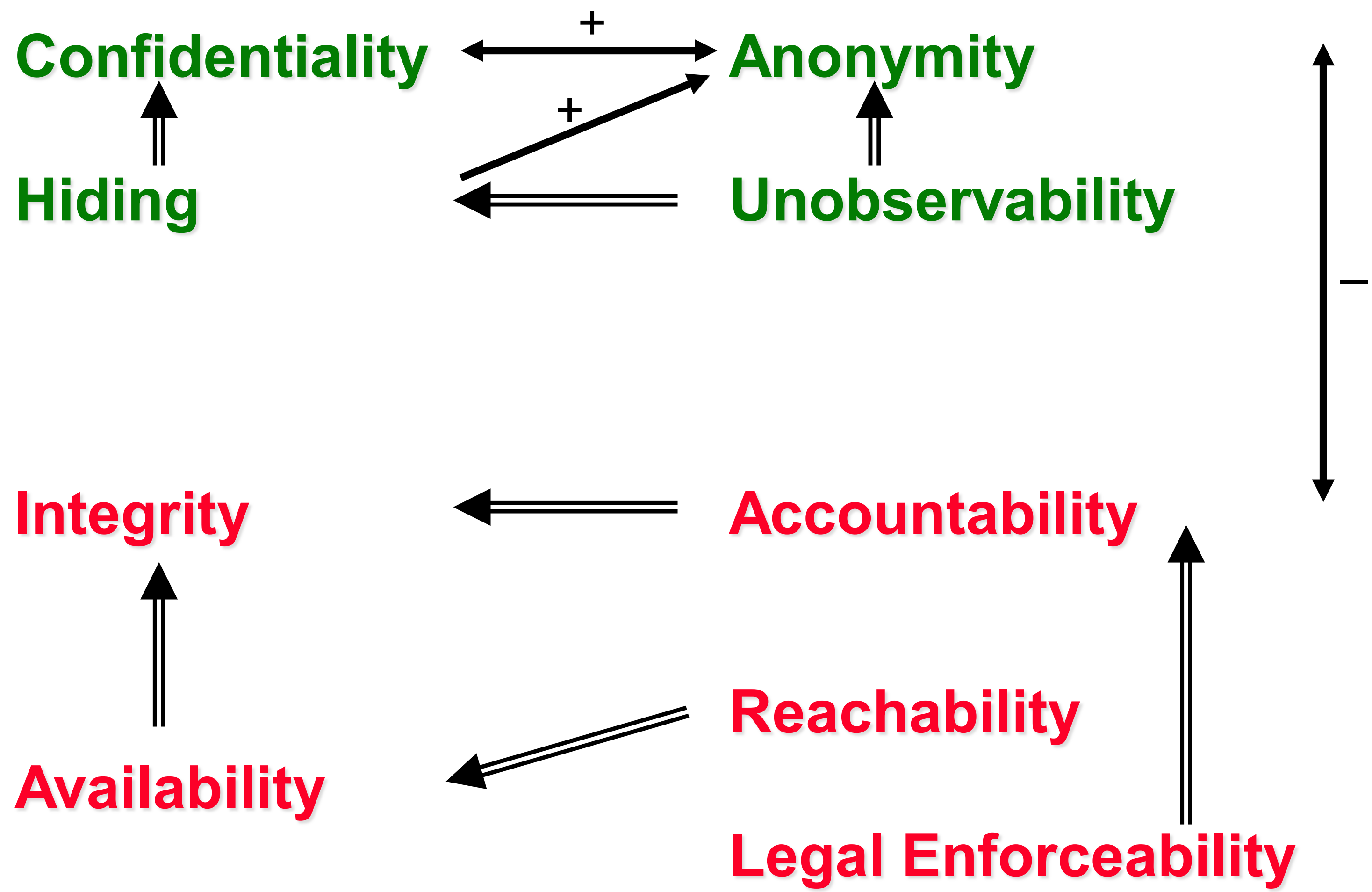
Transparency ensures that that the data collection and data processing operations can be planned, reproduced, checked and evaluated with reasonable efforts.

Intervenability ensures that the user is able to exercise his or her entitled rights within a reasonable period of time.

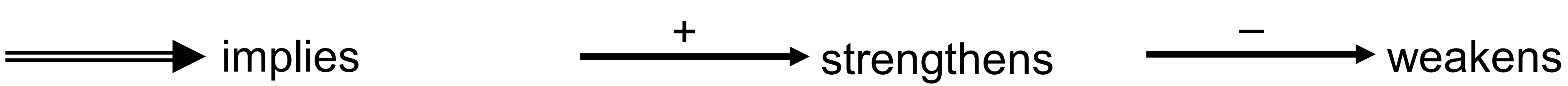
Correlations between protection goals



Correlations between protection goals



Transitive closure to be added



Physical security assumptions

Each technical security measure needs a physical "anchoring" in a part of the system which the attacker has neither read access nor modifying access to.

Range from "computer centre X" to "smart card Y"

What can be expected at best ?

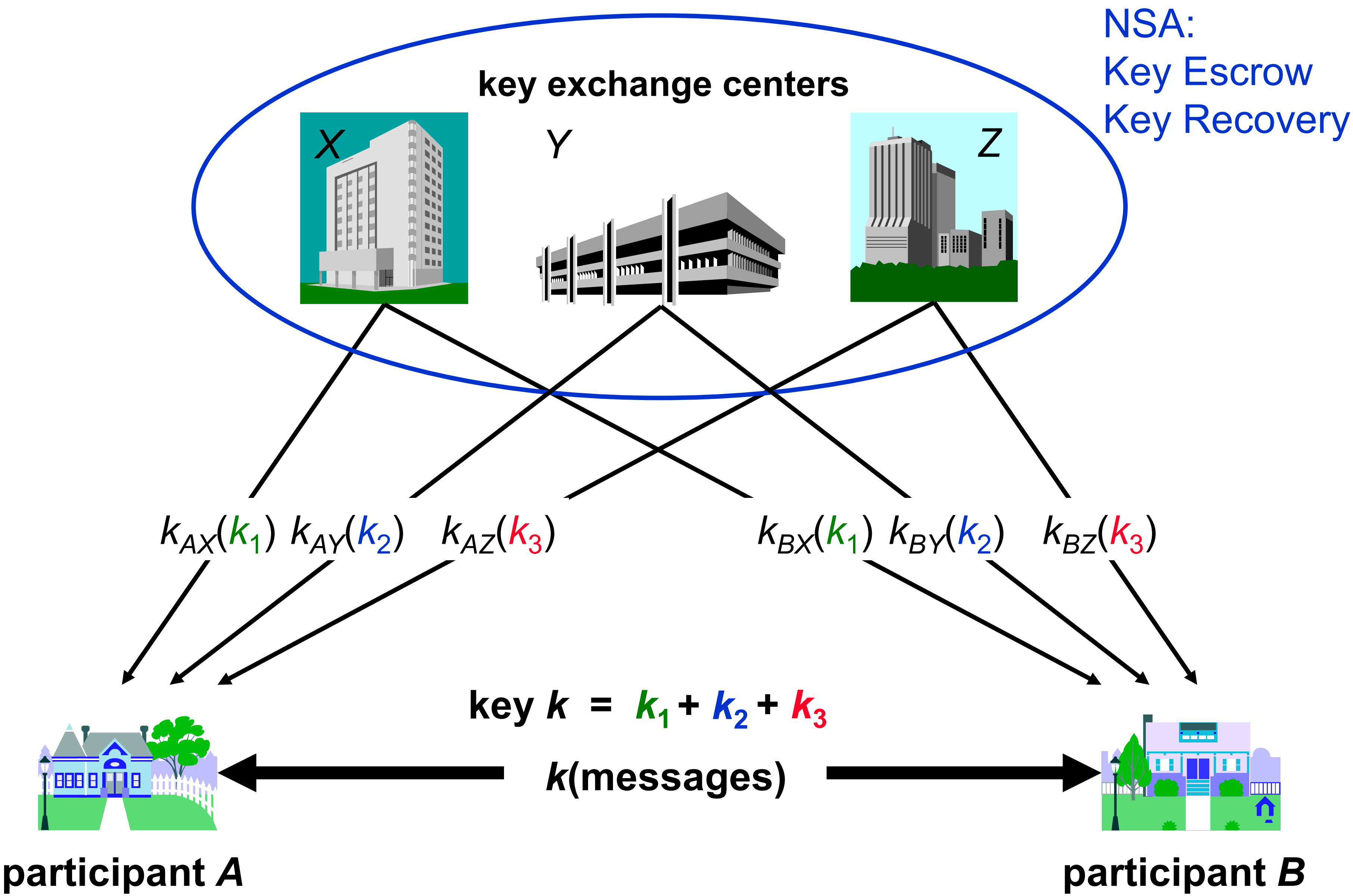
Availability of a locally concentrated part of the system cannot be provided against *realistic* attackers

→ **physically distributed system**

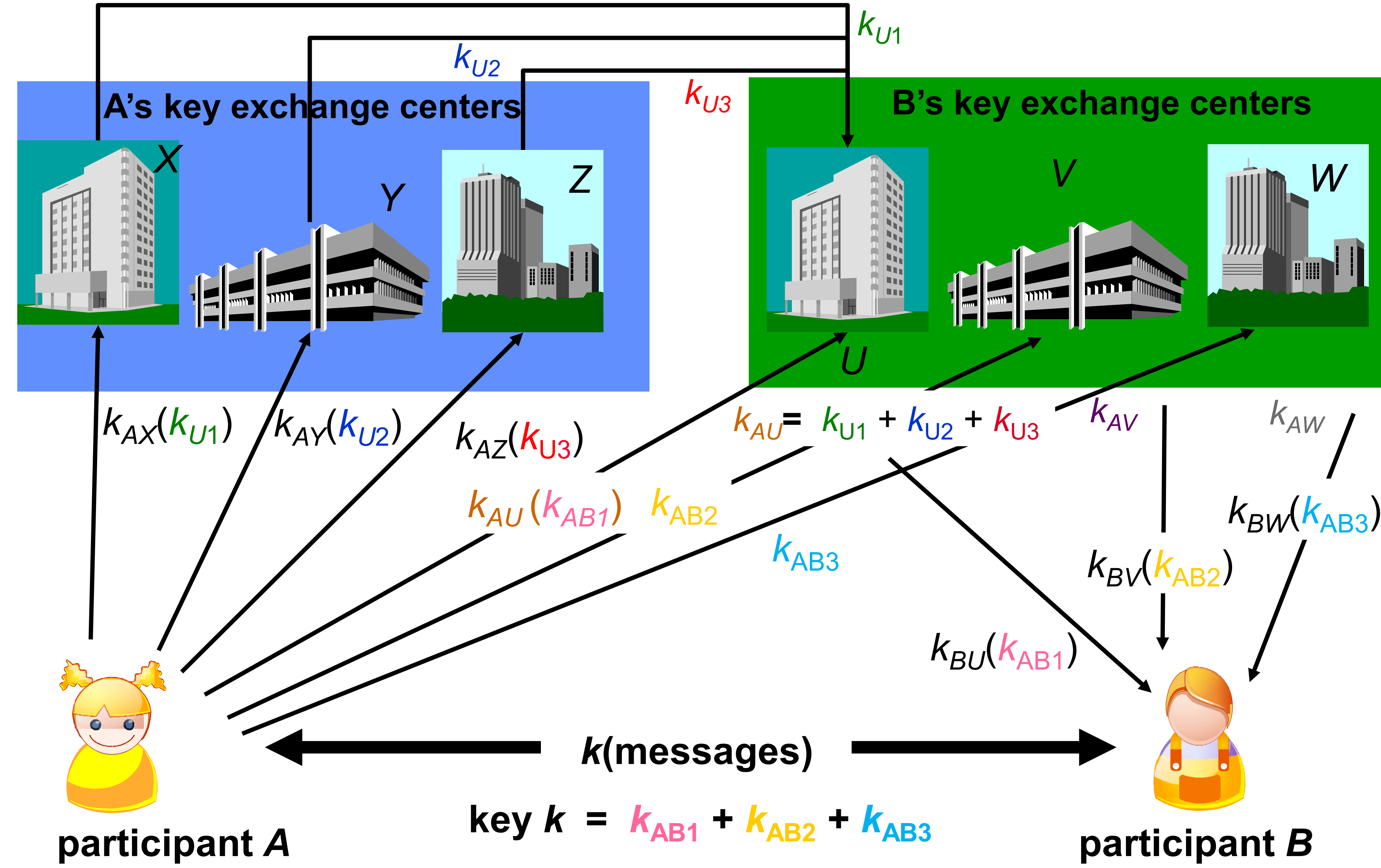
… hope the attacker cannot be at many places at the same time.

Distribution makes **confidentiality** and **integrity** more difficult. But physical measures concerning confidentiality and integrity are more efficient: Protection against *all realistic* attackers seems feasible. If so, physical distribution is quite ok.

Key exchange using symmetric encryption systems



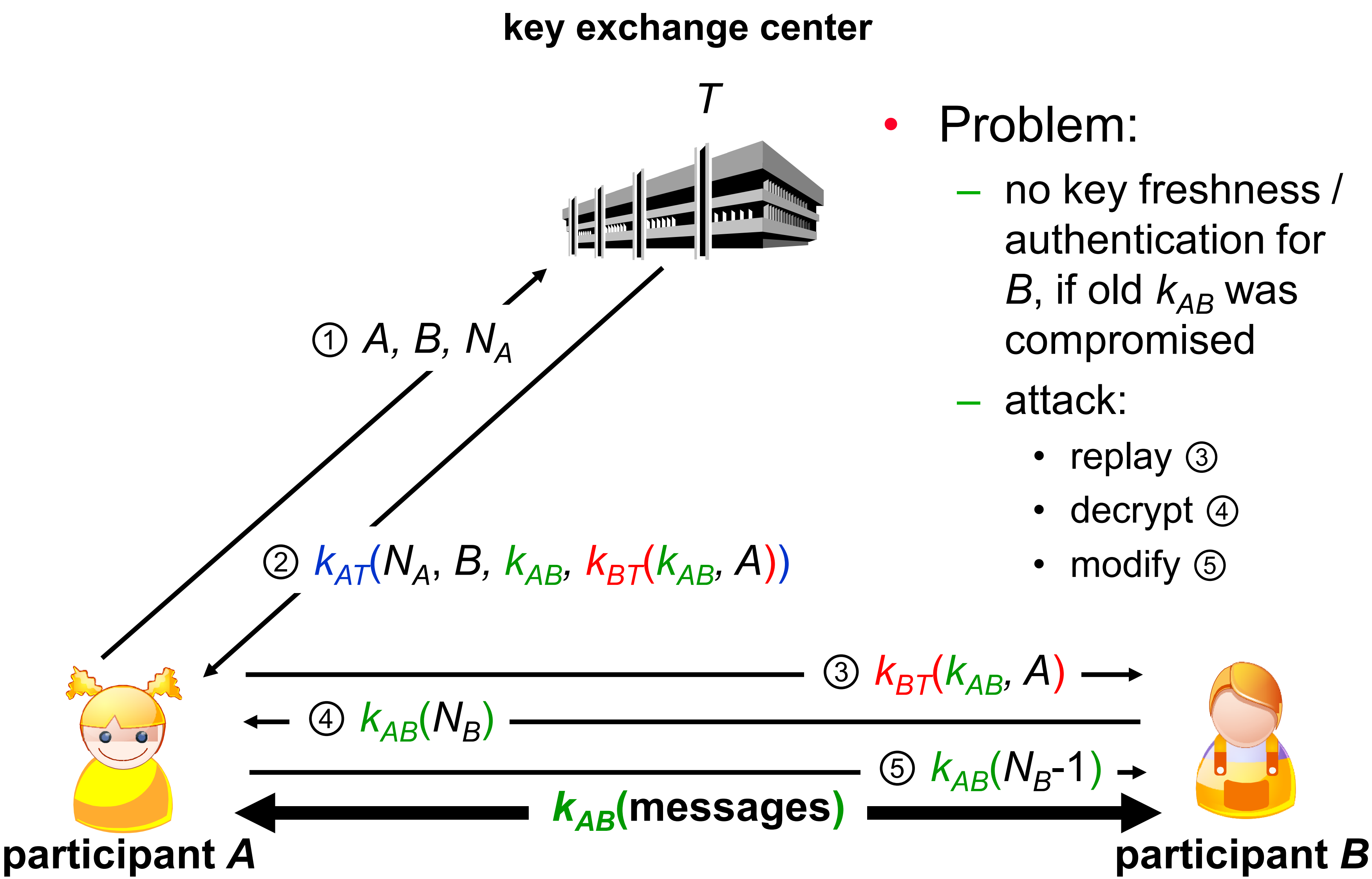
Key exchange using symmetric encryption systems



Needham-Schroeder-Protocol using Symmetric encryption

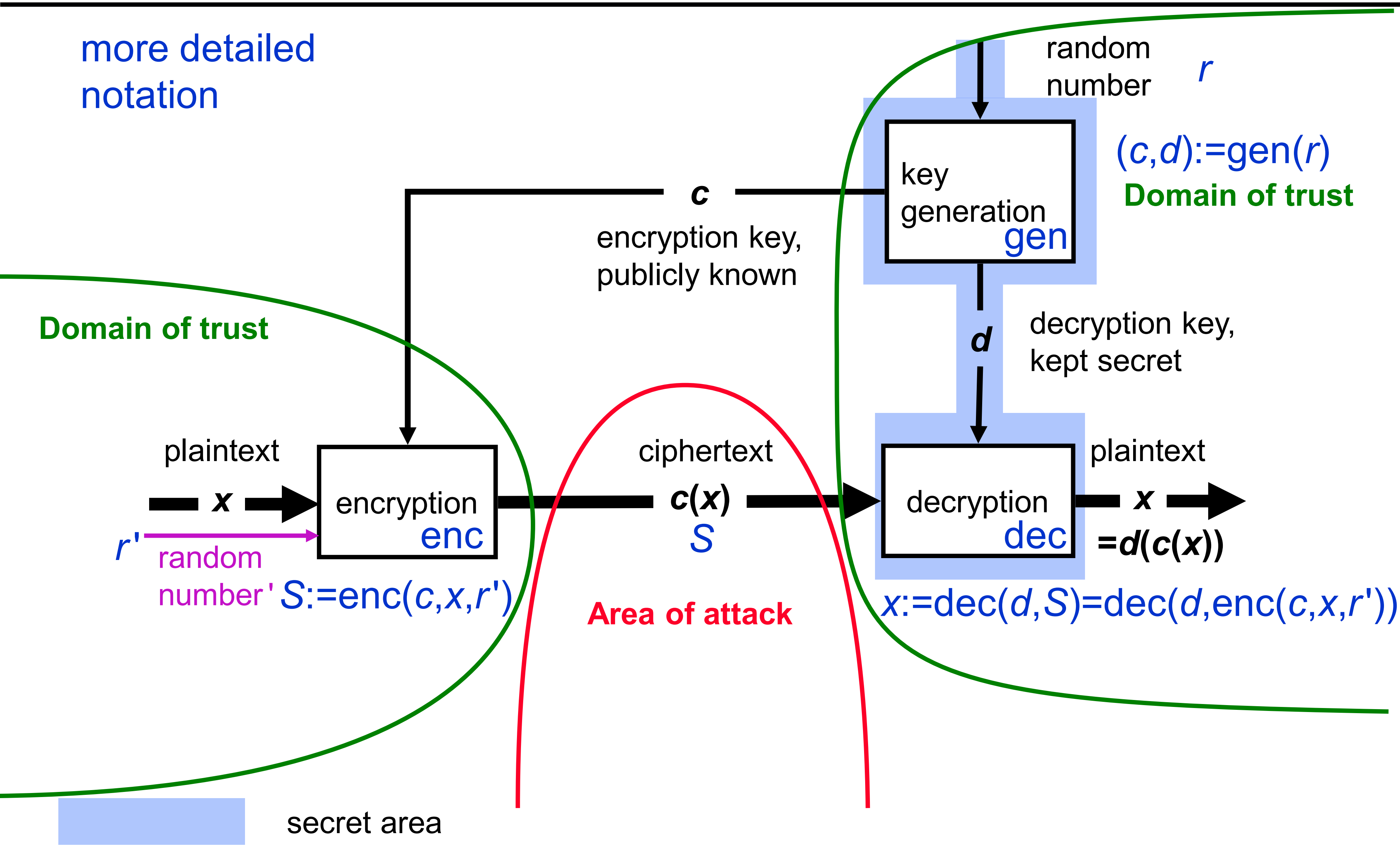
- from 1978
- goals:
  - key freshness:
    - key is „fresh“, i.e. a newly generated one
  - key authentication:
    - key is only known to Alice and Bob (and maybe some trusted third party)
- preconditions:
  - a trusted third party *T*
  - shared term secret keys between Alice (resp. Bob) and the trusted third party:
    - k_{AT}, k_{BT}

Needham-Schroeder-Protocol using Symmetric encryption



- Problem:
  - no key freshness / authentication for B, if old k_{AB} was compromised
  - attack:
    - replay ③
    - decrypt ④
    - modify ⑤

Asymmetric encryption system

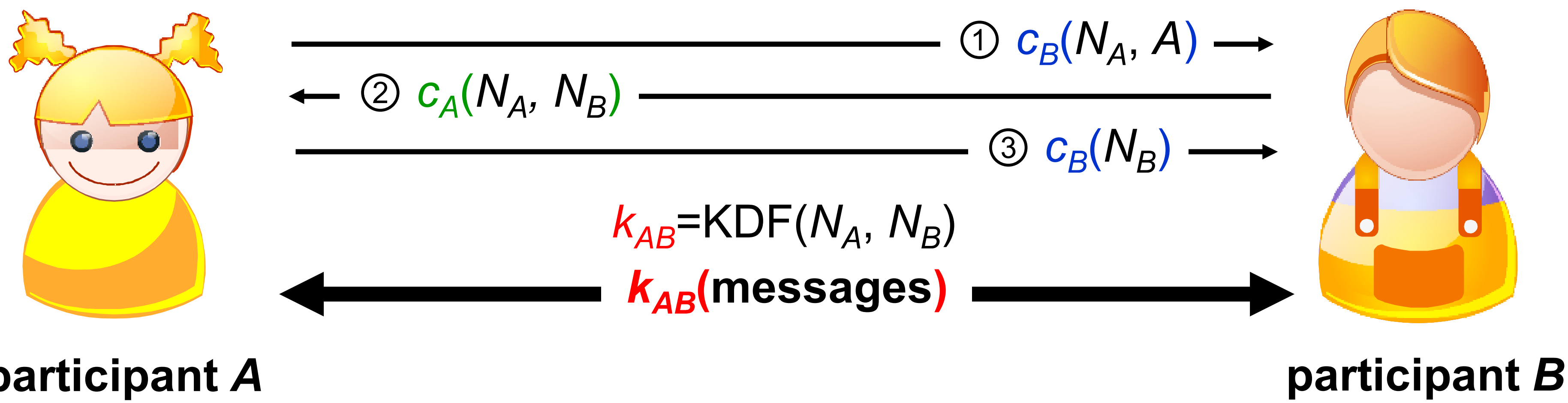


Opaque box with spring lock; 1 key

Needham-Schroeder-Protocol using Asymmetric encryption

- from 1978
- goals:
  - key freshness:
    - key is „fresh“, i.e. a newly generated one
  - key authentication:
    - key is only known to Alice and Bob
- preconditions:
  - public encryption keys of Alice c_A and Bob c_B known to each other

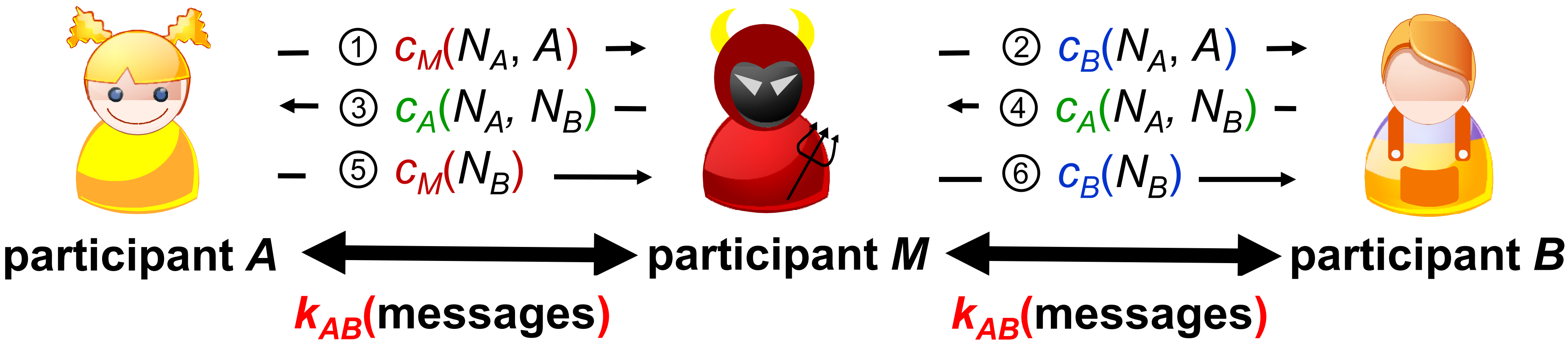
Needham-Schroeder-Protocol using Asymmetric encryption



- Problem:
  - *B* does not know if he really talks to *A*

Attack on asymmetric Needham-Schroeder-Protocol

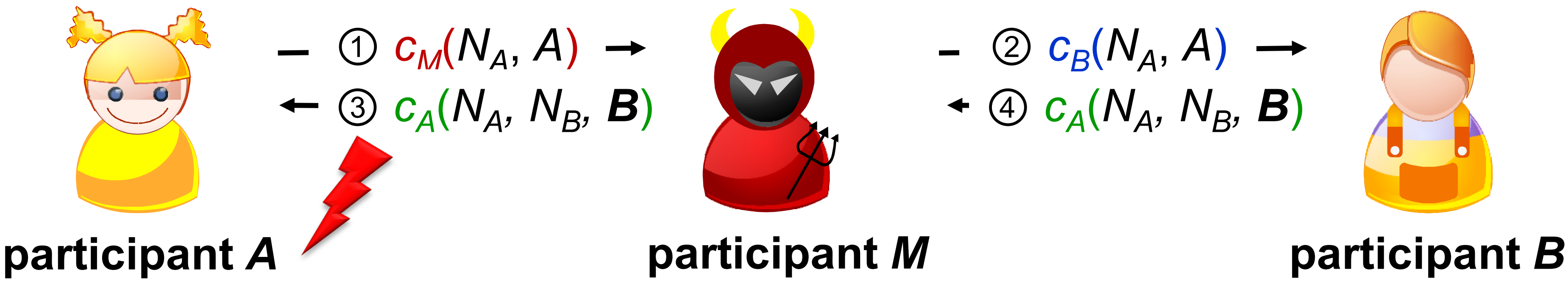
[Loewe 1996!]



k_{AB}=KDF(N_A, N_B)

- Solution:
  - *B* has to include his identity in his message ④

Attack on asymmetric Needham-Schroeder-Protocol



- Note:
  - *encryption* has to be non-mallable

One-Time-Pad mod 4

participant **A**

$c=11_2=3_{10}$ →

participant **B**

- Problem:
  - invert last bit of plain-text

$c=m+k \bmod 4$

$m=c-k \bmod 4$

possible Keys	Plain-text	manipulated Plain-text	manipulated Cipher-text
0	$3=11_2$	$10_2=2$	$2=10_2$
1	$2=10_2$	$11_2=3$	$0=00_2$
2	$1=01_2$	$00_2=0$	$2=10_2$
3	$0=00_2$	$01_2=1$	$0=00_2$

- Problem: $k=3$, $c=2$ → $m=3=11_2$

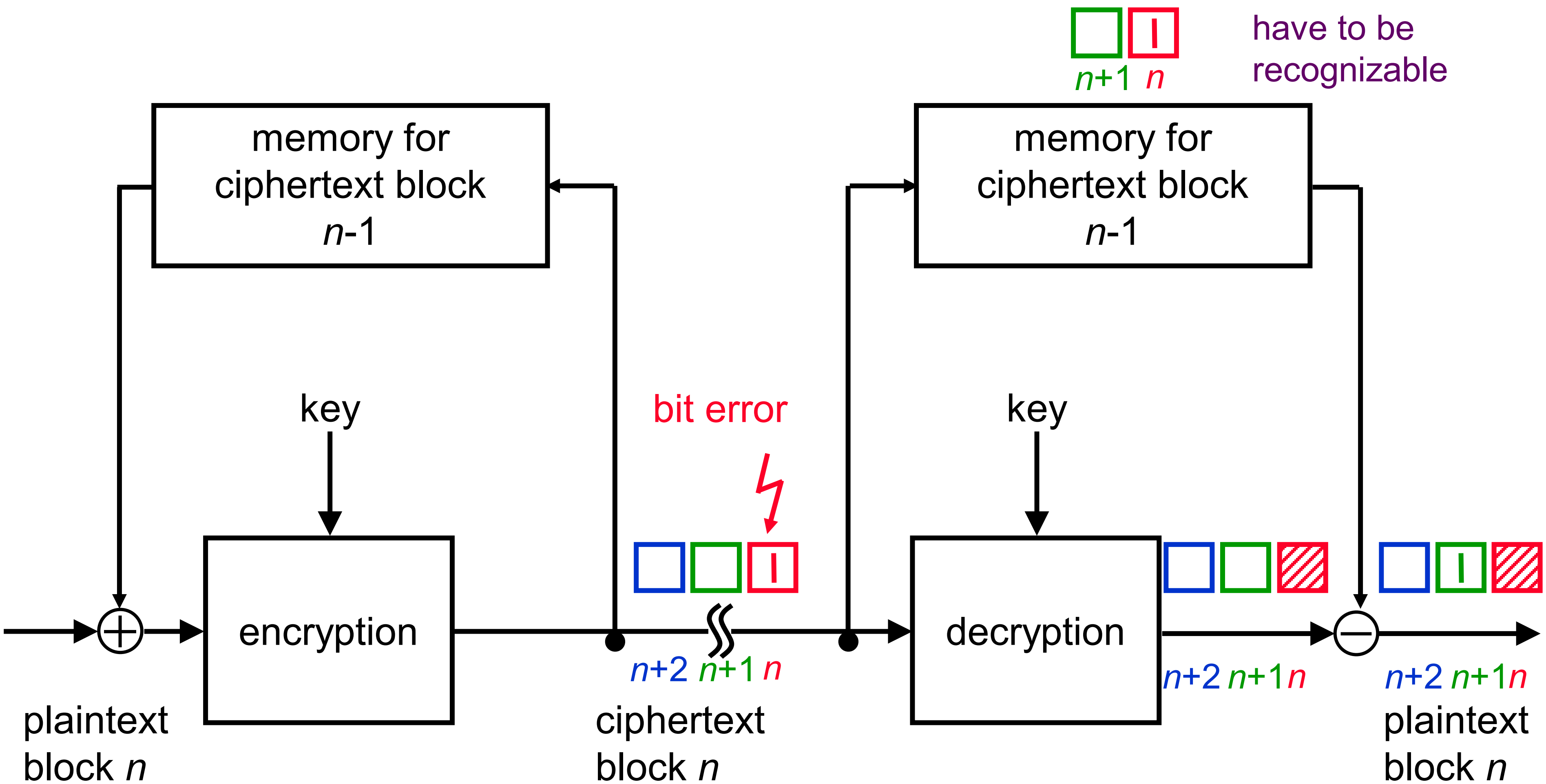
Cipher Block Chaining (CBC)

All lines transmit as many characters as a block comprises

$\oplus$ Addition mod appropriately chosen modulus

$\ominus$ Subtraction mod appropriately chosen modulus

If error on the line: Resynchronization after 2 blocks, but block borders have to be recognizable



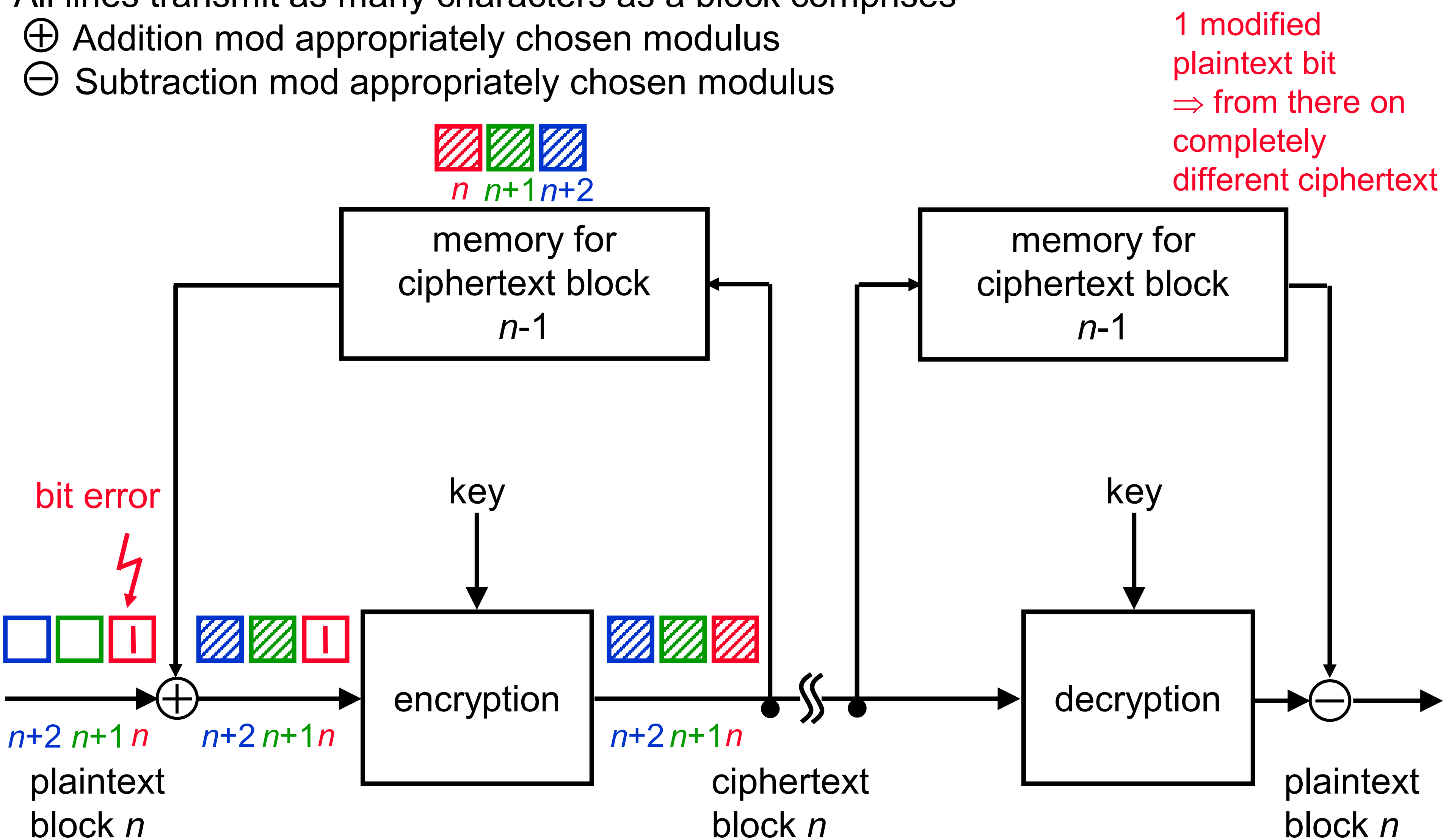
self synchronizing

Cipher Block Chaining (CBC) (2)

All lines transmit as many characters as a block comprises

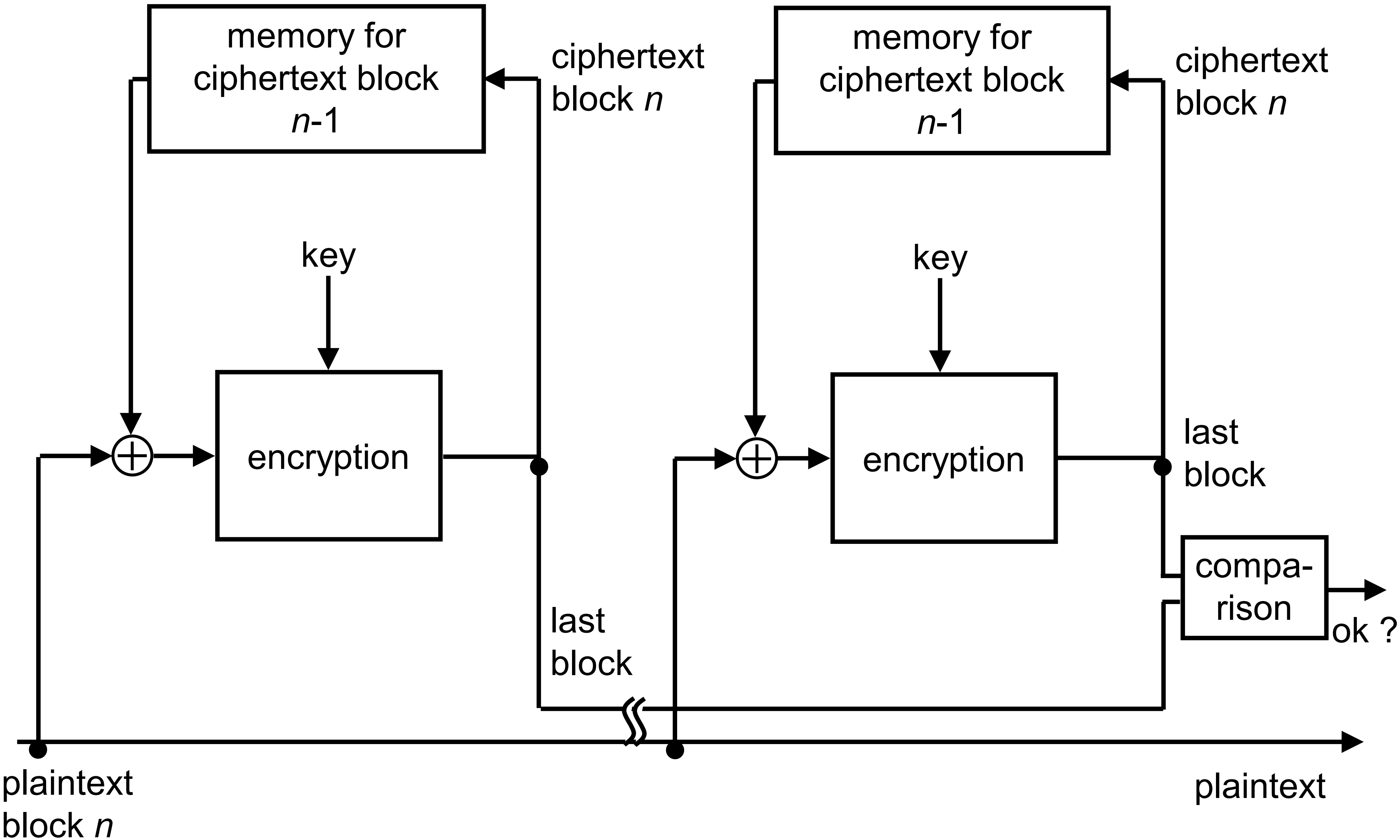
⊕ Addition mod appropriately chosen modulus

⊖ Subtraction mod appropriately chosen modulus

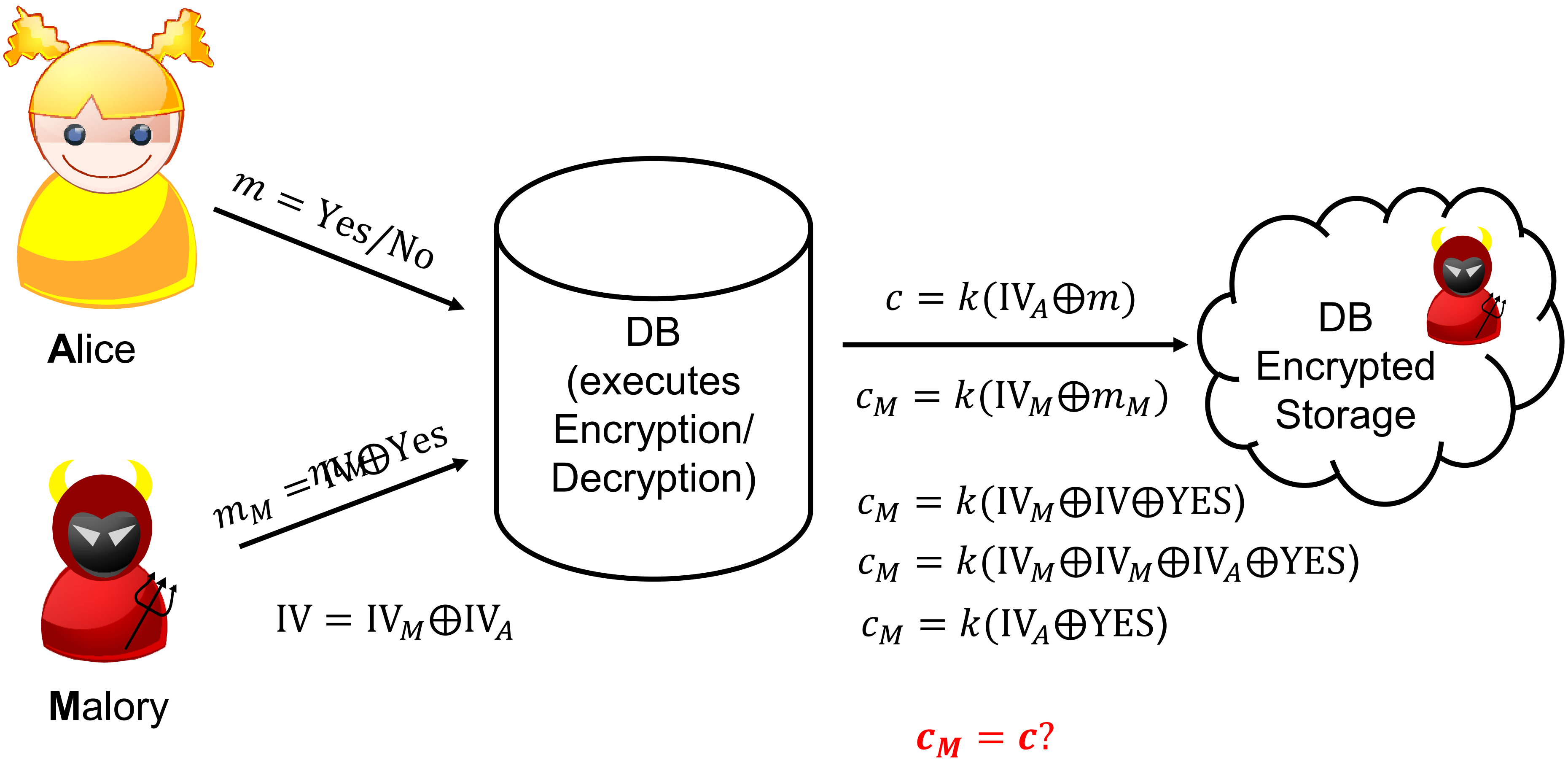


useable for authentication ⇒ use last block as MAC

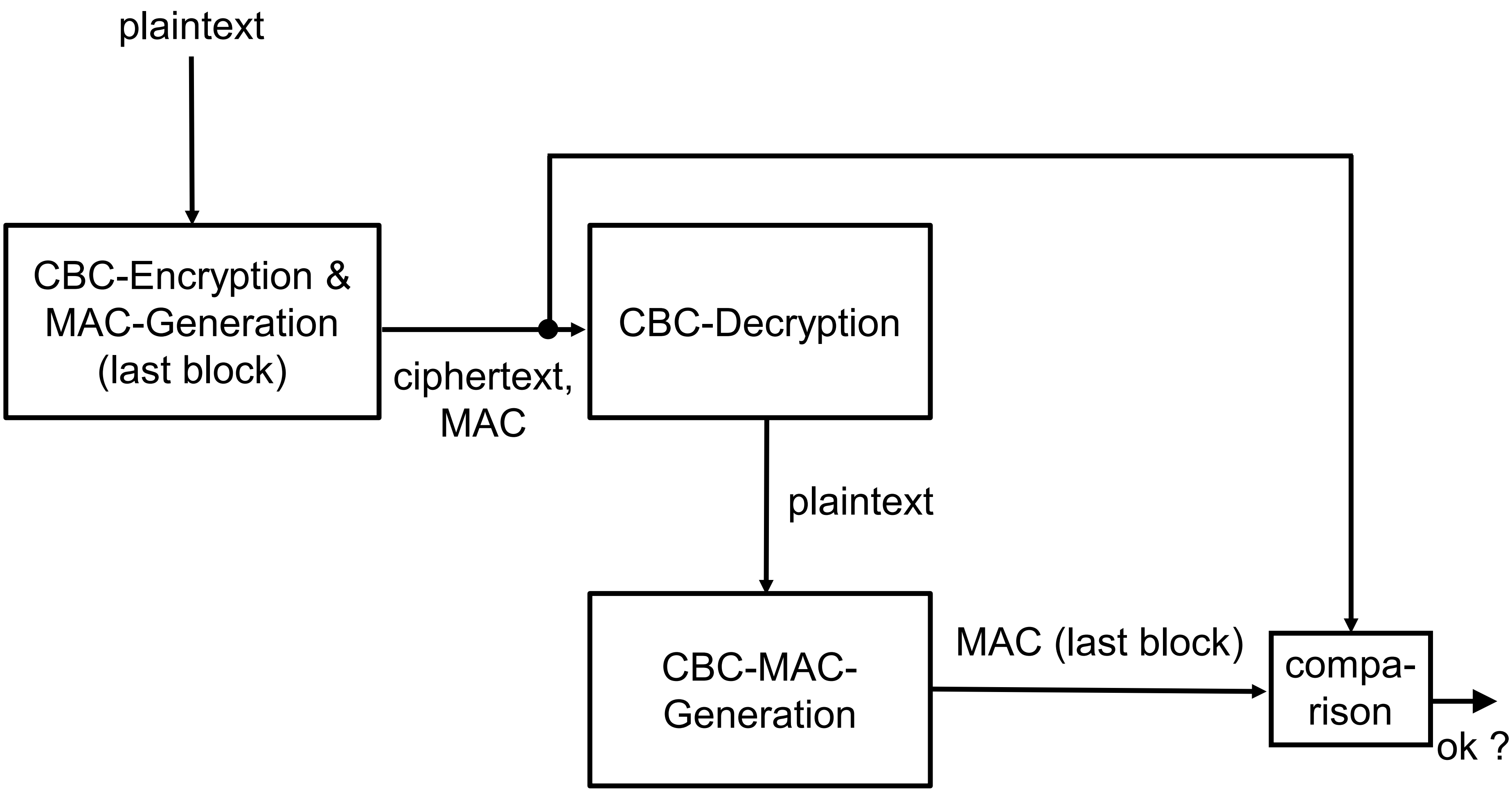
CBC for authentication



Why CBC IV should be random?



CBC for Confidentiality & Integrity



Whole Disk Encryption – Requirements

- The data on the disk should remain confidential
- Manipulations on the data should be detectable
- Data retrieval and storage should both be fast operations, no matter where on the disk the data is stored.
- The encryption method should not waste disk space (i.e., the amount of storage used for encrypted data should not be significantly larger than the size of plaintext)

- Attacker model:
  - they can read the raw contents of the disk at any time
  - they can request the disk to encrypt and store arbitrary files of their choosing
  - and they can modify unused sectors on the disk and then request their decryption

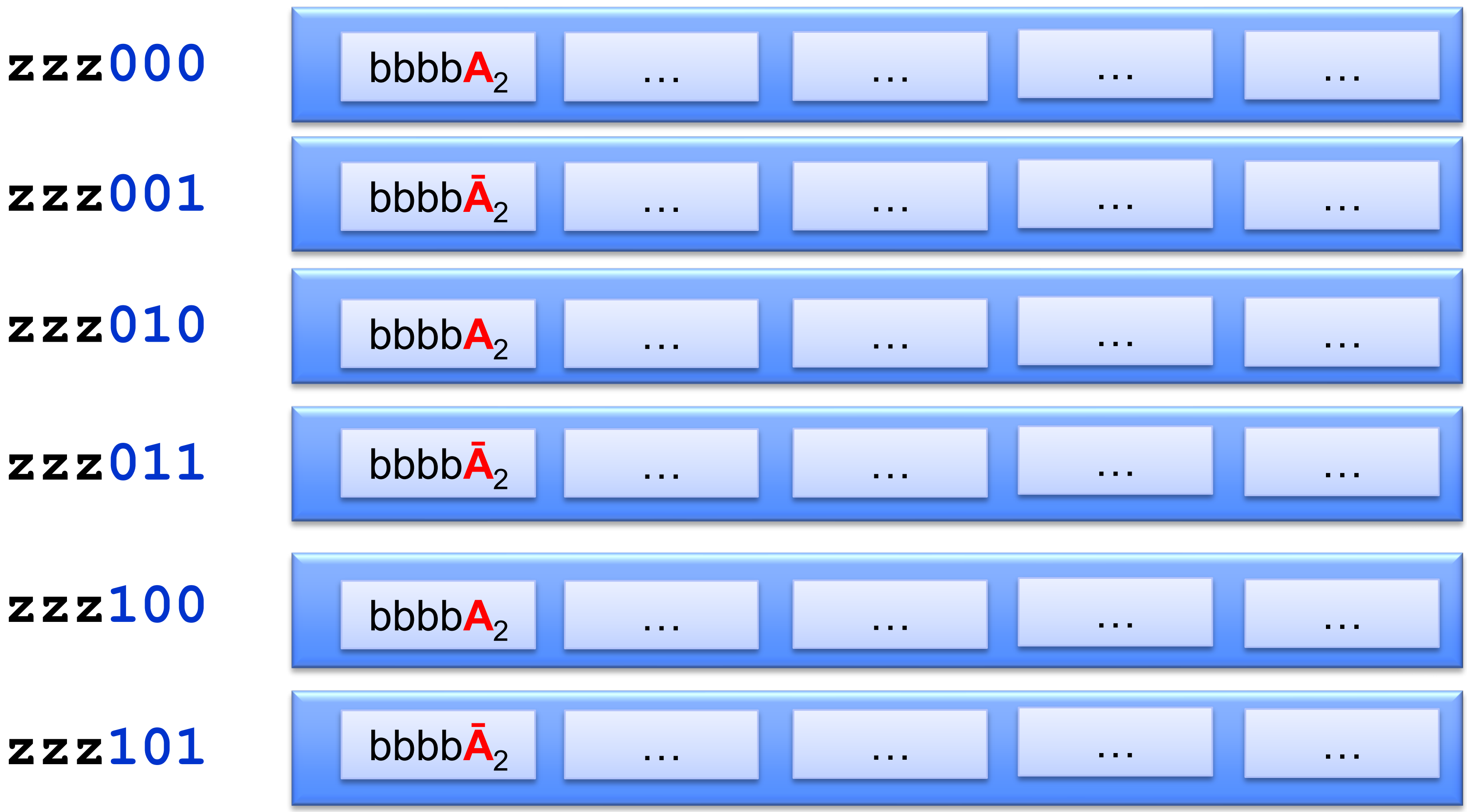
Watermarking Attack on Whole Disk Encryption

- Goal: Detect stored files
- Assumptions regarding Attacker:
  - they can read the raw contents of the disk at any time
  - they can request the disk to encrypt and store arbitrary files of their choosing
- Assumptions regarding Encryption & Storage:
  - CBC (IV, k, m) → CBC (sector number, k, m)
  - Remember first block CBC: Enc(k, m[0] $\oplus$ IV)
  - (parts of) larger files a stored at consecutive sectors
    - SN_x, SN_{x+1}, SN_{x+2},…,SN_{x+y}
    - where will be an *x* where the *t* least significant bits are all 0 and $y \geq 2^t$
      - t=3 → SN_x: `zzzzzz000, …,` SN_{x+7}: `zzzzzz111`
- Attack:
  - Create plaintext such that the first plaintext-blocks stored in each sector differ only in the LSB

Watermarking Attack on Whole Disk Encryption

Sector Number (IV)

Plaintext blocks stored in the sectors



Watermarking Attack on Whole Disk Encryption

Sector Number (IV)	First Plaintext block stored in the sectors	First ciphertext block stored in the sectors
zzz000	bbbbA_2	$\text{Enc}(\text{bbbb}A_2 \oplus \text{zzz000}) = \text{Enc}(p_1) = c_1$
zzz001	bbbb$\bar{A}_2$	$\text{Enc}(\text{bbbb}\bar{A}_2 \oplus \text{zzz001}) = \text{Enc}(p_1) = c_1$
zzz010	bbbbA_2	$\text{Enc}(\text{bbbb}A_2 \oplus \text{zzz010}) = \text{Enc}(p_2) = c_2$
zzz011	bbbb$\bar{A}_2$	$\text{Enc}(\text{bbbb}\bar{A}_2 \oplus \text{zzz011}) = \text{Enc}(p_2) = c_2$
zzz100	bbbbA_2	$\text{Enc}(\text{bbbb}A_2 \oplus \text{zzz100}) = \text{Enc}(p_3) = c_3$
zzz101	bbbb$\bar{A}_2$	$\text{Enc}(\text{bbbb}\bar{A}_2 \oplus \text{zzz101}) = \text{Enc}(p_3) = c_3$

Watermarking Attack on Whole Disk Encryption

Sector Number (IV)	First Plaintext block stored in the sectors	First ciphertext block stored in the sectors
zzz000	bbbbA_2	c_1
zzz001	bbbb$\bar{A}_2$	c_1
zzz010	bbbbA_2	c_2
zzz011	bbbb$\bar{A}_2$	c_2
zzz100	bbbbA_2	c_3
zzz101	bbbb$\bar{A}_2$	c_3

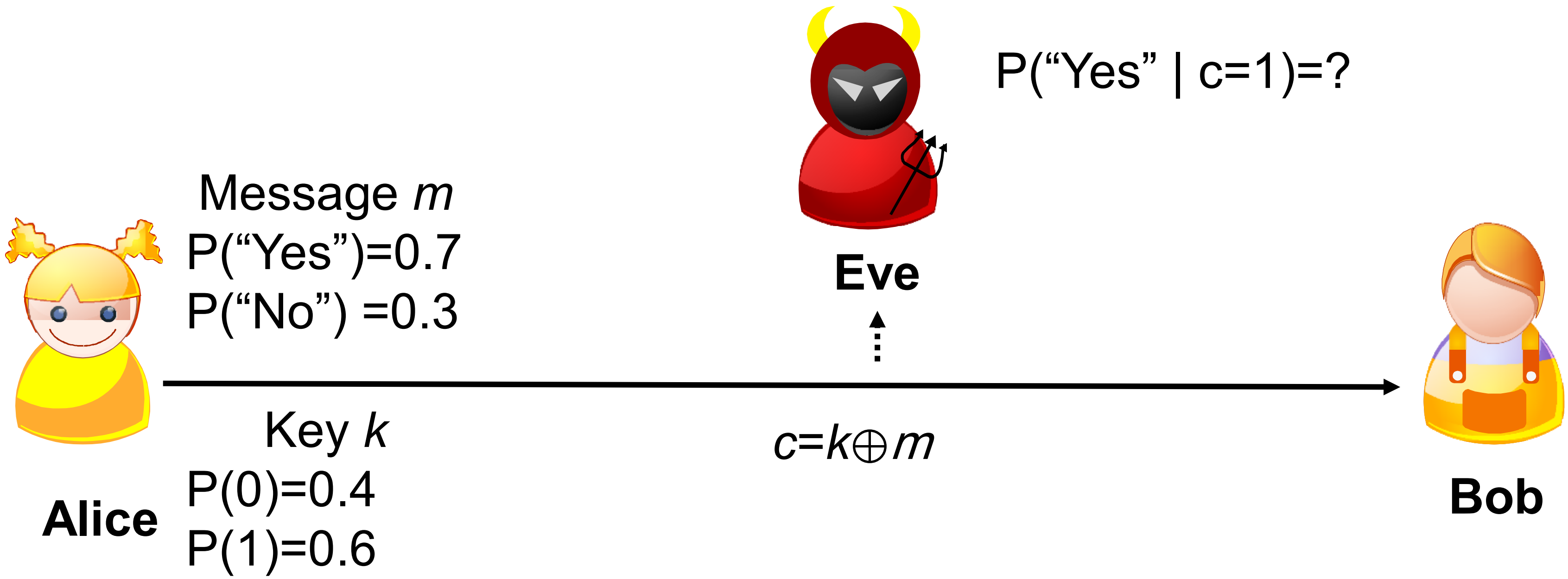
➔ Watermark!

Watermarking Attack on Whole Disk Encryption

- Solution: unpredictable IVs
- Construction:
  - Encrypted salt-sector initialization vector (ESSIV)
  - IV(SN) = Enc (Hash(k), SN)

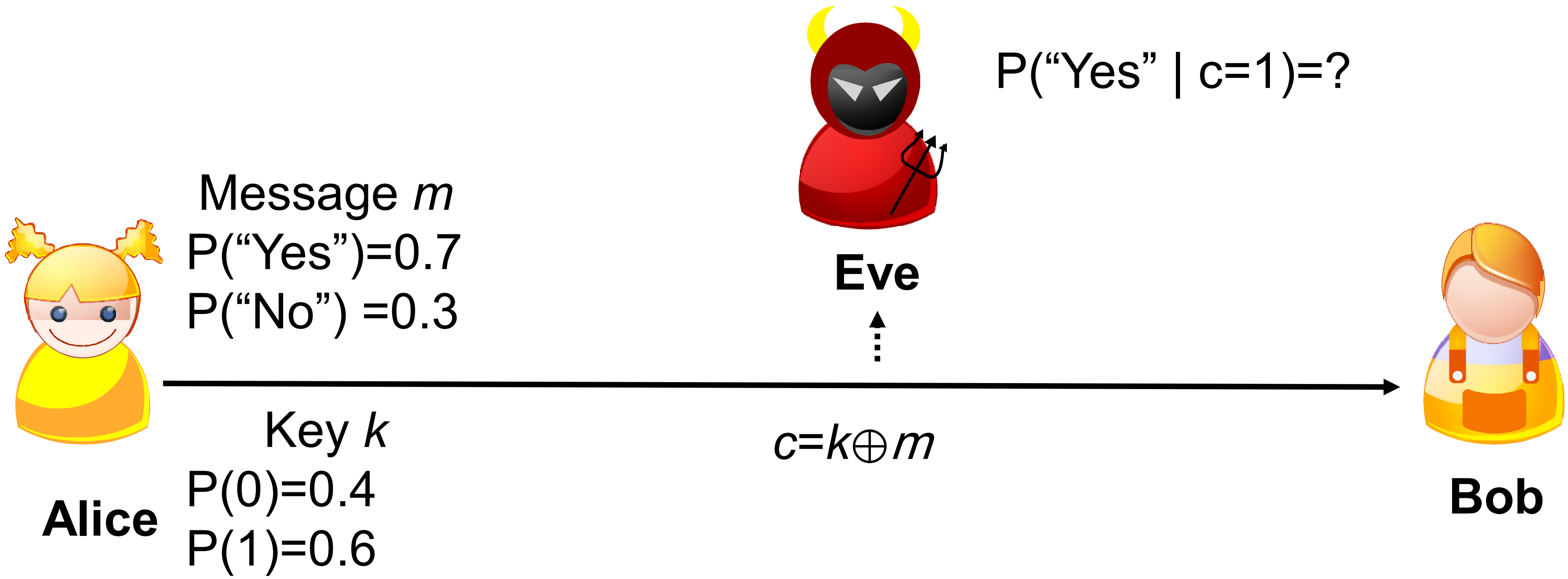
Probability Exercise

P("Yes" | c=1)=?



	k=0	k=1
m=„Yes“		
m=„No“		

Probability Exercise

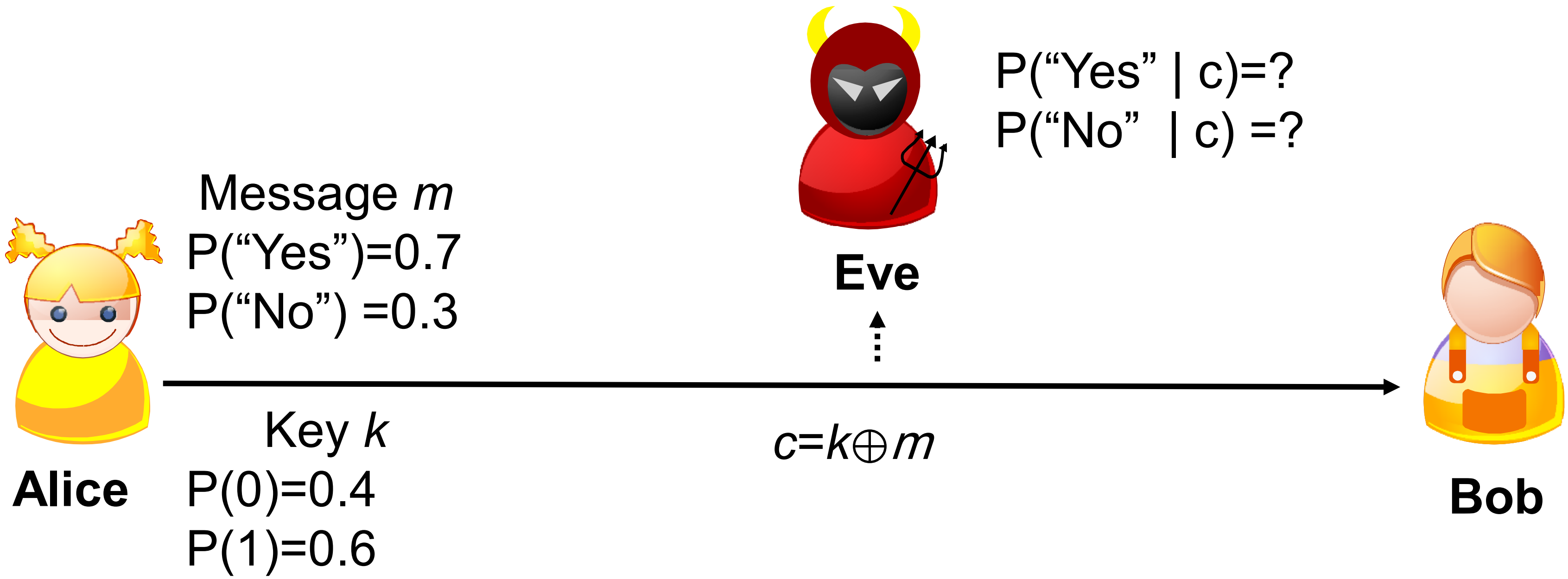


	k=0	k=1
m=„Yes“	c=1 P=0.28	c=0 P=0.42
m=„No“	c=0 P=0.12	c=1 P=0.18

$\sum P=1.0$

P("Yes" | c=1) = 0.28 / (0.28+0.18) = 0.28 / 0.46 ≈ 0.61

Probability Exercise



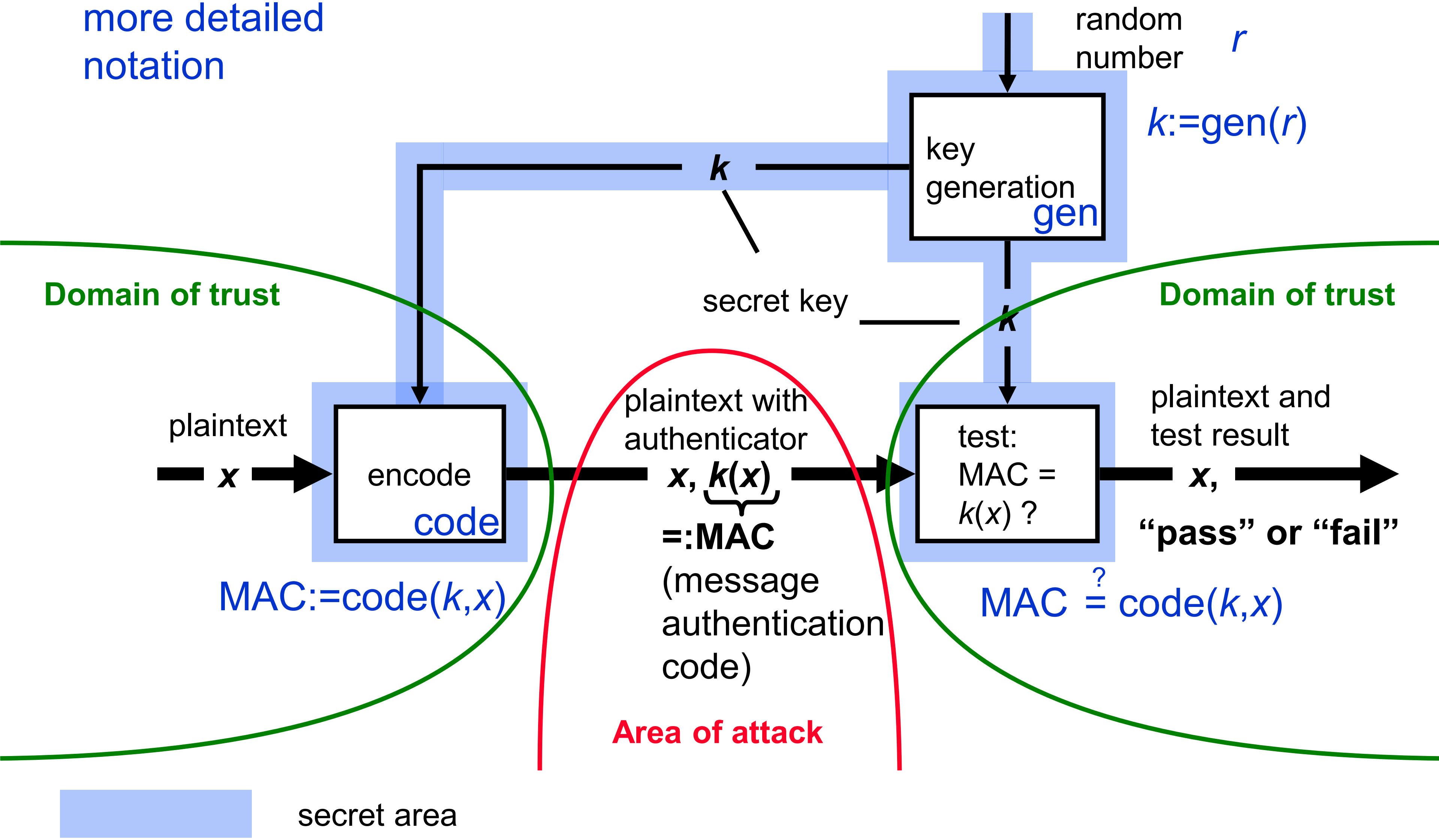
- Remember: $P(A_2|A_1) = \frac{P(A_1 \cap A_2)}{P(A_1)}$
- $P(m|c) = \frac{P(c \cap m)}{P(c)}$
- P(c=0) = P("Yes")·P(k=1) + P("No")·P(k=0) = 0.54
- P(c=0 and m="Yes") = P("yes")·P(k=1) = 0.42
- P("Yes" | c=0) ≈ 0.77 P("Yes" | c=1) ≈ 0.61

One-way functions – cryptographic hash functions

- One-way function *f*:
  - calculating $f(x)=y$ is easy
  - calculating $f^{-1}(y)=x$ is hard
    - computation / storage
  - open question: Do one-way functions exist?
- Cryptographic hash function *h*
  - might have different properties depending on the use case
  - collision resistance:
    - it is hard to find *x*, *y* with $h(y)=h(x)$ and $y \neq x$
    - note: *h* is usually not *collision free*, because $|h(x)| \ll |x|$
  - preimage resistance / one-way function / secrecy
    - given $h(x)$ it is hard to find *x*
  - second-preimage resistance / weak collision resistance / binding
    - given x, $h(x)$ it is hard to find y with $h(y)=h(x)$ and $y \neq x$
  - Note:
    - *h* is not necessarily a “random extractor”
    - only one of “secrecy” and “binding” can be information theoretic secure

Symmetric authentication system

more detailed notation



Show-case with lock; 2 identical keys

Calculating with and without *p*,*q* (2)

Z_n^* : multiplicative group

$a \in Z_n^* \Leftrightarrow \gcd(a,n) = 1$

- Inverting is fast (extended Euclidean Algorithm)

  Determine to *a*,*n* the values *u*,*v* with

  $$a \bullet u + n \bullet v = 1$$

  Then: $u \equiv a^{-1} \bmod n$

  example: $3^{-1} \bmod 11$?

$11 = 3 \bullet 3 + 2$

$3 = 1 \bullet 2 + 1$

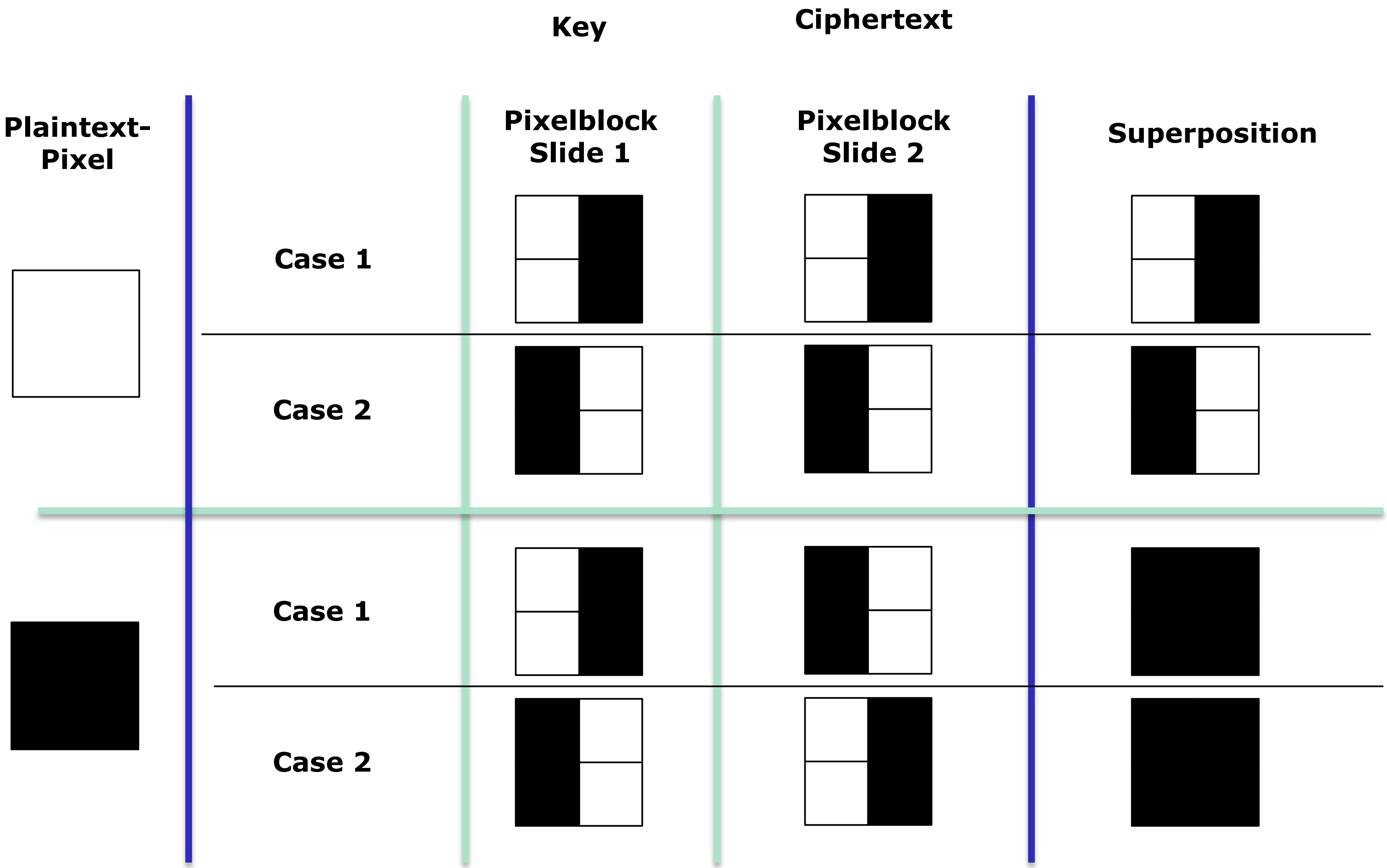
$1 = 1 \bullet 3 - 1 \bullet 2$

$= 1 \bullet 3 - 1 \bullet (11 - 3 \bullet 3)$

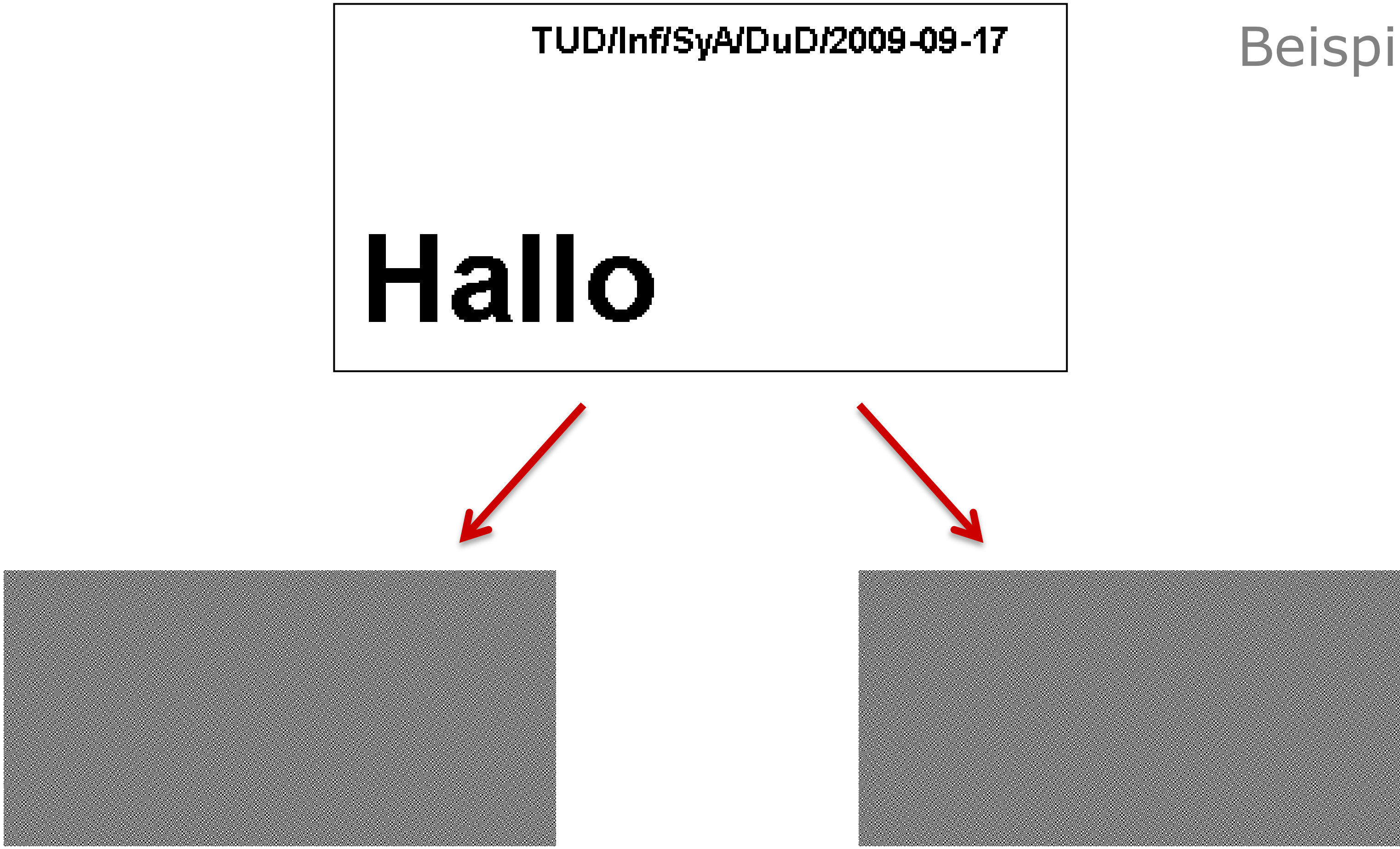
$= -11 + 4 \bullet 3$

$\Rightarrow 3^{-1} \equiv 4 \bmod 11$

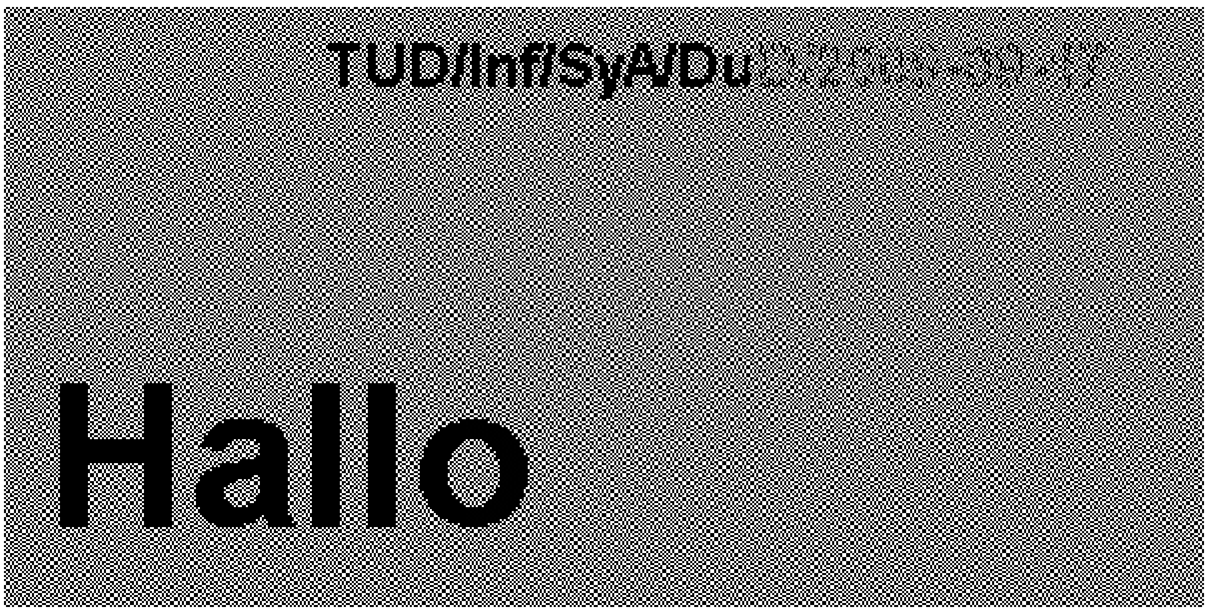
Visual Cryptography Scheme by Naor and Shamir (simplified)



Beispiel



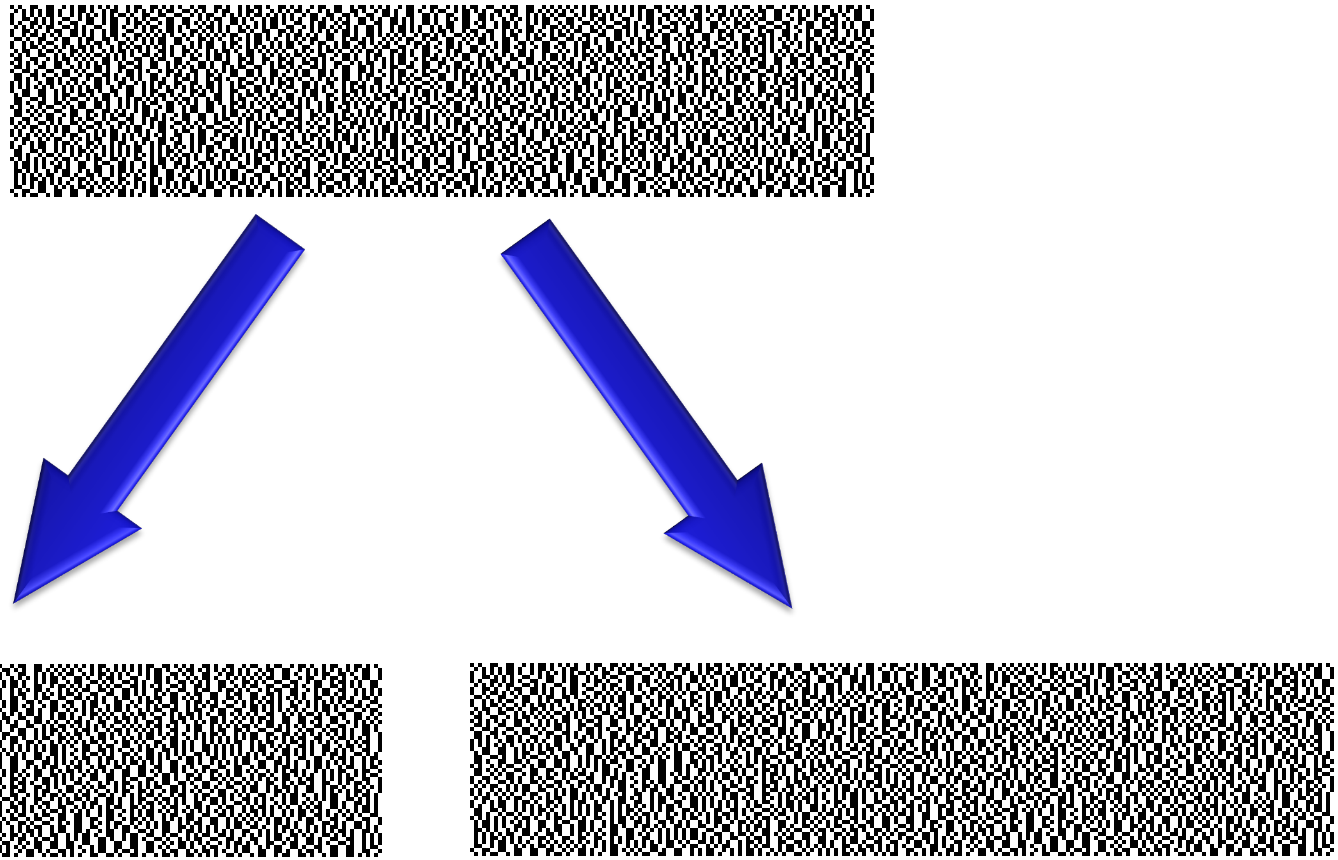
Example



ausprobieren: http://www-sec.uni-regensburg.de/vc/

Plausible Deniability

cipher
text

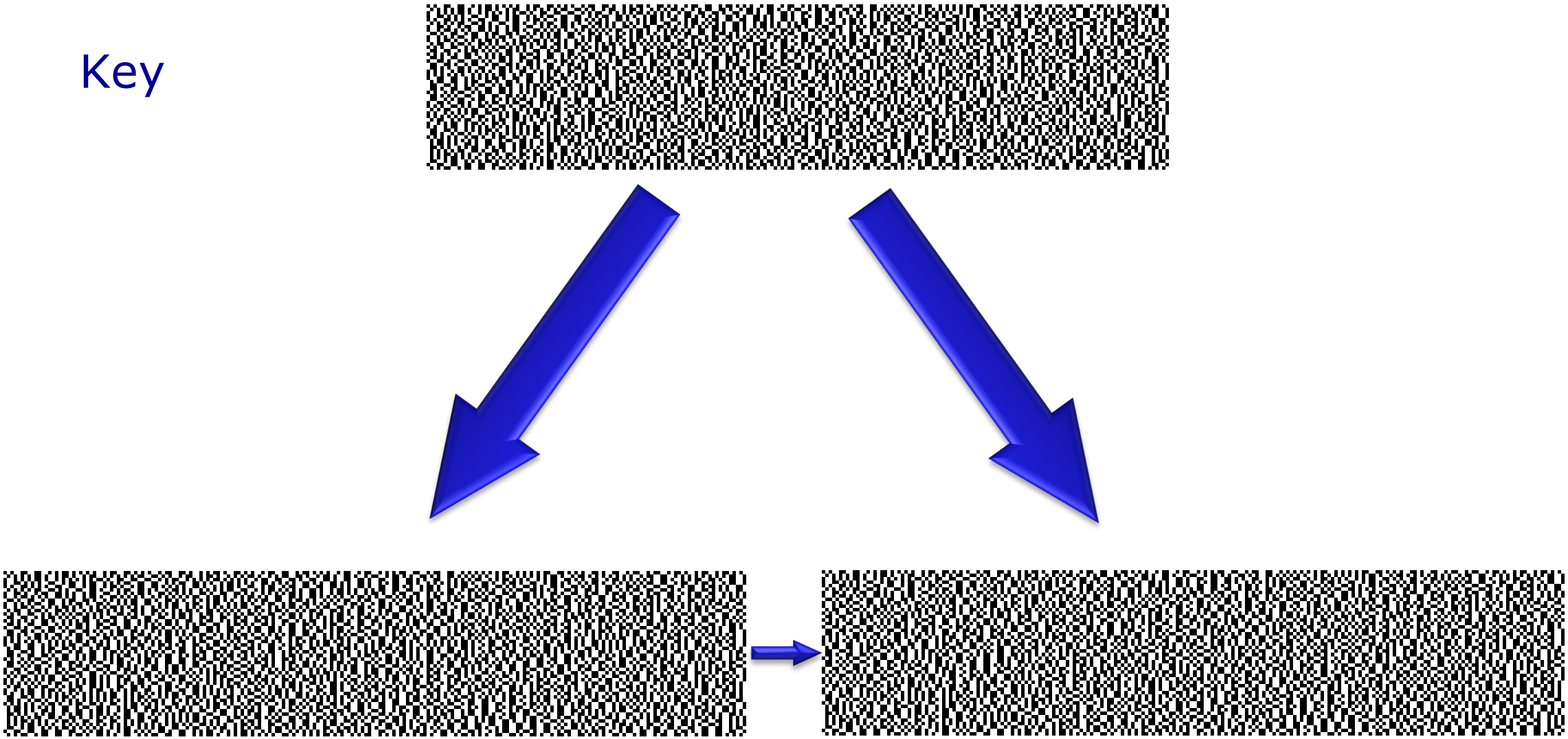


Key 1

Key 2

Do not reuse keys!

Key

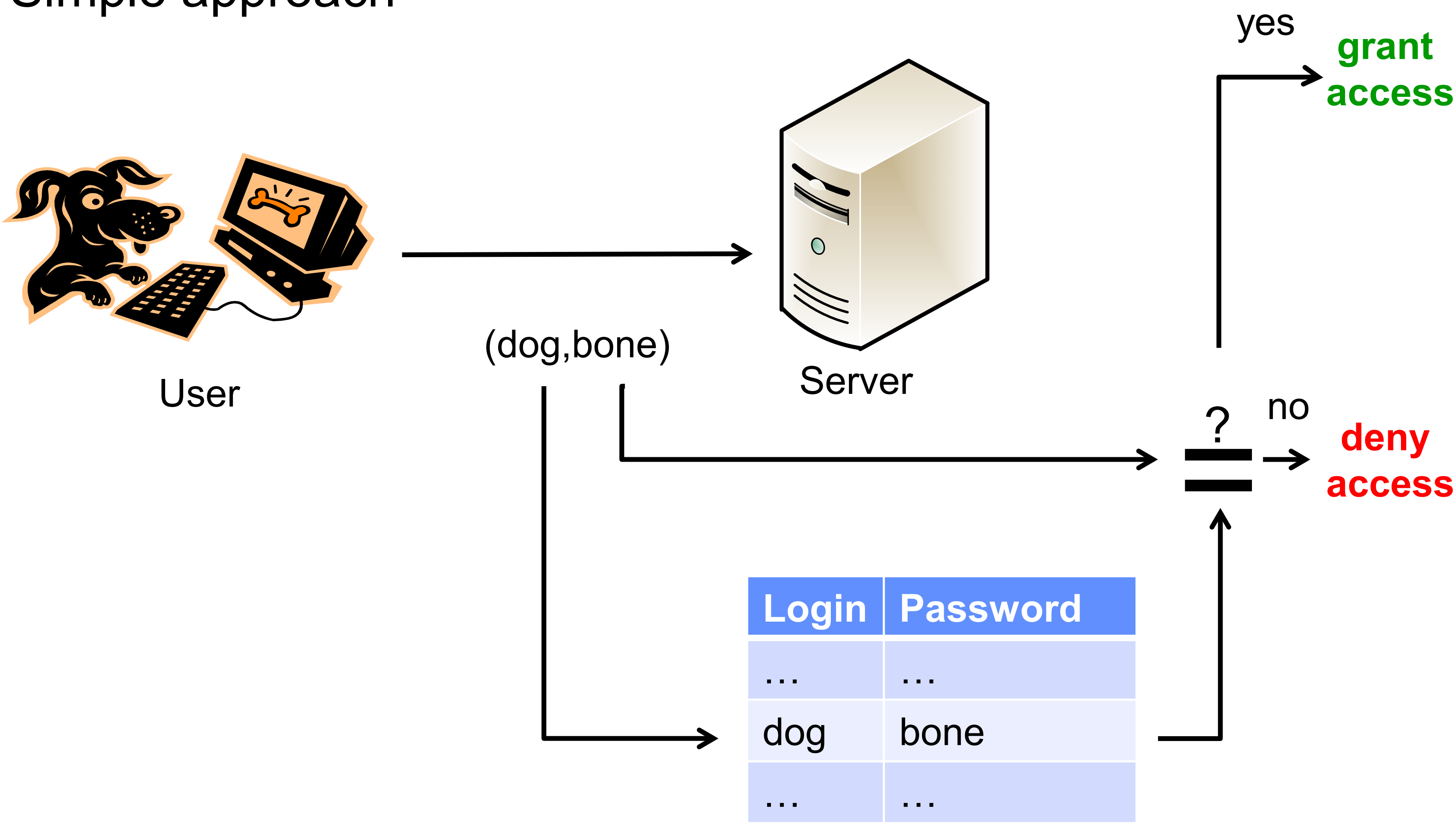


Ciphertext 1

Ciphertext 2

Password based authentication

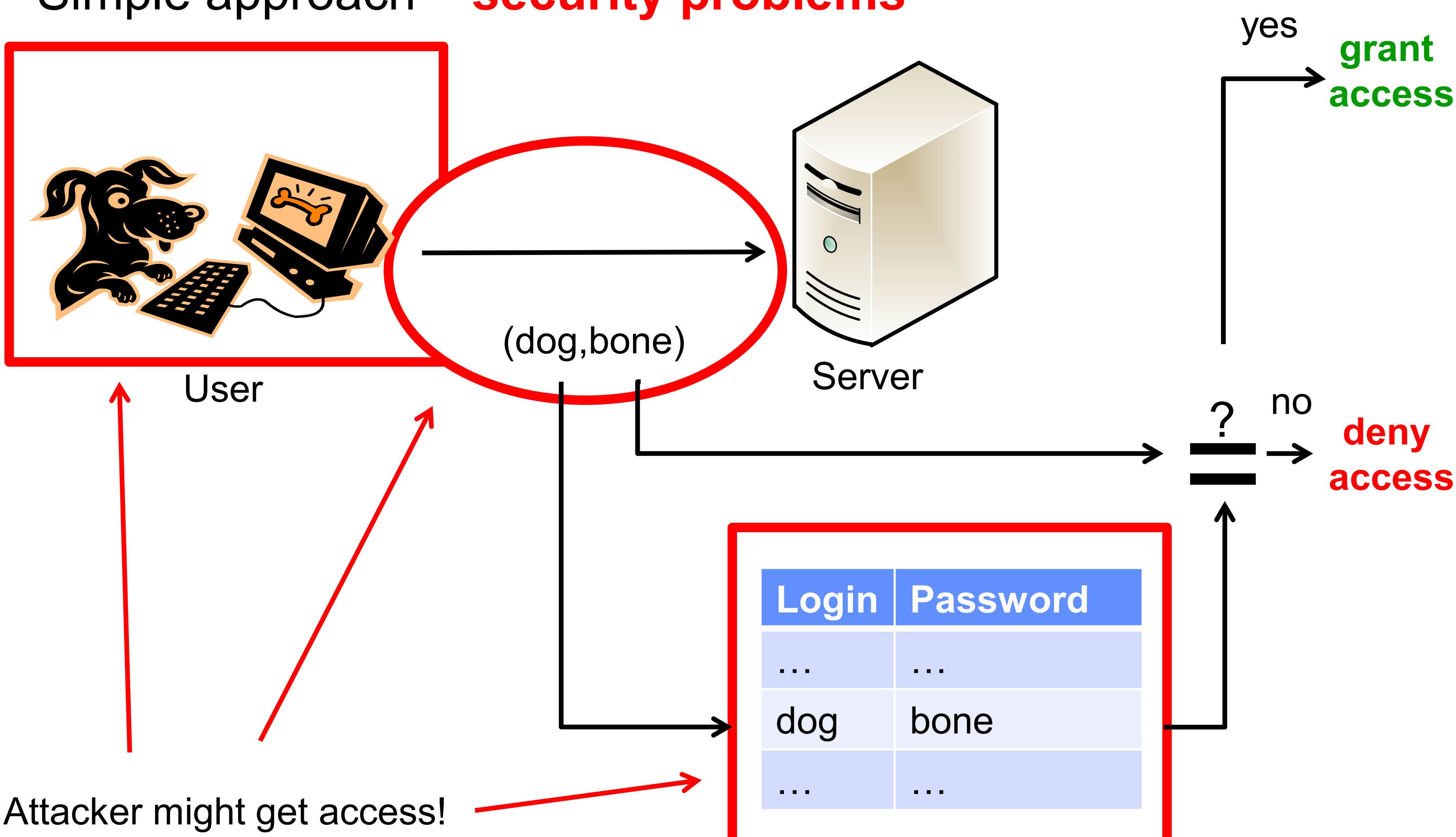
- Simple approach



Login	Password
…	…
dog	bone
…	…

Password based authentication

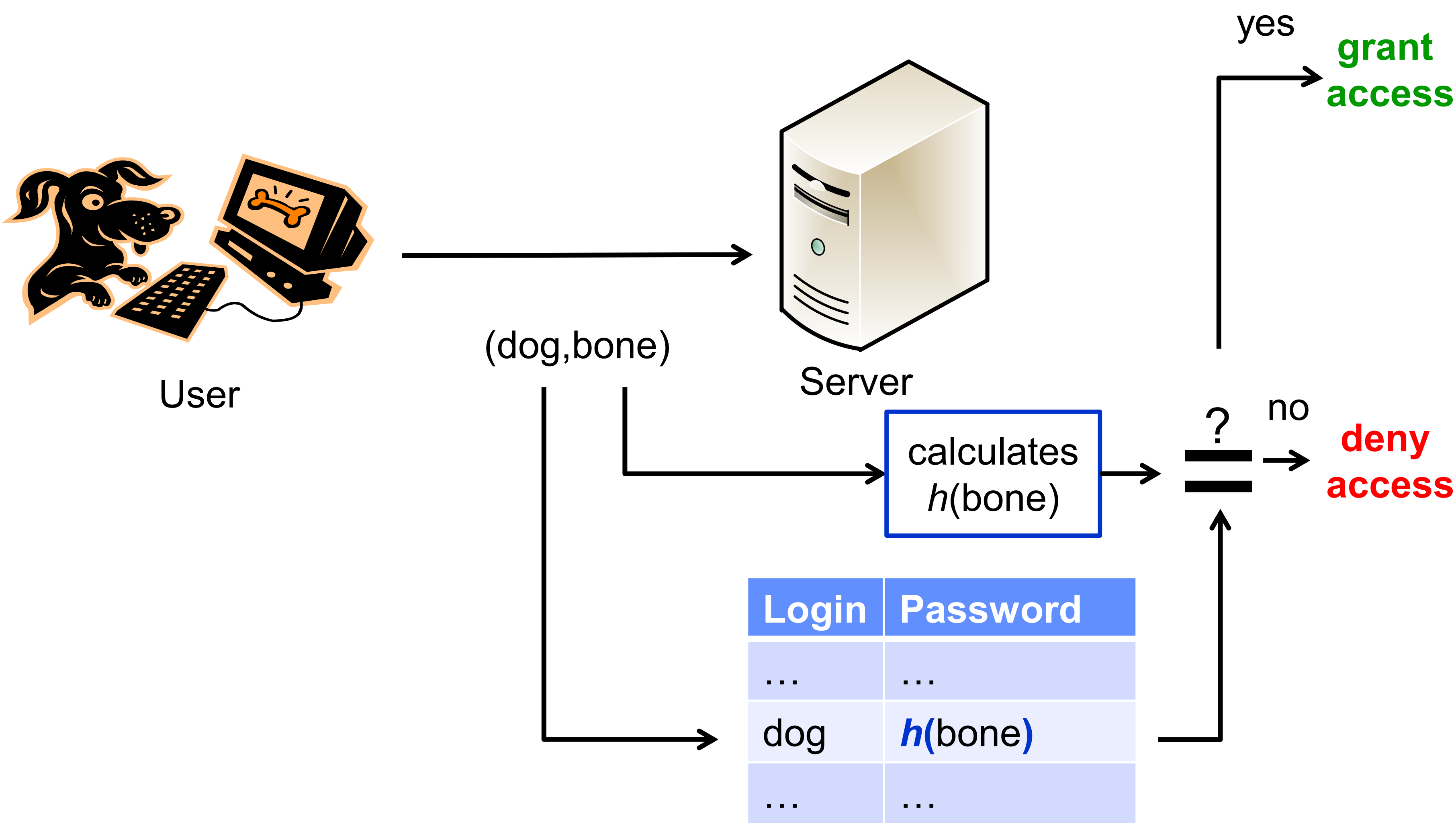
- Simple approach – **security problems**



Login	Password
…	…
dog	bone
…	…

Password based authentication

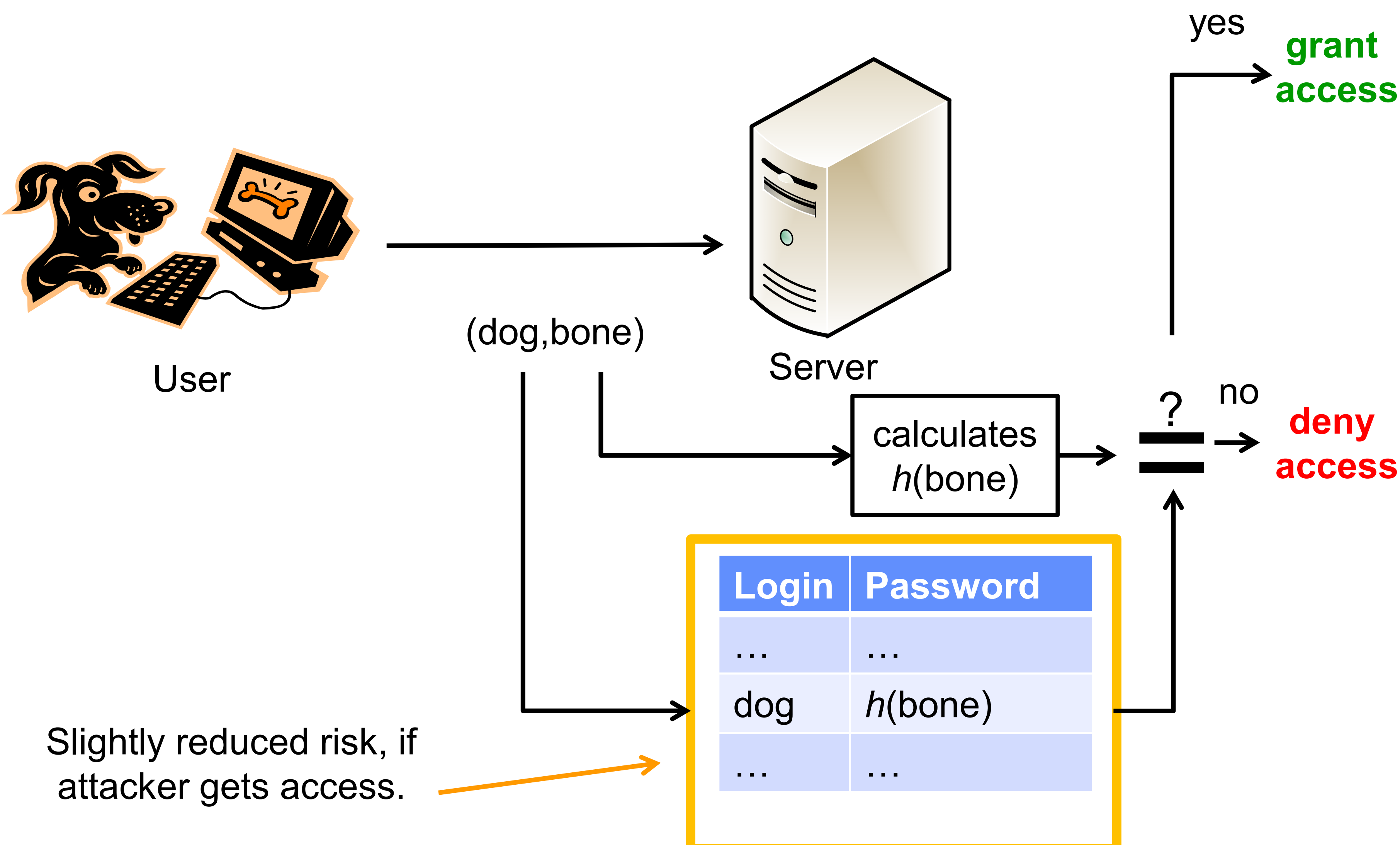
- Enhanced approach using one way (hash) functions



Login	Password
…	…
dog	*h*(bone)
…	…

Password based authentication

- Enhanced approach using one way (hash) functions



Cryptanalytic Time – Memory Trade-Off

- Martin E. Hellman: “A Cryptanalytic Time – Memory Trade-Off”
- main idea:
  - store only certain parts of the lookup table
  - regenerate the missing parts on demand
- requires “reduce” function *f*
  - *f*: H → P (H: set of hash values, P: set of passwords)
  - note: *f* is NOT the inverse of *h*
- general procedure:
  - calculate a chain of hash and reduce function calls
    - p → h() → f() → h() → f() → h () … → f() → p’
  - store first and last value in a table
    - sort by the last value
  - length of chain influences Time – Memory trade-off

Cryptanalytic Time – Memory Trade-Off

- Example:

Cryptanalytic Time – Memory Trade-Off

- 2^{nd} example
  - breaking of PINs
  - $h(x) := (x \cdot 7807) \bmod 16157$
  - $f(x) := x \bmod 9000 + 1000$

- PIN-table:

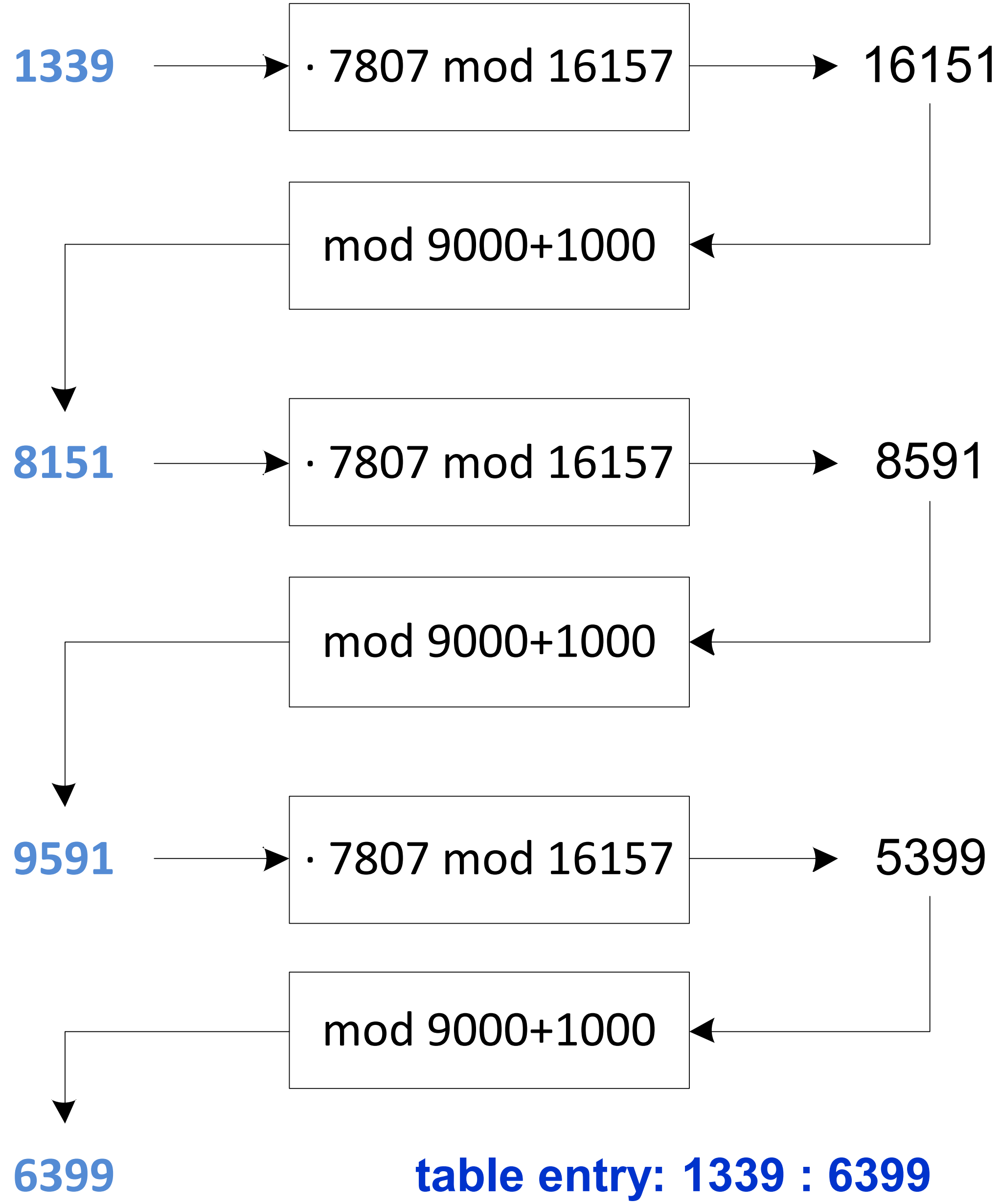
1309–9139–7018–**2139**

2439–9327–4447–**4493**

1084–4677–6676–**5207**

1339–8151–9591–**6399**

3128–8069–6697–**7584**

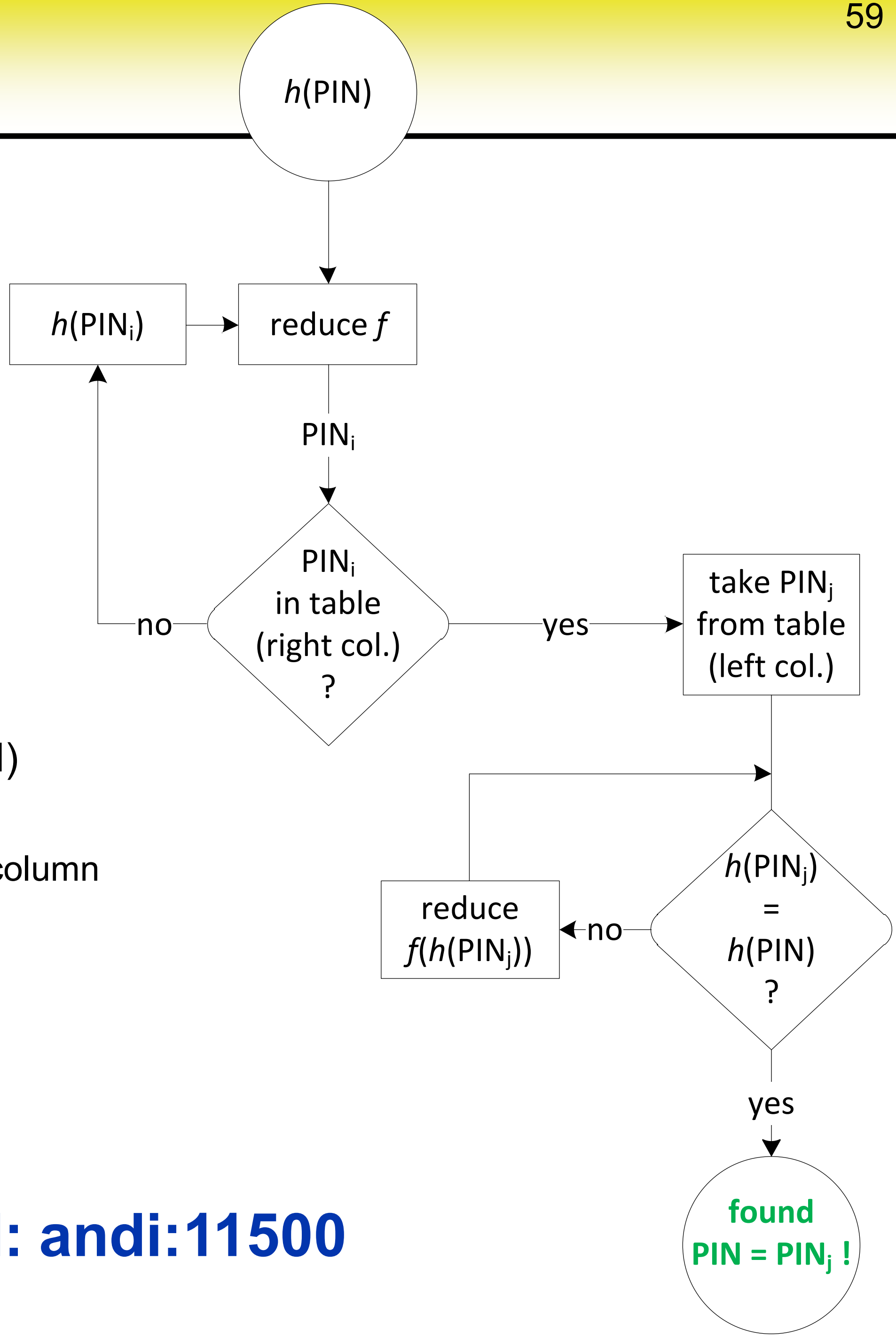


Cryptanalytic Time – Memory Trade-Off

- PIN-table:

2439–9327–4447–4493–**1003**
1084–4677–6676–5207–**1037**
3128–8069–6697–7584–**1040**
2824–9820–7932–3500–**4013**
1339–8151–9591–6399–**7706**
1309–9139–7018–2139–**9992**

- Breaking a PIN:
  - Goal: find PIN for given hash value *h*(PIN)
  - Algorithm:
    - 1. hash / reduce until value is in the right column
    - 2. take left column value
    - 2. hash / reduce until PIN is found
  - $h(x) := (x \cdot 7807) \bmod 16157$
  - $f(x) := x \bmod 9000 + 1000$



Try to break Andi's password: andi:11500

Remaining problems of password based authentication based one way functions

Login	Password
…	…
dog	*h*(bone)
…	…

- remaining possible attack:
  - pre-computation
- countermeasure:
  - salt & pepper!
  - *h*(x) → *h*(**salt**,x) → *h*(**salt**,*f*(x,**pepper**))
  - salt:
    - long (e.g. 128 bit) random value
    - some part could be stored together with password (i.e. 104 bit)
    - some part could not be stored at all (i.e. 24 bit)
      - verification: iterate over all possible salt values
  - pepper:
    - random value
    - stored separate from password list
    - unique per system or per password
  - additional: repeated hashing

➔ pre-computation has to be done *for each possible salt & pepper*

Remaining problems of password based authentication based one way functions

- remaining possible attack:
  - **dictionary attack**
  - problem: people do not chose passwords **randomly**
  - often names, words or predictable numbers are used
  - http://www.whatsmypass.com/the-top-500-worst-passwords-of-all-time
  - attacker uses dictionaries for brute force attack
  - prominent program: *John the Ripper*
    - supports dictionary attacks and password patterns

Login	Password
…	…
dog	*h*(salt,bone)
…	…

- possible solutions:
  - enforce password rules
    - consider usability
  - pre-check passwords (e.g. using John)
  - train people to "generate" good passwords
    - Example: sentence → password
    - "This is the password I use for Google mail" → "TitpIu4Gm"

Hash functions for storing passwords

Usual requirement on cryptographic hash functions:

- quickly process large amounts of input data

Problem:

- makes brute force attacks more efficient
- Example: GPU – 200.000.000.000 Hash/s (>2^{37}) [MD5]

Special case passwords:

- small inputs (<512 bit)
- some waiting time for login acceptable
  - ~ 1 second

  ➔ Hash function does not need to be super efficient

Therefore, to make brute force attack more difficult:

- Hash function ***should not be efficient*** (implementable)

Hash functions for storing passwords

Hash function ***should not be efficient*** implementable

- Software:
  - Consider modern CPUs
    - Multi-Core / Multi-Threaded
    - SIMD / Vector Extensions (AVX512)
    - crypto extensions (AES / SHA)
    - Cache-sizes (L1, L2, L3 Cache)
    - Branch Prediction
    - …
  - Consider commodity „special hardware“
    - GPUs
    - KI /ML accelerators

- Hardware
  - FPGA
  - special ASICs
    - Bitcoin-Mining

- future proven
  - easily adaptable (parameters) considering future hardware (improvements)

Hash functions for storing passwords

Some examples

- **`bcrypt`**
  - Niels Provos, David Mazières: „A Future-Adaptable Password Scheme“, USENIX, 1999
  - based on Blowfish
    - symmetric block cipher

```
round_keys=EksBlowfishSetup(cost, salt, input) // inefficient!
hash="OrpheanBeholderScryDoubt" // 3 x  64-bit blocks
loop (64)
   {
        hash=Blowfish_ECB(round_keys,hash)
   }
```

– good protection against software / GPU based brute-force-attacks
– weak protection against ASIC-based brute-force-attacks

Hash functions for storing passwords

Some Examples

- **PBKDF2** (Password-Based Key Derivation Function 2)
  - originally part of RSA Laboratories PKCS#5-standard
    - purpose: derive symmetric keys from password
    - now RFC 2898
    - approved by NIST in SP 800-132 (December 2010)
  - *h*=`PBKDF2` (passwd, salt, iterations)

```
{
  h=Hash(passwd||salt||iterations);
  loop(iterations-1)
    {
       h=h XOR Hash(passwd||h);
    }
  return h;
}
```

  – good protection against CPU-based software brute-force-attacks
  – weak protection against ASIC/FPGA/GPU-based brute-force-attacks

PBKDF2— increase memory consumption

- *h*=PBKDF2 (passwd, salt, iterations)

```
{
  i=0
  h[i++]=Hash(passwd||salt||iterations);
  loop(iterations-1)
    {
      h[i+1]=h[i] XOR Hash(passwd||h[i]);
      i++;
    }
  sort(h[]);
  i=0;
  loop(iterations/2)
    {
     res=res + h[i] * h[i+1];
     i+=2;
    }
  return res;
}
```

Warning: Hand crafted crypto! - unverified -

Hash functions for storing passwords

- Some Examples
  - **`scrypt`**
    - Colin Percival: “Stronger Key Derivation Via Sequential Memory-Hard Functions”, 2009
    - published in RFC 7914
    - Goal: make hardware implementation expensive
    - Strategy:
      - Increase memory consumption
    - Realisation:
      - algorithm requires large vector of pseudorandom elements, which are access in pseudorandom order
    - Additionally: „Costs“ can be parameterised

Hash functions for storing passwords

- Some Examples
  - **`Argon2`**
    - Alex Biryukov, Daniel Dinu, Dmitry Khovratovich: "Argon2: the memory-hard function for password hashing and other applications", 2015
    - Winner of the Password Hashing Competition (PHC)
      - Community driven competition (2013-2015)
    - similar goals / solutions like scrypt
    - not so well analysed yet

Symmetric authentication systems

Key distribution:

like for symmetric encryption systems

Simple example (view of attacker)

The outcome of tossing a coin (Head (H) or Tail (T)) shall be sent in an authenticated fashion:

Message, MAC		Message	
		H	T
key	00	H,0	T,0
	01	H,0	T,1
	10	H,1	T,0
	11	H,1	T,1

Security: e.g. attacker wants to send T.

a) blind: get caught with a probability of 0.5

b) seeing: e.g. attacker gets (H,0) $\Rightarrow$ $k \in \{00, 01\}$

still both: (T,0) and (T,1) have a probability of 0.5

One Time Pad – Attacks on Integrity

- addition mod 4
- original ciphertext: **11**
- cases:

key	00	01	10	11
plaintext	11	10	01	00
manipulated plaintext	10	11	00	01
manipulated ciphertext	10	00	10	00

- 1. possibility: sent ciphertext **00**

resulting plaintext	00	11	10	01

- 2. possibility: sent ciphertext **10**

resulting plaintext	10	01	00	11

Goal/success of attack

a) key (total break)

b) procedure equivalent to key (universal break)

c) individual messages,

e.g. especially for authentication systems

- c1) one selected message (selective break)
- c2) any message (existential break)

Types of attack

severity

a) passive

- **a1) ciphertext-only attack**
- **a2) known-plaintext attack**

b) active

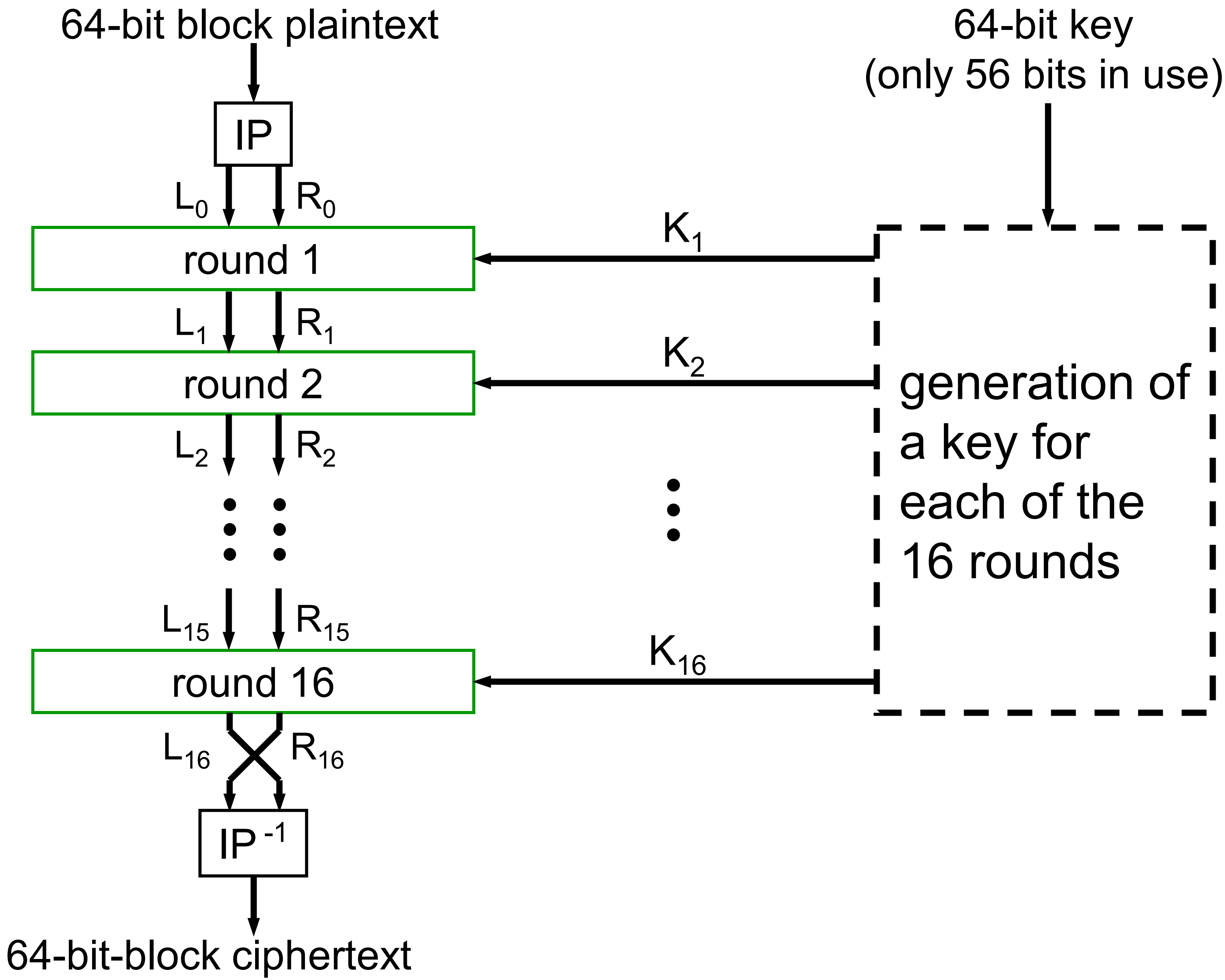
(according to encryption system; asym.: either b1 or b2;
sym.: b1 or b2)

- **b1) signature system: plaintext → ciphertext (signature)**
  (chosen-plaintext attack)
- **b2) encryption system: ciphertext → plaintext**
  (chosen-ciphertext attack)

adaptivity

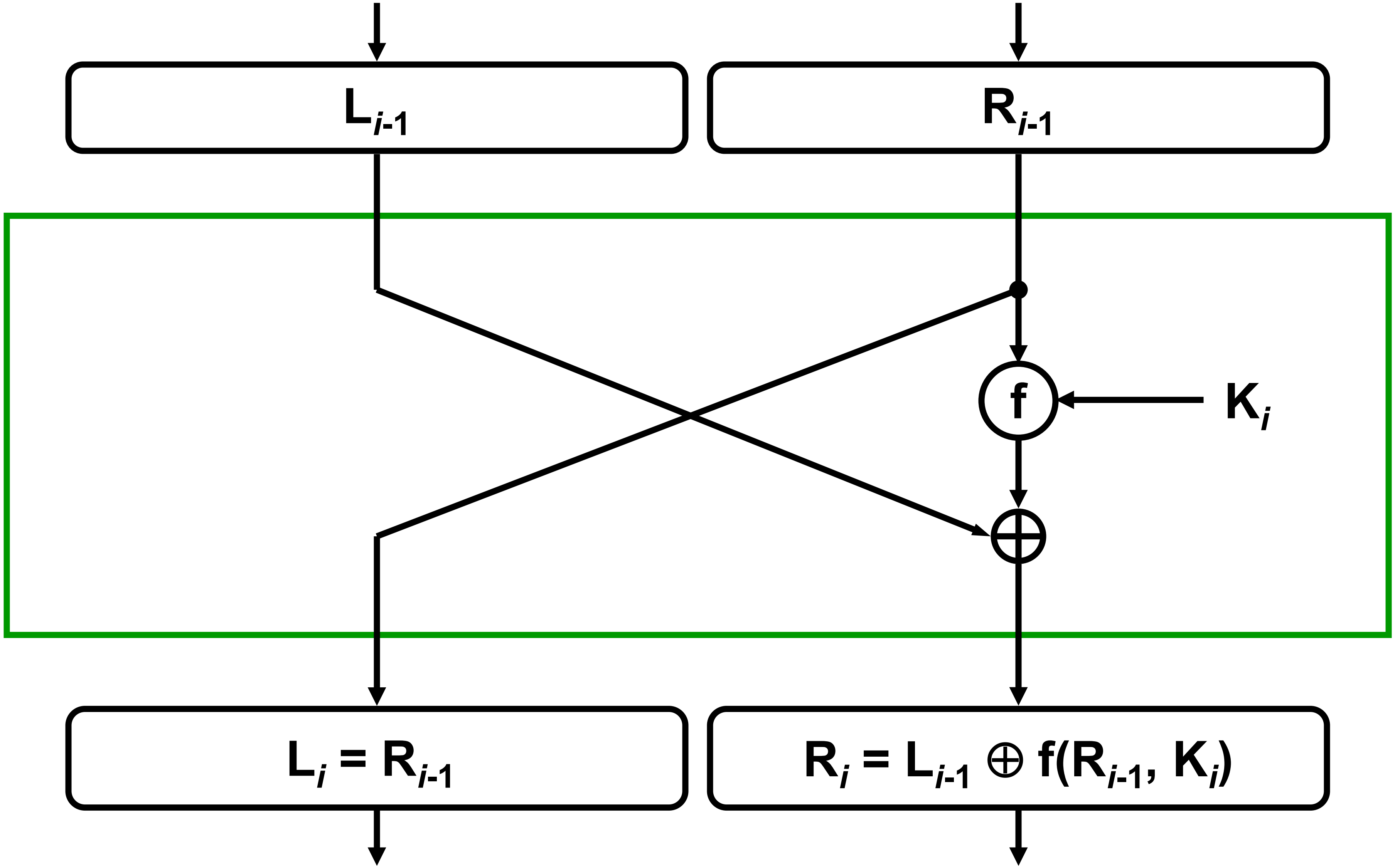
- **not adaptive**
- **adaptive**

Symmetric Cryptosystem DES

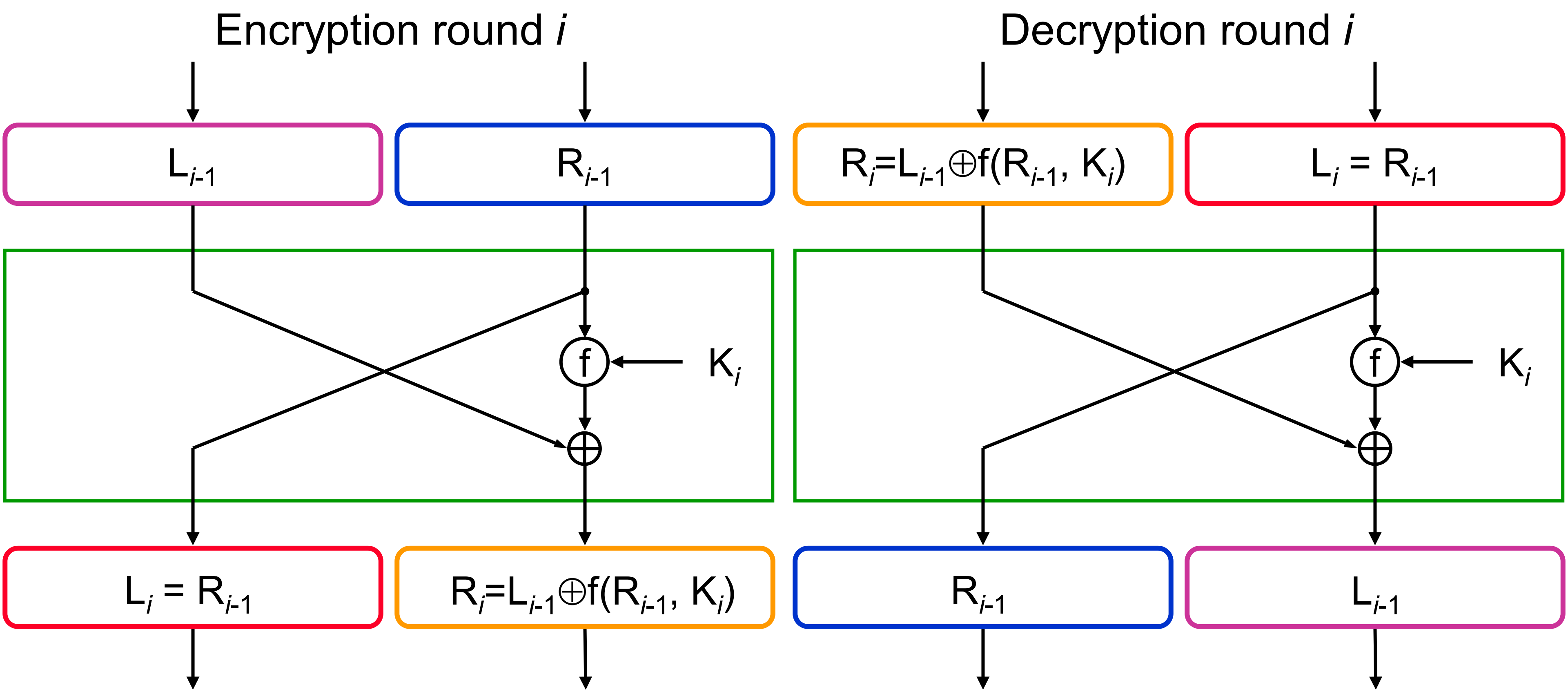


One round

Feistel ciphers



Why does decryption work?



Decryption

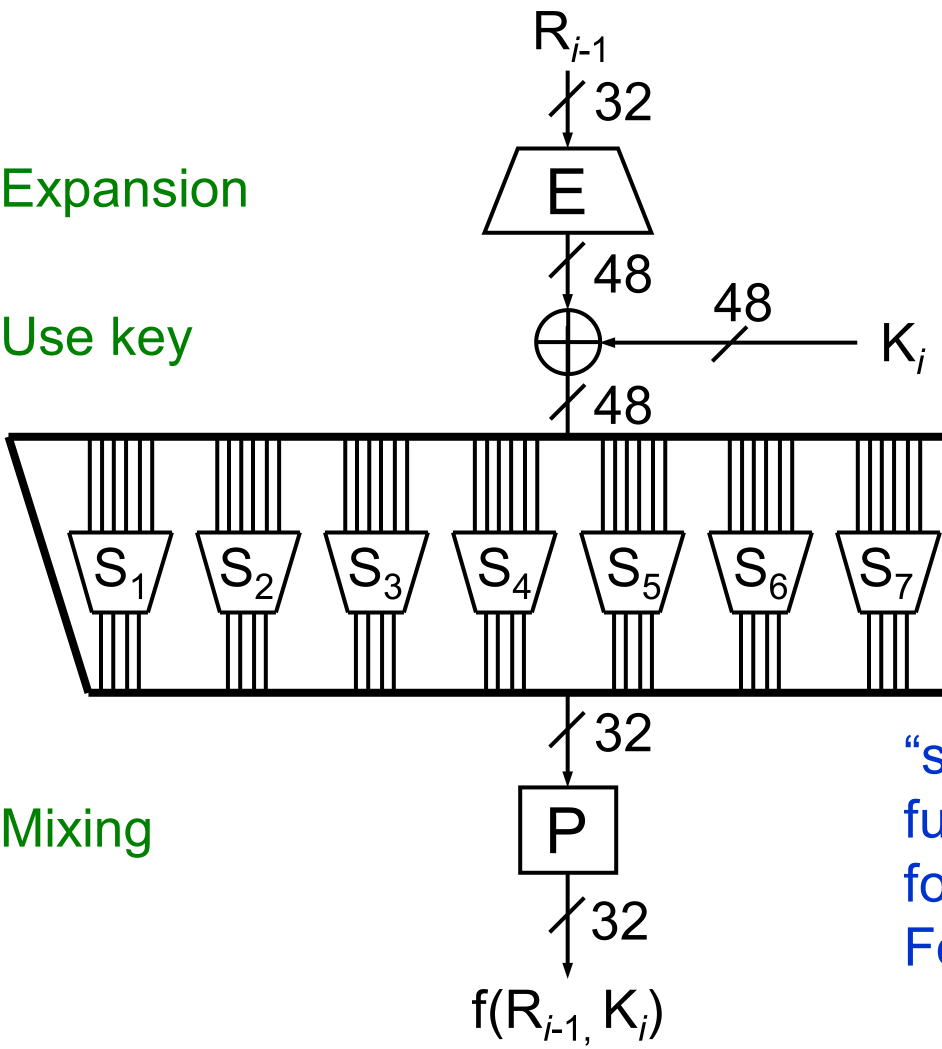
→ trivial

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

replace R_{i-1} by L_i

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

Encryption function f



Expansion

Use key

Mixing

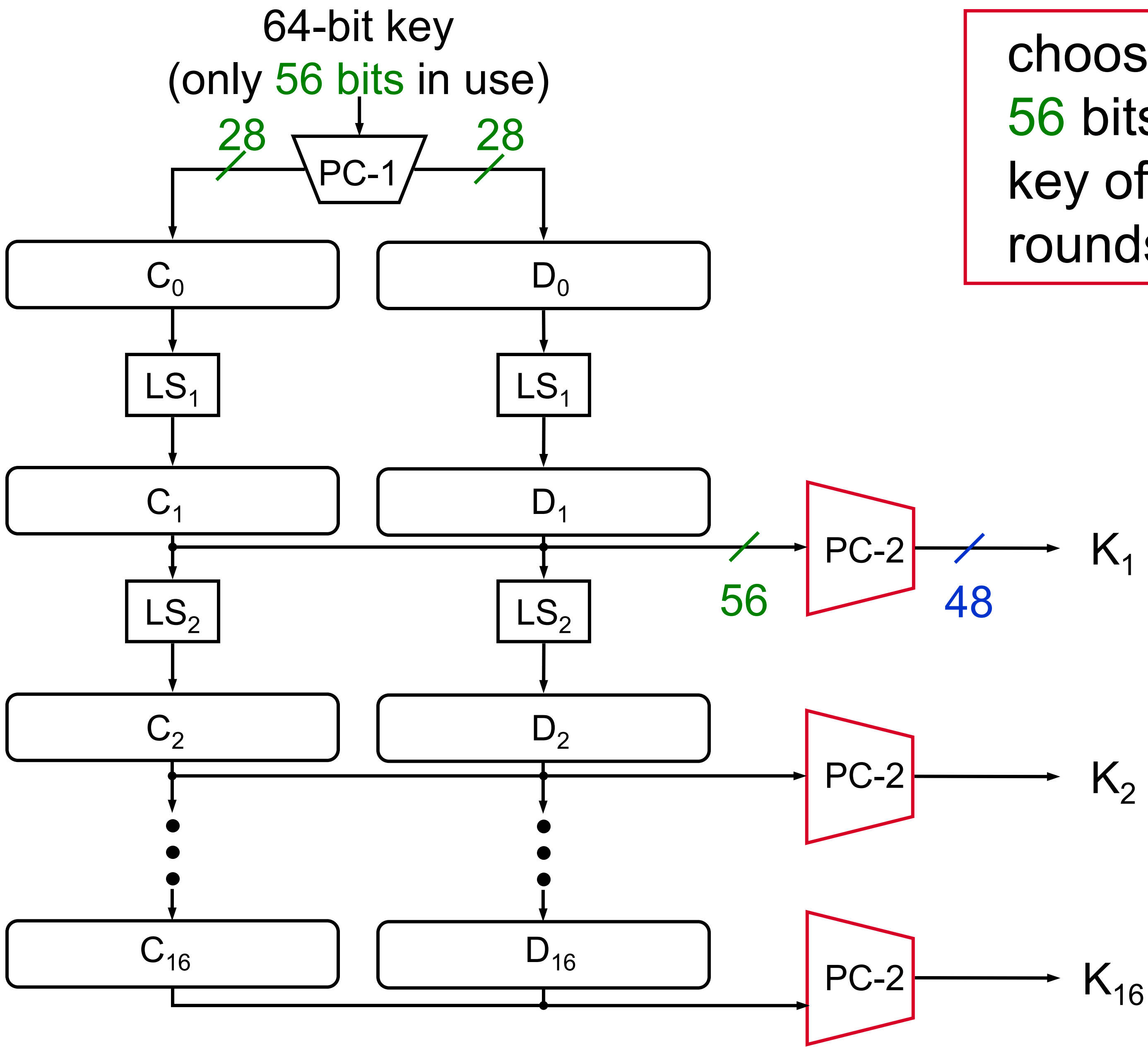
Make f (and DES) non-linear (permutations and $\oplus$ are linear)

“substitution box” S can implement any function $s : \{0,1\}^6 \rightarrow \{0,1\}^4$, for example as table.
For DES, the functions are fixed.

Terms

- Substitution-permutation networks
- Confusion - diffusion

Generation of a key for each of the 16 rounds

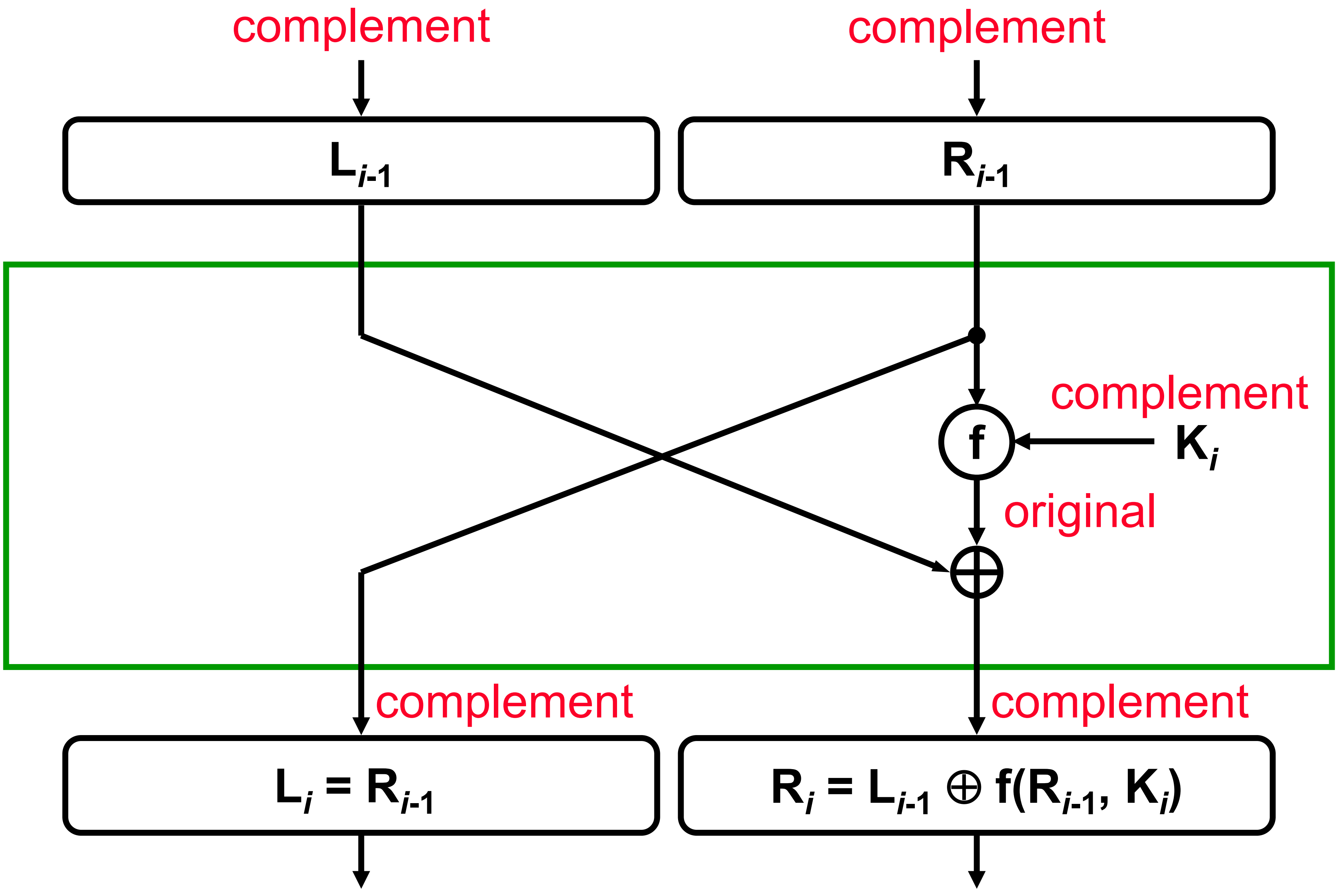


choose 48 of the 56 bits for each key of the 16 rounds

The complementation property of DES

$$\mathrm{DES}(\bar{k}, \bar{x}) = \overline{\mathrm{DES}(k, x)}$$

One round


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

Encryption function f