Willkommen zur Vorlesung

Sicherheit:
Fragen und Lösungsansätze

im Wintersemester 2012 / 2013

Prof. Dr. Jan Jürjens
TU Dortmund, Fakultät Informatik, Lehrstuhl XIV

Teil 5 (Fundamentals of Cryptography)
v. 03.12.2012

Themen der Vorlesung

Part I: Challenges and Basic Approaches

1) Interests, Requirements, Challenges, and Vulnerabilities
2) Key Ideas and Combined Techniques

Part II: Control and Monitoring

3) Fundamentals of Control and Monitoring
4) Case Study: UNIX

Part III: Cryptography

5) **Fundamentals of Cryptography**
6) Case Studies: PGP and Kerberos
7) Symmetric Encryption
8) Asymmetric Encryption and Digital Signatures with RSA
9) Some Further Cryptographic Protocols

Part IV: Access Control

10) Discretionary Access Control and Privileges
11) Mandatory Access Control and Security Levels

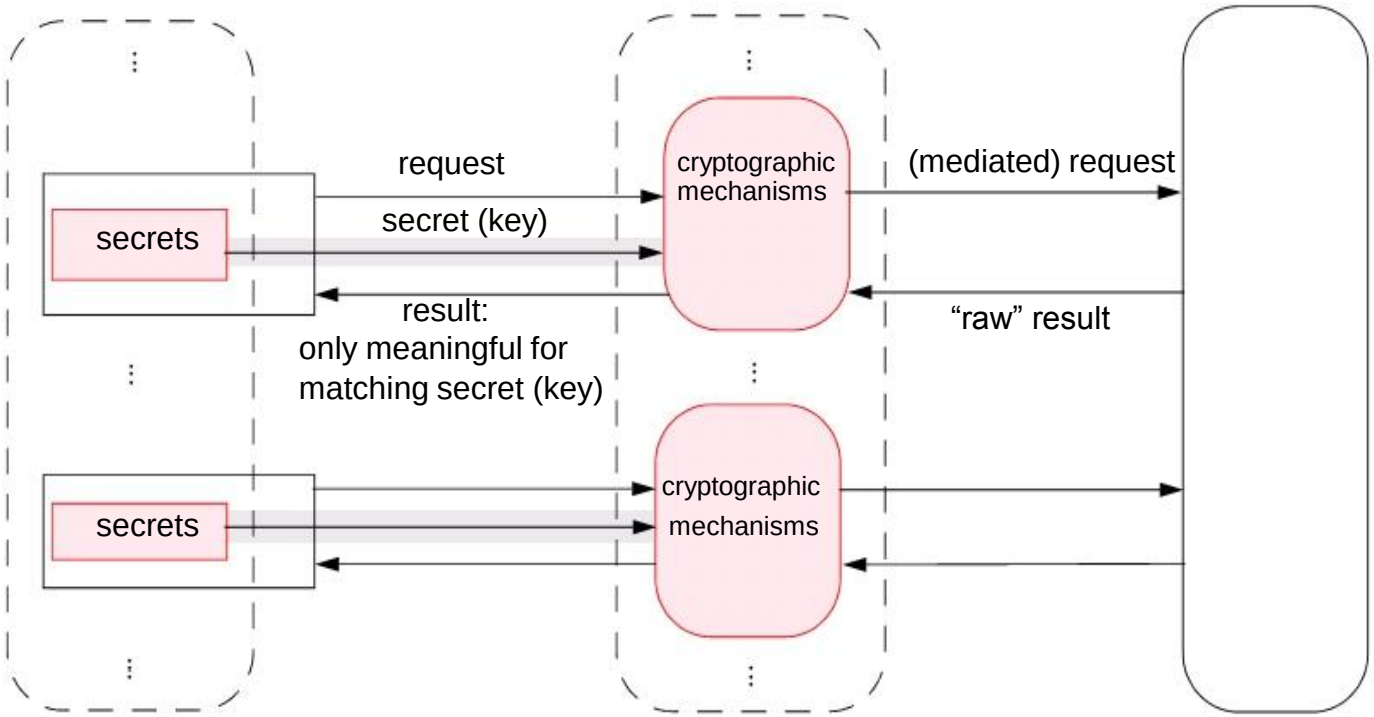
Part V: Security Architecture

12) Layered Design Including Certificates and Credentials
13) Intrusion Detection and Reaction

Cryptography

conceptually: (cryptographic)
knowledge base on permissions
(and prohibitions)

conceptually: (cryptographic)
control component



participating subjects

- generate, store and employ secrets
- exploit physical isolation
  (indicated by the gray areas)

controlled objects

Cryptography

- is usually closely intertwined with control and monitoring

- binds a successful and meaningful execution of an operation or interaction to providing a suitable *secret key* as input

- achieves *virtual isolation* between participants:
participants that share a cryptographic key are
virtually isolated from those that do not

- enables *cooperation in the presence of threats* based on limited trust:
participants that autonomously generate and secretly keep appropriate cryptographic keys can enforce their security *interests* by themselves

Basic cryptographic blocks

- encryption
- authentication
- anonymization
- randomness and pseudorandomness
- one-way hash functions
- timestamps

Encryption: functionality

- the sender *S* transforms the original bit string *m* to be transmitted into another bit string *m*

  such that *only* the designated receiver *R* (and possibly the sender) is enabled to recover the original bit string

- (probabilistic) *key generation* algorithm *Gen* (one parameter and one result):
  - *l* — *security parameter* (key length, ...)
  - (ek_R, dk_R) — matching *key pair*
- (probabilistic) *encryption* algorithm *Enc* (two parameters and one result):
  - ek_R — *encryption key*
  - *m* — *plaintext* (original message)
  - *m*' = *Enc* (ek_R, *m*) — *ciphertext* (transformed bit string)
- (probabilistic) *decryption* algorithm *Dec* (two parameters and one result):
  - dk_R — *decryption key*
  - c — *ciphertext*
  - c' = *Dec* (dk_R, c) — (hopefully) recovered plaintext

Encryption: correctness property

- encryption algorithm *Enc* and decryption algorithm *Dec* should be inverse whenever a *matching key pair* (ek_R, dk_R) generated by *Gen* has been employed:

  for all plaintexts m, $Dec\ (dk_R, Enc\ (ek_R, m)) = m$

Encryption: secrecy property

- ***naive version***
  for all plaintexts *m*, without a knowledge of the decryption key dk_R ,
  m cannot be “determined” from the ciphertext *m*

- **(*informal*) *semantic version***
  an unauthorized observer of a ciphertext cannot infer
  anything new about the corresponding plaintext, i.e.,

  for all plaintexts *m*, without a knowledge of the decryption key dk_R ,
  any *property* of *m* that can be “determined” from the ciphertext *m*
  could also be “determined” without knowing *m* at all

- **(*informal*) *operational version***
  an unauthorized observer of ciphertexts
  cannot separate apart any pair of ciphertexts, and thus
  cannot solve the problem of

  assigning a specific plaintext to a ciphertext

Operational secrecy as indistinguishability

for a *probabilistic* setting, considering
sequences of plaintexts and of matching key pairs
of *increasing* length (taken as a *security parameter*),
we have *indistinguishability of ciphertexts*:

for any pair of plaintext sequences
(m_1', m_1'', m_1''',…)
and
(m_2', m_2'', m_2''',…),

without a knowledge of the sequence of decryption keys employed,

the resulting sequences of ciphertexts are
“computationally indistinguishable”

Basic assumptions

- approved algorithms *Gen* and *Dec* and *Enc* are publicly known

- decryption keys are strictly kept secret

- given approved algorithms and seen from the perspective of the end-users, enforcing the *confidentiality* of messages by encryption
  basically relies *only* on
  - selecting appropriate keys (as determined by the security parameter)
  - actually hiding the decryption keys

Relationship between encryption key and decryption key

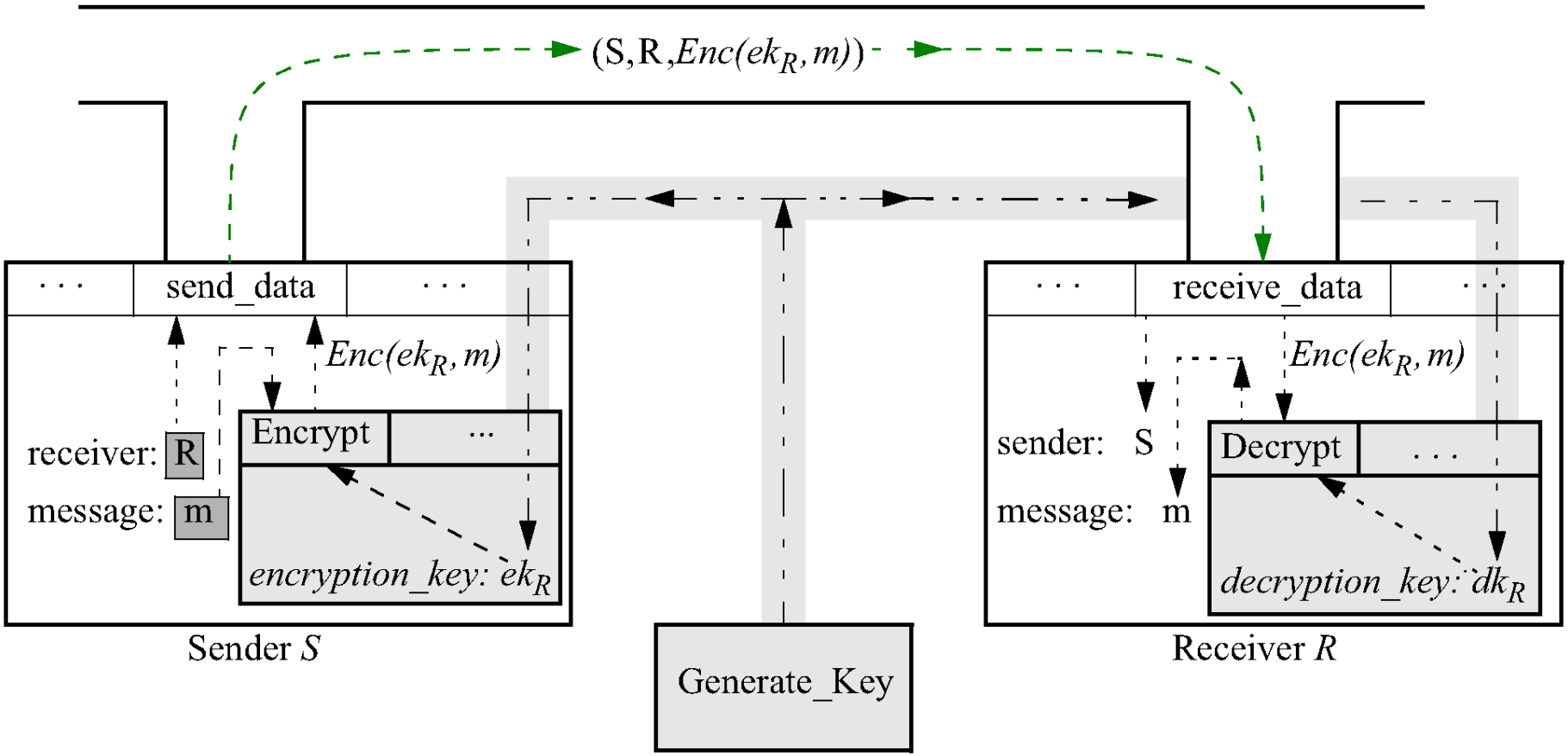
- ***symmetric* (or *secret-key*) mechanism:**
  - the encryption key is (basically) *equal* to the decryption key

- ***asymmetric* (or *public-key*) mechanism:**
  - the encryption key is essentially *different* from the decryption key
  - an additional *secrecy property* (*naive* version) is required:

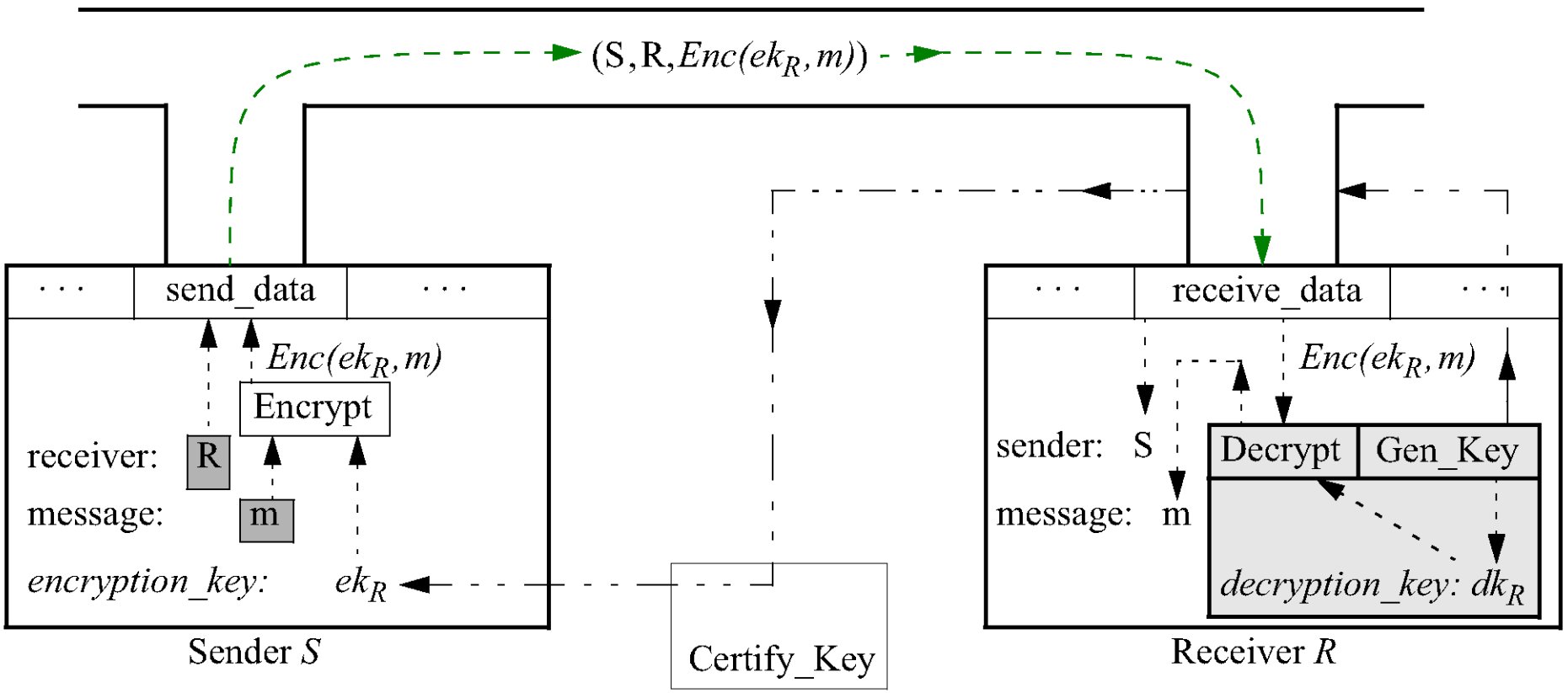
    the (private) decryption key dk_R

    cannot be “determined” from the (public) encryption key ek_R

Symmetric encryption



Asymmetric encryption



Symmetric and asymmetric encryption mechanisms

Feature	Symmetric	Asymmetric
generating and *distributing* keys	both partners are *equally* involved	designated *receiver* has a *distinguished* role
protection requirements	key generation / communication of the secret key and storage of the secret key must be protected on *both sides*	key generation and storage of the private key must be protected on the side of the *receiver only*
contributions of the *trusted third parties*	generate and distribute secret keys	certify public keys

Authentication: basic approach

designated sender *S*

- prepares for transmitting a bit string *m* as a message
  by computing another bit string $red_{S,m}$
  as a *cryptographic piece of evidence*
  (*cryptographic exhibit* or *cryptographic check redundancy*)
- forwards the compound (S, m, $red_{S,m}$)
  - S sender identification
  - m original bit string
  - $red_{S,m}$ computed bit string

receiver

- receives such a compound of the form (S , m , $red_{S,m}$)
- checks whether the message part originates from the claimed sender without modification by inspecting the included cryptographic exhibit (must depend on both the designated sender and the message)
- either accepts (as authentic) or rejects the received message

Authentication: functionality

- (probabilistic) *key generation* algorithm *Gen* (one parameter and one result):
  - l — *security parameter* (key length, ...)
  - (tks, aks) — matching *key pair*

- (probabilistic) *authentication* algorithm *Aut* (two parameters and one result):
  - ak_S — *authentication key*
  - m — message
  - $red_{S,m} = Aut\ (ak_S, m)$ — *cryptographic exhibit*

- (probabilistic) Boolean-valued *authenticity verification* algorithm *Test* (three parameters and Boolean result):
  - tk_S — *test key/verification key*
  - m — received message
  - red — cryptographic exhibit

Authentication: (weak) correctness property

- authentication algorithm *Aut* and authenticity verification algorithm *Test* should be complementary
  whenever a *matching key pair* (tk_S, ak_S) generated by *Gen*
  has been employed:

  for all messages m, *Test* (tk_S, m, *Aut* (ak_S, m)) = *true*

Authentication: unforgeability

- **(*naive*) *unforgeability property***
  for all messages *m*,
  without a knowledge of the authentication key ak_S ,
  one cannot "determine" a bit string *red* such that *Test* (tk_S, *m*, *red*) = *true*

- **(*naive*) *strong correctness property*,**
  complemented by a ***weak unforgeability property***

  for all messages *m* and for all bit strings *red*,
  Test (tk_S, *m*, *red*) = *true* iff *red* = *Aut* (ak_S, *m*)

  and

  without a knowledge of the authentication key ak_S ,
  one cannot "determine" this solely accepted cryptographic exhibit

Basic assumptions

- approved algorithms *Gen*, *Aut* and *Test* are publicly known
- authentication keys are strictly kept secret
- given approved algorithms and seen from the perspective of the endusers, enforcing the *integrity* and *authenticity* of messages (in the sense of detection of violations) by authentication basically relies *only* on
  - selecting appropriate keys (as determined by the security parameter)
  - actually hiding the authentication keys

Relationship between test key and authentication key

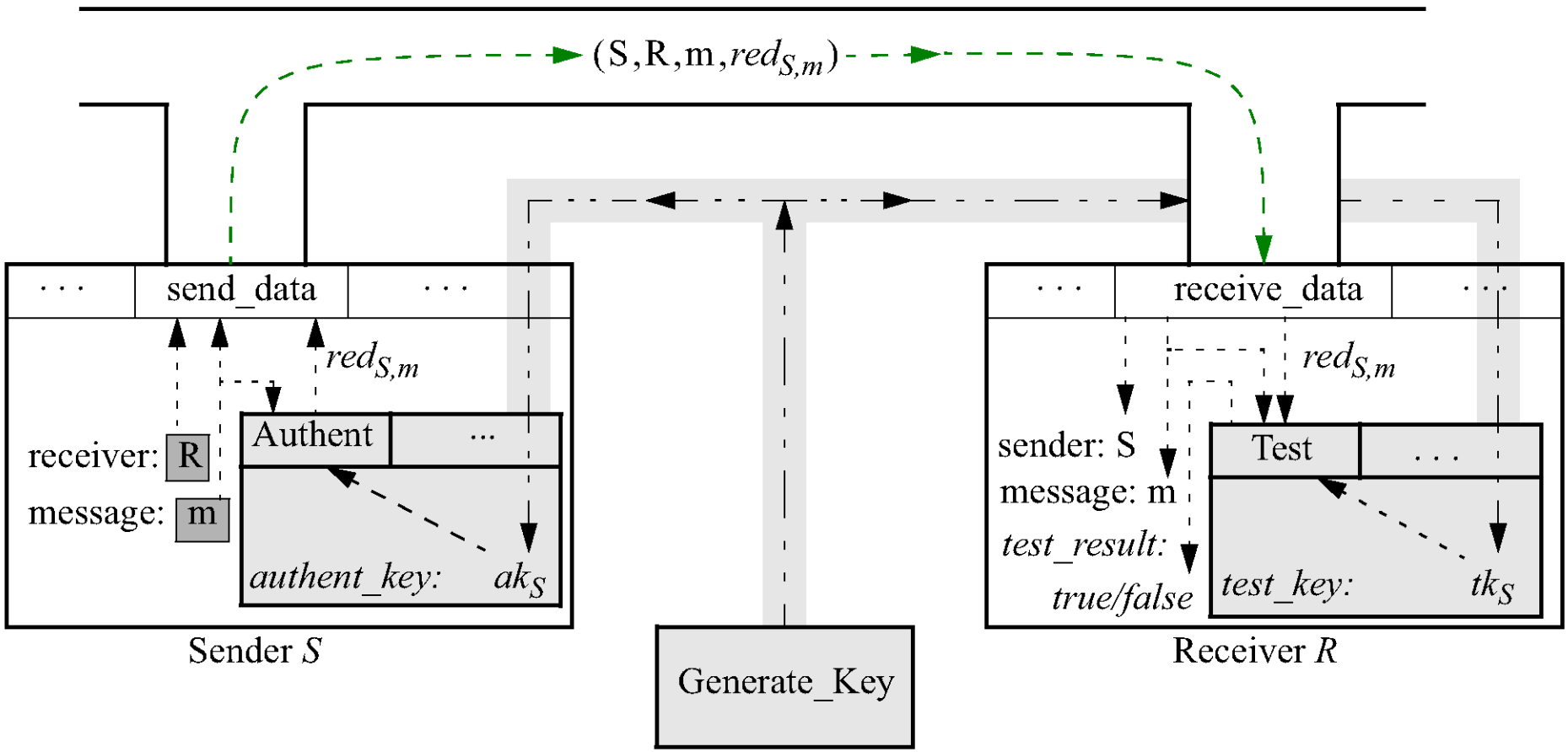
- ***symmetric* (or *secret-key*) mechanism:**
  - the test key is (basically) *equal* to the authentication key

- ***asymmetric* (or *public-key*) mechanism:**
  - the test key is essentially *different* from the authentication key
  - an additional *secrecy property* (*naive* version) is required:

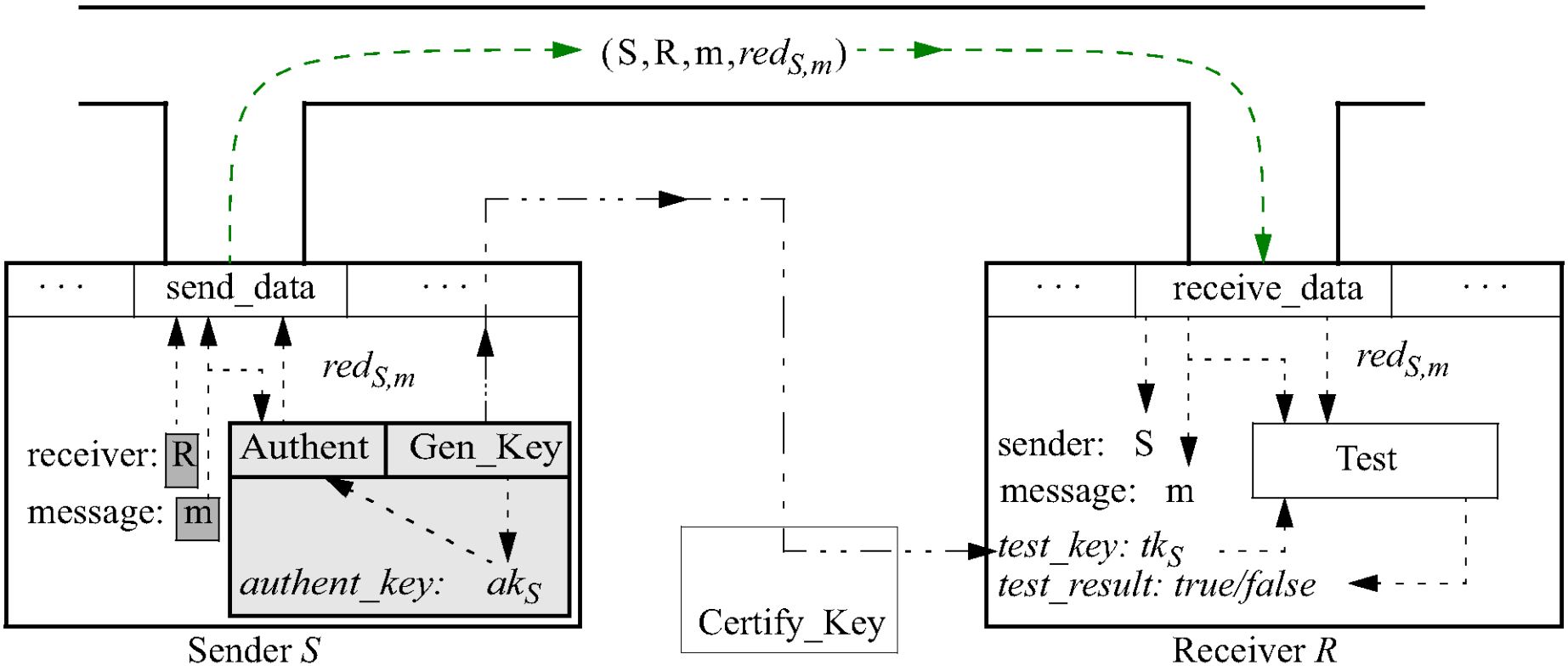
    the (private) authentication key ak_S

    cannot be “determined” from the (public) test key tk_S

Symmetric authentication



Asymmetric authentication (digital signing)



Symmetric and asymmetric authentication mechanisms

Feature	Symmetric	Asymmetric
generating and distributing keys	both partners are equally involved	designated sender has a distinguished role
protection requirements	key generation/ communication of the secret key and storage of the secret key must be protected on both sides	key generation and storage of the private key must be protected on the side of the sender only
contributions of the trusted third parties	generate and distribute secret keys	certify public keys
non-repudiation/ digital signatures	no	yes

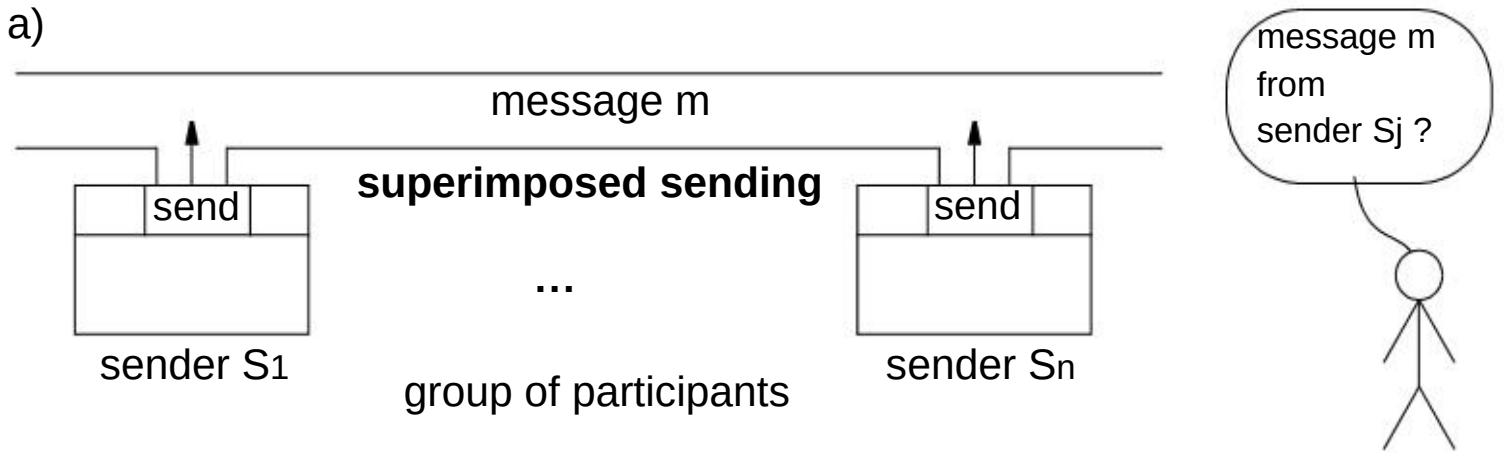
Anonymization

- the interest in *anonymity* or, more generally, in *non-observability* can be seen as strengthened forms of (message) *confidentiality*:
  - not only the message itself should be kept secret
  - but also the full *activity* of a message transmission

- from the point of view of an observer who is not designated to learn about an activity or a sequence of activities:
  any actually occurring activity
  is *indistinguishable* from
  any other activity in a preferably large activity domain
  from which the actually occurring activity has been selected

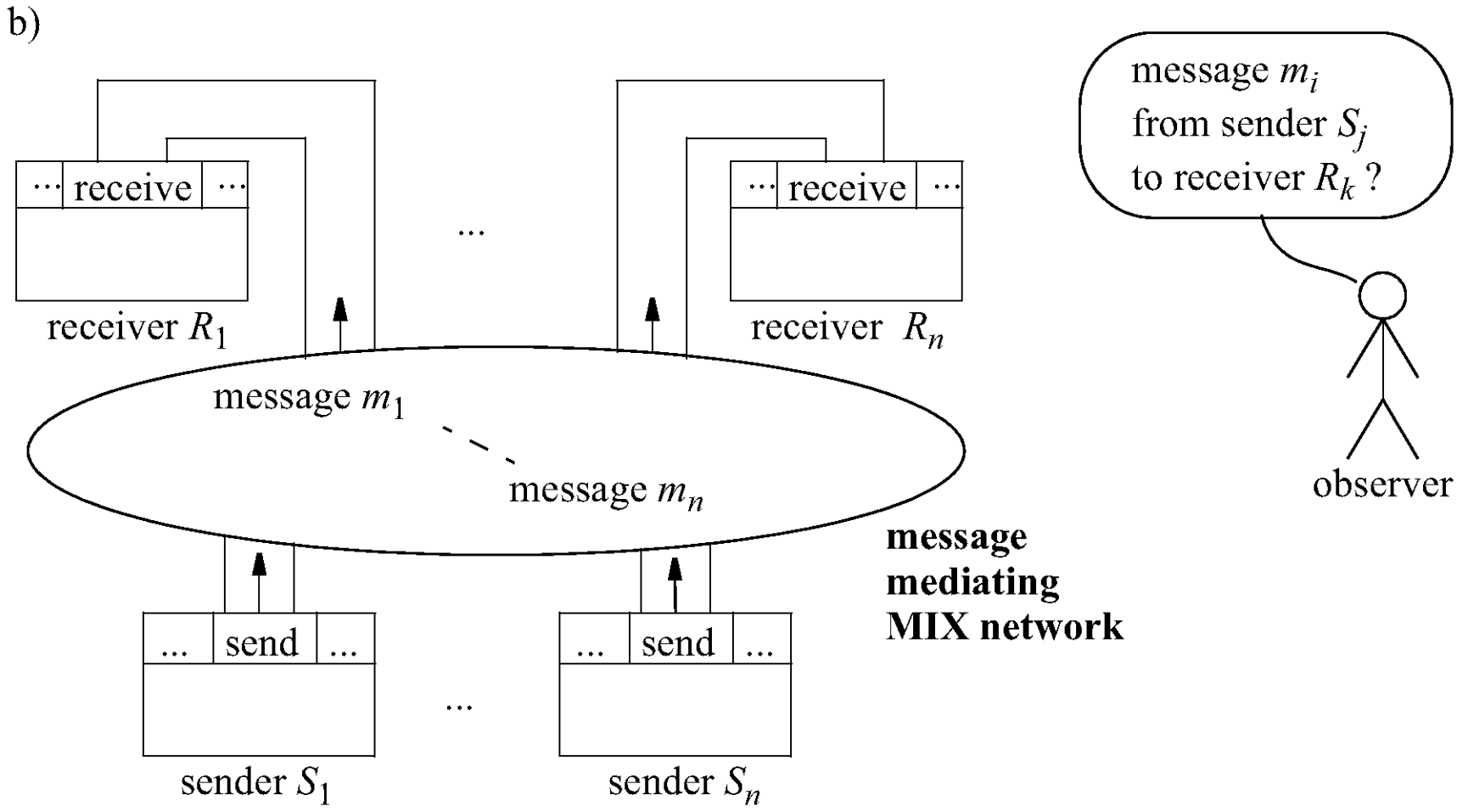
- the actual activity is *indistinguishably hidden*
  in a preferably large domain of other possibilities,
  often called an *anonymity class*

Sender anonymity

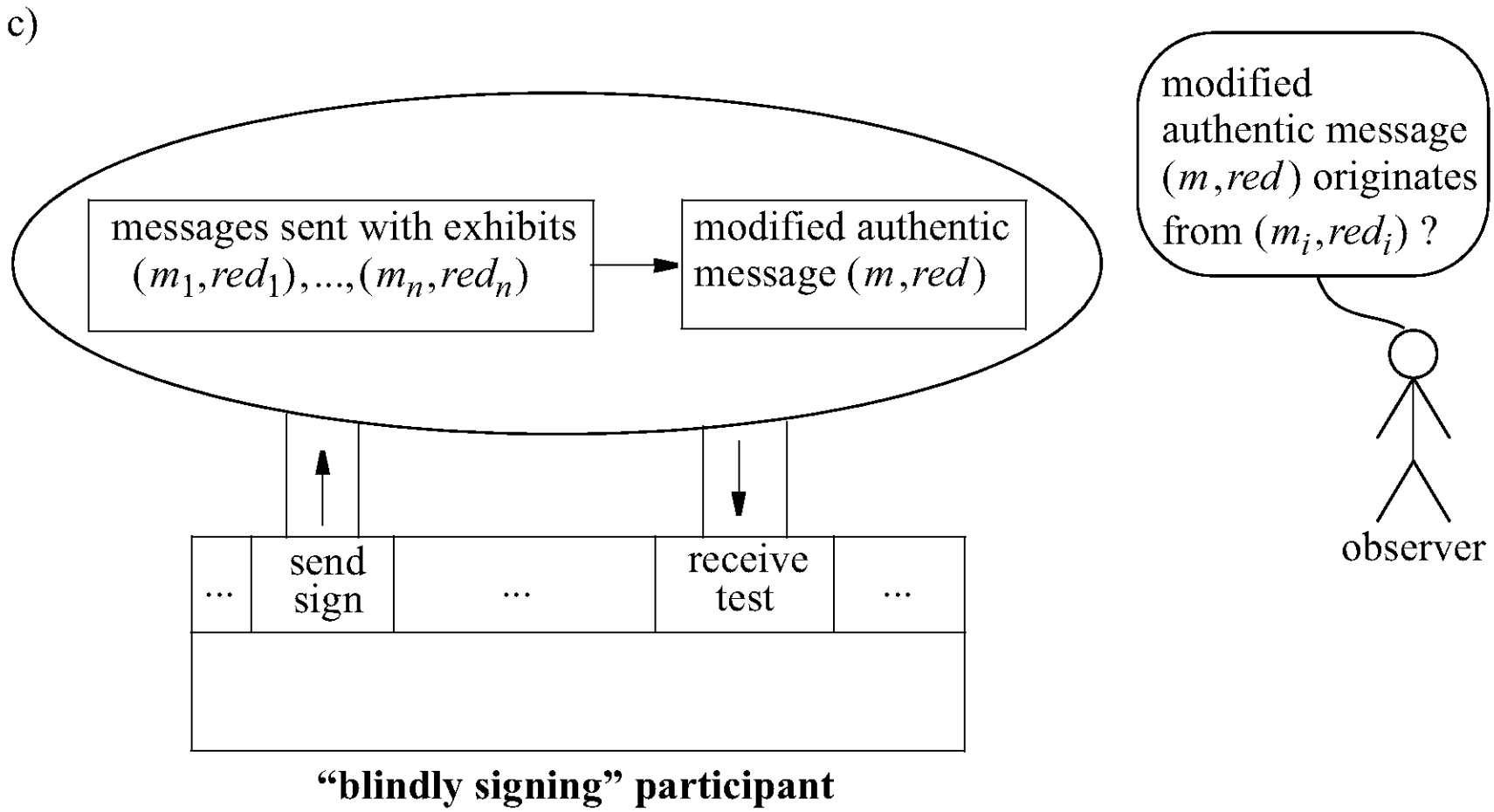


- **activity domain:**
  participants $S_1,\ldots, S_n$ sending and receiving messages
- **anonymity property:**
  by observing an actual message *m*,
  a non-designated observer cannot “determine” the actual sender S_j
- **mechanism:**
  superimposed sending

Sender-receiver anonymity



Anonymity by unlinkability



Unlinkability and blind signatures

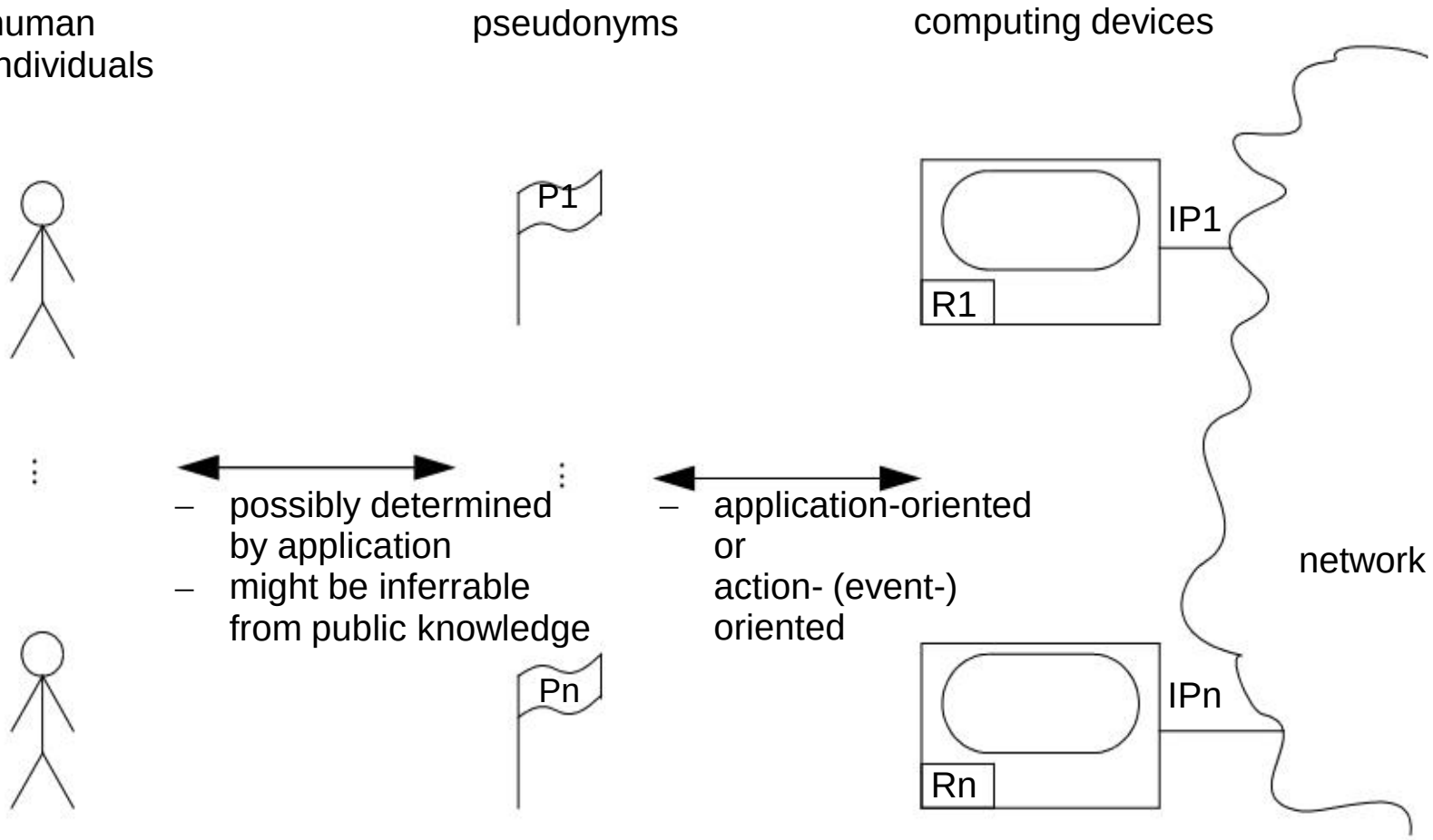
- **activity domain:**
  - one distinguished participant *issues* (sends) digital documents (digitally signed messages) expressing some *obligation* to receivers
  - receivers/holders *present* digital documents as a *credential* (*digital legitimation*) to be redeemed to the distinguished participant
- ***unlinkability property*:**
  knowing the issued documents {(m_1, red_1),…,(m_n, red_n)} and
  seeing a presented modified document (m, red) with a verified signature red,
  a non-designated observer
  cannot "determine" the link
  from the presented document
  to the corresponding issued document
- **mechanism:**
  blind signatures

A classification of pseudonyms

- regarding the *dissemination of knowledge* about the relationship between the pseudonym and the substituted subject, a pseudonym can be seen as
  - *public* (e.g., a phone number of an employee)
  - *confidential* (e.g., a bank account of a citizen)
  - *secret* (also called an *anonym*)

- regarding the intended *potentials for multiple use* and the resulting linkability, there are
  - *subject pseudonyms* for a broad range of activities
  - *role pseudonyms* for specific activities
  - *relationship pseudonyms* for activities addressing specific partners
  - combined *role-relationship pseudonyms* for specific activities addressing specific partners
  - *transaction pseudonyms* (*event pseudonyms*) for *single use* only

Meanings of the notion of “participant” and their relationships



Sufficient randomness

- to achieve the indistinguishability goals of *cryptographic mechanisms*, *sufficient randomness* is needed

- a cryptographic mechanism *superimposes*
  the randomness of a secretly selected key, and possibly further inputs,
  on the returned items of interest such that
    the output items (ciphertexts, exhibits,…)
    again appear to be randomly taken

- making “sufficient randomness” algorithmically available
is an outstanding open problem in computer science

- in fact, precisely defining the notion of “sufficient randomness”
has already turned out to be a great challenge
  that has raised various proposals for an answer

Pseudorandom generator

- is a deterministic polynomial-time algorithm

- stretches a seed, a short and supposedly random input,
into a much larger output sequence
  appearing again to be “sufficiently random”

- delivers outputs that should be *computationally indistinguishable* from
a family of (ideal) uniformly distributed sequences:

  - there is no probabilistic polynomial-time algorithm
    that can distinguish the algorithmic outputs
    from the abstract ideal sequences
    with a non-negligible probability without knowing the seeds

Guidelines for generating and employing pseudorandom sequences

- use some *physical source* for supplying (supposedly) "truly random" *seeds* of short length
- use a *pseudorandom generator* for stretching a supposedly random input seed into a much larger output sequence appearing again to be "sufficiently random"
- design a cryptographic mechanism (for encryption, authentication, etc.)
  - to take a "truly random" input
  - to superimpose the randomness of this input on the returned items (to be proven to comply with pertinent indistinguishability as well)
- for an actual *implementation*, however, replace the (ideal) "truly random" input by an actually available pseudorandom sequence
- verify a compositionality property of the *indistinguishability properties*, to ensure that the replacement does not affect the quality of the returned items

Goals of random input: examples

- to *generate* a *secret key* for some cryptographic mechanism:
  to designate its holder(s) as distinguished from all other participants

- to employ a random input as a *nonce*:
  to mark a message within some cryptographic protocol as unique and personal

- to *pad* a value from some (too small) domain with a random input:
  to define a modified domain sufficiently large to prevent successful guessing

- to *blind* some data with a random input using a reversible algebraic operation:
to present that data to somebody else without revealing the actual value

- most generally, to *randomize* some algorithm of a cryptographic mechanism:
to achieve a wanted indistinguishability property

One-way hash functions

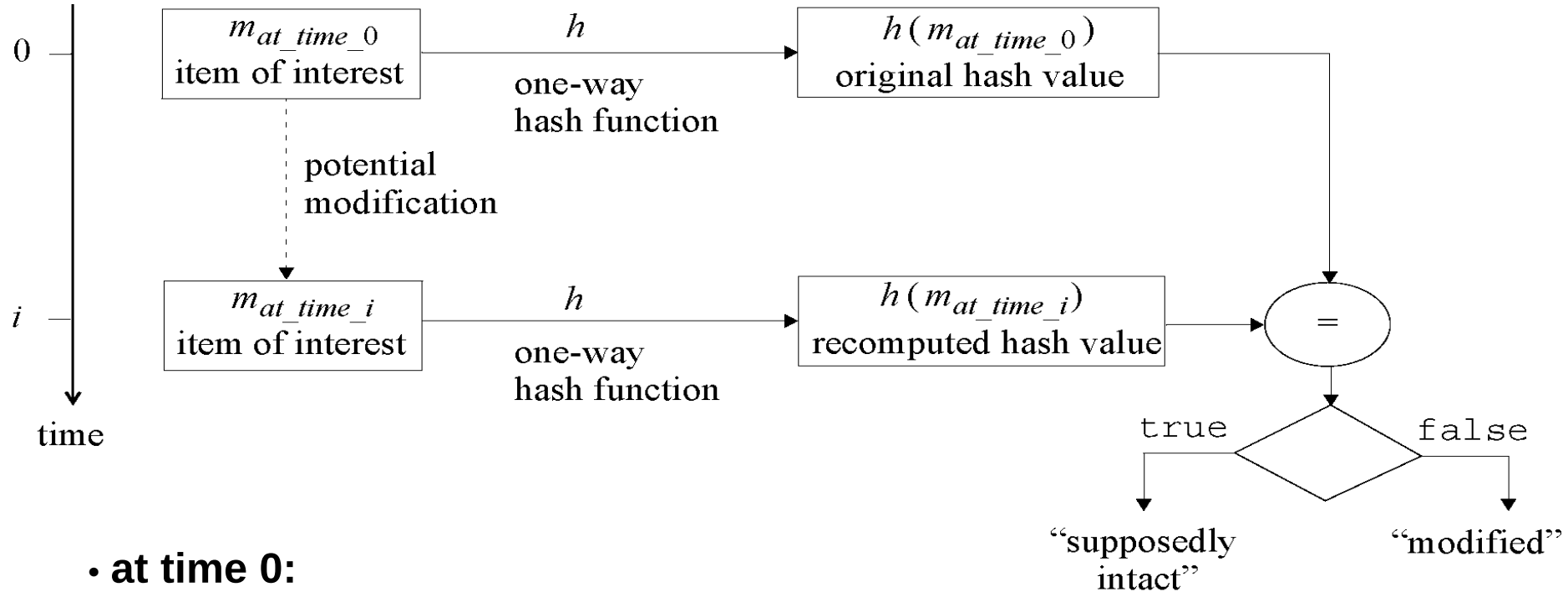
- some item of interest is often represented in a concise, disguised and unforgeable form, called a *fingerprint*, a *digest* or a *hash value*
- *concise*:
  - representation consists of a suitably short bit string of an agreed format
  - a large domain of items is mapped onto a small domain of representations: there must be collisions
- *disguised*:
  a represented item cannot be “determined” from its representation
- *unforgeable*:
  nobody can “determine” a representation of an item without a knowledge of that item
- *collision resistant*:
  nobody can “determine” pairs of items that share a representation

Application: representations with fixed short format

- a cryptographic protocol might demand an argument complying with a fixed short format for further processing, but the items of interest might vary or even be of arbitrary length

- example:
  some authentication protocols digitally sign the representations instead of the represented items

Application: enforcing integrity (detection of modification)



- **at time 0:**
  - map the item onto its representation (original hash value)
  - store the item and its representation in different locations
- **at a later time *i*:**
  - compare the retrieved representation (original hash value) with a recomputed representation of the retrieved item (recomputed hash value)

One-way hash functions: functionality and properties

- function h maps any element m of a (large) domain D (might be infinite) onto a bit string of a (short) fixed length l, i.e., onto an element of $\{0,1\}^l$
- an assigned value $h(m)$ is called the *hash value* of m
- the function h must be efficiently computable, i.e., there is an efficient algorithm H that computes $h(m)$ on input of m
- the *inversion* of h must be computationally infeasible, i.e., the following roughly circumscribed *one-way property* is required:

  for all values $z \in \{0,1\}^l$,
  one cannot "determine" a domain element $m \in D$ such that $h(m) = z$

- regarding the inevitable collisions (for large domain and short length), the function h must be *collision-resistant*

Weak collision-resistance property

- should protect against a fraud
  where a given message *m* is exchanged for another one:

- for all domain elements $m \in D$,
  one cannot “determine” a different domain element $m \in D$
  such that $h(m) = h(m)$

Strong collision-resistance property

- should totally block any attempt at a fraudulent exchange

- one cannot “determine”
  two different domain elements $m \in D$ and $m \in D$
  such that $h(m) = h(m)$

- equivalent to requiring
  that one cannot “determine”
  an element $m \in D$
  that violates the weak version

Timestamps

- sometimes, *integrity* as *temporal correctness* should be supported
- in a proof of authenticity, the receiver should be able to evaluate
  - not only *who* has formed and sent a message
  - but also *when* these two events happened
- to prevent replay attacks or to achieve related goals,
  before authenticating a message, the sender can include a current *timestamp*
- considering the time span between
  - when the message was *formed* and
  - when it was *received*,

  the receiver can decide whether he is willing
  to accept the message as authentic or not
- all participants involved must share *synchronized clocks*;
  the receiver should take tolerable discrepancies in local times into account

Timestamps

- combined *temporal correctness and unforgeability property* is desired:

for all messages *m* with an included timestamp *ts*
and suitably authenticated by the sender,
from the perspective of a receiver,
the actual *forming time* of the message
coincides with the included *timestamp*

- participants might prefer to employ weaker
but more readily manageable means than timestamps

- if only *relative* forming times are important,
the sender might include *serial numbers* (instead of timestamps)

- a receiver not willing to rely on synchronized clocks might
ask a sender to follow a *challenge-response procedure*
in order to obtain evidence for the freshness of a received message

Quality in terms of attacks

- cryptography aims at enabling participants to autonomously enforce their security interests even in the presence of threats
- a *threat* is instantiated by somebody/something performing a specific *attack*
- attack in theoretical investigations:
  an execution of a polynomially time-bounded probabilistic Turing machine
- attack in more practical investigations:
  exploiting a concrete attacking strategy
- *security requirements*:
  to be specified in terms of attacks
- *evaluating* a cryptographic mechanism:
  includes an analysis of the mechanism's robustness against attacks
- *classification framework* for attacks (on encryption mechanisms):
  here, from the point of view of attackers, describing their options for success

A classification framework for attacks against encryption

- ***kind* of success**
  - *exact*: exact new knowledge
  - *probability-theoretic*: improved probability distribution
- ***extent* of success**
  - *universal*: functional equivalence with decryption algorithm
  - *complete*: gain of secret key
  - *message-selective*: plaintexts of selected ciphertexts
  - *message-existential*: plaintext of some ciphertext
- ***target* of attack**
  - affect *human individuals*
  - exploit *computing system*
  - affect individuals and the system in *coordination*

A classification framework for attacks against encryption

- ***time* of attack/attacked *part***
  - subvert *overall system*
  - subvert *key generation*
  - subvert *key distribution*
  - exploit *message transmissions*
- ***method* of attack (against message transmissions)**
  - *passive*: observe messages [ciphertext/plaintext pairs]
  - *active*: observe plaintexts [ciphertexts] of chosen ciphertexts [plaintexts]
- ***planning* of active attack**
  - *non-adaptive*: choose statically at the beginning
  - *adaptive*: choose dynamically depending on progress
- ***expectation* of success**
  - *probability-theoretic*: upper bound for success probability
  - *complexity-theoretic*: lower bound for needed resources
  - *combined*: upper bound for success probability with limited resources

Cryptographic security

- a participant designated to hold some secret or private keys
  must be able to secretly generate, store and use these keys;
  best if the participant controls a *personal tamper-resistant computing device*

- secret and private keys and possibly further items have to appear as random,
and, accordingly, some *source of randomness* should be available;
best possibility being a *truly random* physical source

- items to appear as random must have *sufficient length*
to resist attacks based on *exhaustive search and trials*

- some assistance of a *trusted third party* is normally required

- various further *external participants* contribute
to an application of a cryptographic mechanism;
assigning *trust* to them should be based on
open design and *informational assurances*