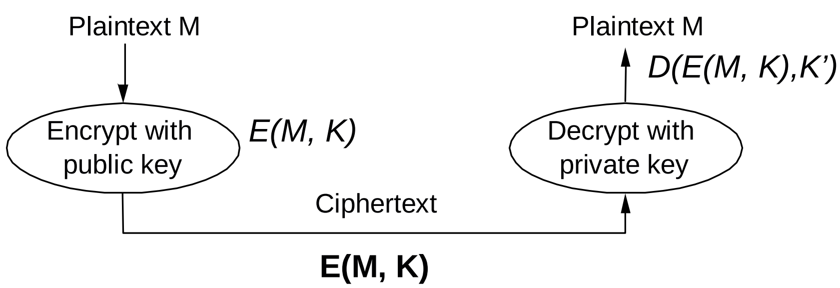
# CSE/EE 461
# Network Security

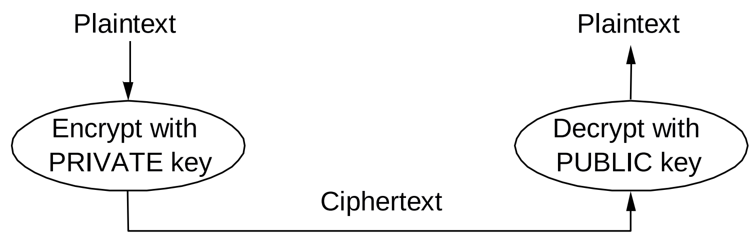
Part 2

## Public Key Functions (RSA)




- Public and private key related mathematically
  - Public key can be published; private is a secret
  - Very Hard to deduce private key from public key

## Authentication with Digital Signature




- Notice that we reversed the role of the keys (and the math just works out) so only one party can send the message but anyone can check it's authenticity

3

## An Example Protocol Built on a Mechanism

- Bob receives the message "HELLO" encrypted with Alice's private key
- Bob decrypts the msg using Alice's public key and sees the word "HELLO"
- Bob concludes that Alice sent the message to Bob.

The protocol defines:

- The messages that go back and forth.
- The way the messages are manipulated.
- The conclusions that can be drawn from the dialog.

Three ways to attack a protocol:

1. Break the key.
2. Lead one or both parties to the wrong conclusions.
3. Infect the implementation to thwart, spoof, or circumvent.

**Bob has drawn the wrong conclusion.**

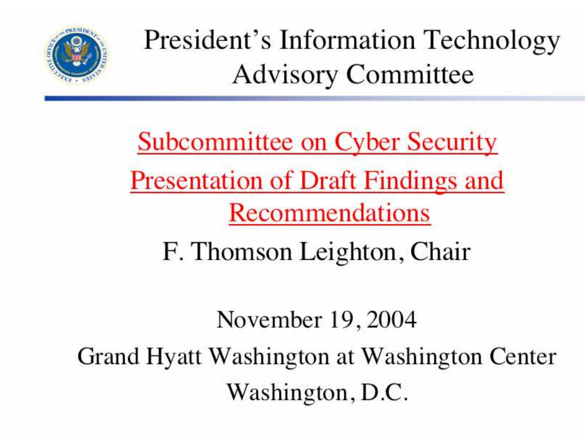
4

## Strong PROTOCOLS vs. Strong ALGORITHMS

- Algorithms let you encode the bits
- Protocols tell you how to decide if the bits are valid
- Looks pretty easy
- But in practice, it's pretty hard
  - The Protocols Must Be Correctly Implemented
    - They Achieve What They are Intended to Achieve
    - ***They Can Not Be Bypassed***
- Most failures come from NOT from attacks on the the algorithm BUT
  - Attacks on the protocol
  - Attacks on the (protocol) implementation
  - Attacks on some other aspects of the target implementation thereby circumventing the protocol itself
    - Can happen when a system is strongly, but not broadly secured.
- In other words, why work hard to break DES when it's so easy to break IE.

5

## Today, the Protocol is not the Weakest Link




6

## Societal Consequences of Information Technology Vulnerabilities (1)

- IT is at the heart of society; IT runs critical infrastructures: electric power grid, financial systems, air traffic control, food distribution, defense networks, etc.
- The use of IT (and the faith in it) has had enormous positive impact on productivity, with tremendous remaining potential (e.g., see PITAC Health Care report).

## Societal Consequences of Information Technology Vulnerabilities (2)

- Ubiquitous interconnection is central to what makes IT important to society.
- *But* ubiquitous interconnection is also a primary source of widespread vulnerability.



## The Problems are Growing at a Dramatic Rate (1)

- The number of new vulnerabilities discovered in software is growing at 140% per year, and is now in excess of 4000 per year (CERT).
- The average time between disclosure of a vulnerability and release of an associated exploit has dropped to 5.8 days (Symantec).
- The percent of PCs infected per month has grown from 1% in 1996 to over 10% in 2003 (ICSA Labs).
- The rate at which new hosts are "zombied" rose from 2,000 per day to 30,000 per day during the first 6 months of 2004 (Symantec).

## The Problems are Growing at a Dramatic Rate (2)

- 92% of organizations experienced "virus disasters" in 2003 (ICSA Labs).
- 83% of financial institutions experienced compromised systems in 2003, more than double the rate in 2002 (Deloitte).
- Hostile (worm) traffic originated from 40% of networks controlled by Fortune 100 companies in 1H04, despite the fact that these companies have taken a variety of protective measures (Symantec).

## The Problems are Growing at a Dramatic Rate (3)

- 17% of 100 companies surveyed reported being the target of cyber extortion (CMU-Information Week)
- The number of unique phishing attacks is doubling every month with 2000 different attacks perpetrated against millions of users in July alone (Anti-Phishing Working Group).
- 1% of US households fell victim to phishing attacks in early 2004, at a cost of over $400M in direct monetary losses (Consumers Union).

The Washington Times
www.washingtontimes.com

**Tenet calls for Internet security**

By Shaun Waterman
UNITED PRESS INTERNATIONAL
Published December 2, 2004

Former CIA Director George J. Tenet yesterday called for new security measures to guard against attacks on the United States that use the Internet, which he called "a potential Achilles' heel."

## Drawing the wrong conclusions

For how to break keys, take a crypto class.

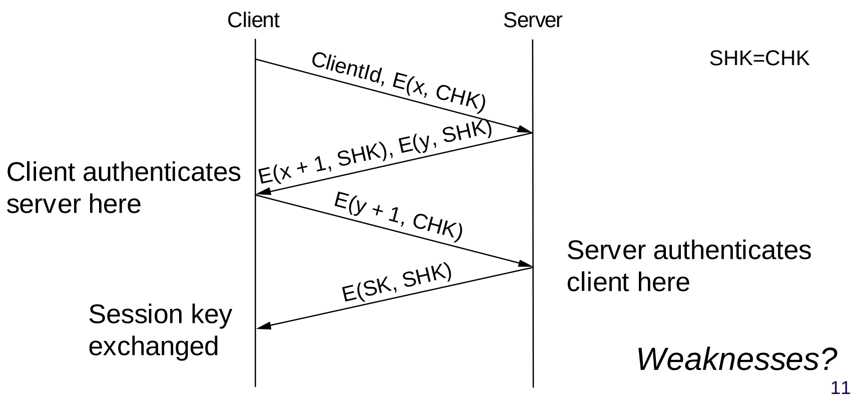
For how to infect a target, take an OS class and read the literature

## Understanding a Protocol and the Conclusions that it Leads To

- What does this protocol achieve?
- Does this protocol need more assumptions than another one?
- Does this protocol do anything unnecessary that could be left out without weakening it?
- Does this protocol encrypt something that could be sent in the clear?

## Authentication Protocols

- Three-way handshake for mutual authentication
  - Client and server share secrets, e.g., login password

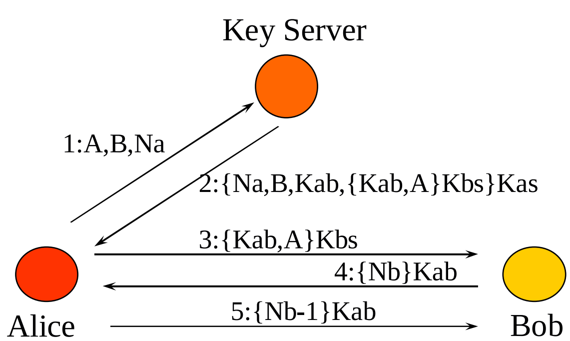



*Weaknesses?*

11

## Another Simple Protocol

- Needham-Schroeder Protocol -- 1978
  - Forms the basis of Kerberos & MS Auth Protocols

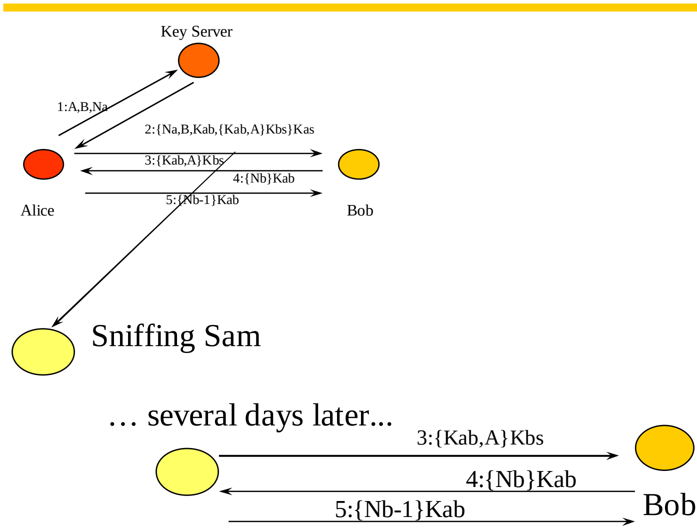



What's the weakness in the protocol?

# Simple Assumptions Can Lead to Weaknesses

- Bob assumes that the key it is being given by Alice is *fresh*.
- With this assumption, Bob believes that Alice is in fact Alice, and will provide Alice with whatever Alice wants.
- No protection against replay
- This doesn't mean the protocol is broken, only that it makes certain assumptions.

# A Replay Attack



## Why did this weakness slip by?

- Eyeball verification is not very effective
- Assumptions are often not explicit
  - eg, the key is fresh
- An attacker will leverage these assumptions to break the protocol
- What's needed is a way to reason about authentication
- Stay tuned for part 3…

## A Logic of Authentication

- Seminal paper published in 1991 SOSP by Burrows, Abadi and Needham
  - BAN Logic
- Simple idea
  - make explicit assumptions in an authentication protocol
  - describe protocol by formal algebra
    - make explicit initial states
    - derive belief relationships through state transitions
    - final state tells us what we can know

## Example Questions

- What does this protocol achieve?
- Does this protocol need more assumptions than another one?
- Does this protocol do anything unnecessary that could be left out without weakening it?
- Does this protocol encrypt something that could be sent in the clear

## Main Principles

- Freshness
  - if you've sent Joe a number (nonce) you've never sent him before, and if you receive back from Joe something that depends on the number, then you ought to believe that Joe's message is fresher than yours
- Private key validity
  - if you believe that you and Joe know K, then you ought to believe that anything you receive encrypted with K comes from Joe.

## More Principles

- Public validity
  - [illegible] that any message you can decrypt with K comes from that any message you can decrypt with K comes from Joe.
- Shared secrets
  - if you believe that only you and Joe know X, then you ought to believe that any encrypted message you receive containing X originally comes from Joe

## Principles are not the same as facts

- Joe could reveal his secret to someone.
- A bad guy could deduce a public/private key.
- A nonce may not actually be so fresh or random.

## The Approach

- Describe the messages sent in a protocol
  - 2: A->B: {A, Kab}Kbs
    - "A sends to B a message containing {A, Kab} encrypted with the private key Kbs, where Kab is a key suggested by some server S."
- Transform each message into a idealized message that can lead to beliefs
  - 2: A->B: {A <-Kab-> B}Kbs
    - "A says to B that Kab is a good key for communicating between A and B according to S."
  - Generally omit cleartext components since they can be forged and can not lead to new beliefs,
- Beliefs yield assertions about the system
  - B believes that S once said that Kab is a good key for communication between A and B.

## Beliefs, Past and Present

- Parties only say what they believe.
- Present
  - begins with the start of the protocol
- Past
  - anything before the present
- If you believe something in the present, then you believe it for the run of the protocol.
- A belief held in the past (before the current run of the protocol) is not necessarily valid in the present.
  - *beware of old beliefs*

## Encrypted Messages

- An encrypted message is a logical state concealed by an encryption key
  - (A <-K->B)Kbs
  - "K is a good key for use between A and B"
- An encrypted message can not be understood by a principal who does not have the key

## The Logic

- A,B,S denote principles
- Kab, Kas, Kbs denote shared (secret) keys
- Ka, Kb, Ks denote public keys
  - 1/Ka, 1/Kb, 1/Ks denote matching private keys
- Na, Nb, Ns denote specific statements
  - eg, nonces, which can be used to establish freshness
- Conjunction (,) is only propositional connective

## Some Constructs

- P |= X
  - P *believes* X
    - P may act as though X were true
- P <| X
  - P *sees* X
    - someone has sent P a message containing X. P may repeat X in other messages
- P |~ X
  - P *once said* X
    - P at one time sent a message containing the statement X. No one knows how long ago.
- P => X
  - P *has jurisdiction* over X
    - P is an authority on X and should be trusted on this matter. Used for delegation, eg, servers that generate keys.
- #(X)
  - The formula X is fresh
    - never been sent before this run of the protocol
- P <-K->Q
  - P and Q may use the shared key K to communicate
    - K will never be discovered by any principal except P or Q.
- K:->P
  - P has K as a public key
    - 1/K (matching private key) will never be discovered by any party but P
- P=X=Q
  - The formula X is a secret known only to P and Q, and possibly to parties trusted by them
    - Only P and Q may use X to prove their identities to one another (eg, a password)
- {X}K
  - X encrypted with the key K
- <X>Y
  - X combined with Y
    - intent is that Y is a secret, eg passwd
    - Y proves the origin of X.

## Message Meaning Rules

- How to derive beliefs from the origins of messages

$$\frac{P \mid= Q <- K -> P,\ P <\mid \{X\}K}{P \mid= Q \mid\sim X}$$

**SHARED KEYS**: If P believes that K is a good key for P and Q, and P sees X encrypted with K, then P believes that Q once said X.

$$\frac{P \mid= K:->Q,\ P<\mid \{X\}1/K}{P \mid= Q \mid\sim X}$$

**PUBLIC KEYS**:If P believes that K is Q's public key, and P sees X encrypted with K's private key, then P believes that Q once said X.

$$\frac{P \mid= Q =Y= P,\ P <\mid <X>Y}{P \mid= Q \mid\sim X}$$

**SHARED SECRETS**: If P believes that Y is a secret shared between Q and P, and P see's <X>Y, then P believes that Q once said X.

## Nonce Verification

- Decryption of a message only says that it was uttered at some point, possibly in the past.
  - does not say if the sender still believes it
    - eg, could be result of a replay

$$\frac{P \mid\equiv \#(X),\ P \mid\equiv Q \mid\sim X}{P \mid\equiv Q \mid\equiv X}$$

If P believes that X was said recently, and that Q said X, then P believes that Q believes X

***This is the only formula that promotes*** |~ to |=.
It reflects essence of challenge/response protocols.
Fresh statement is challenge.
Any message containing challenge is also fresh.

## Jurisdiction Rule

- If P believes that Q has jurisdiction over X, then P trusts Q on the truth of X

$$\frac{P \mid\equiv Q \Rightarrow X,\ P \mid\equiv Q \mid\equiv X}{P \mid\equiv X}$$

This rule gets used a lot when thinking about key servers.

# Some Other Rules

P believes a set of statements iff P believes each individual statement.

$$\frac{P \mid\equiv X,\ P \mid\equiv Y}{P \mid\equiv (X,Y)} \qquad \frac{P \mid\equiv (X,Y)}{P \mid\equiv X} \qquad \frac{P \mid\equiv Q \mid\equiv (X,Y)}{P \mid\equiv X}$$

If a principal sees a formula, then he can see its components (provided keys are known).

$$\frac{P \triangleleft (X,Y)}{P \triangleleft X} \qquad \frac{P \triangleleft \langle X \rangle Y}{P \triangleleft X} \qquad \frac{P \mid\equiv Q \overset{K}{\leftrightarrow} P,\ P \triangleleft \{X\}K}{P \triangleleft X}$$

If a part of a formula is known to be fresh, the entire formula is fresh (freshness distributes)

$$\frac{P \mid\equiv \#(X)}{P \mid\equiv \#(X,Y)}$$

# Idealized Protocols

- Transform a message step into an idealized protocol step
  - include only information that contributes to the beliefs of the recipient
    - eliminate hints
  - make explicit beliefs
- Example
  - Protocol Step
    - A->B : {A, Kab}Kbs
    - intended to tell B, who knows Kbs, that Kab is a good key for communicating with A.
  - Idealized Protocol Step
    - A ->B : {A <-Kab-> B} Kbs
    - Allows us to deduce
      - B <| {A <-Kab->B}Kbs
    - If B |= B<-Kbs->S and S=>Kab, then
      - B |= S |~ {A<-Kab->B}
      - "if B believes that Kbs is a good key for B and S, and S has jurisdiction over Kab, then B believes that S once said that Kab is a good key for use between A and B."
      - is it still a good key?
        - » who knows??

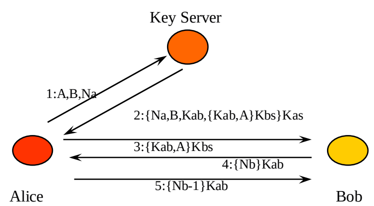
## How to reason about a protocol

- Derive idealized protocol from original one
- Write assumptions about initial system state
- Attach logical formulas to the statements of the protocol
- Attach assertions to the statements of the protocol
- Apply postulates to assumptions and assertions to derive new beliefs
  - first assertion contains assumptions
  - last assertion contains the conclusions.
- Repeat until convinced.

## Formal Goals of Authentication

- Initial assumptions state what keys are shared between principles, which principles are trusted, and which statements are fresh
- Authentication then means
  - A |= A<-K->B
  - B |= A<-K->B
- Could also mean in addition
  - A |= B |= A <-K->B
  - B |= A |= A<-K->B

# Needham and Schroeder Revisited




1: A->S: A, B, Na
2: S->A: {Na, B, Kab}, {Kab, A}Kbs}}Kbs
3: A->B:{Kab, A}Kbs
4: B->A: {Nb}Kab
5: A->B: {Nb-1}Kab

# The Idealized Protocol

1: A->S: A, B, Na
2: S->A: {Na, B, Kab}, {Kab, A}Kbs}}Kbs
3: A->B:{Kab, A}Kbs
4: B->A: {Nb}Kab
5: A->B: {Nb-1}Kab

The actual protocol
(messages sent)

1: !! Contributes nothing !!
2: S->A: {
Na, (A<-Kab->B),
#(A<-Kab->B),
{A<-Kab->B}Kbs
}Kas
3:A->B: {A<-Kab->B}Kbs
4:B->A: {Nb, (A<-Kab->B)}Kab
5:A->B: {Nb, (A<-Kab->B)}Kab

The idealized protocol
(statements made)

## Analyzing the protocol: Initial Assumptions

A |= A<-Kas->S
S |= A <-Kas->S
S |= A <-Kab->B
A |= (S => A<-K-> B)
~~A |= (S => #(A<-K->B))~~
A |= #(Na)
S |= #(A <-Kab-> B)

B |= B<-Kbs->S
S |= B <-Kbs->S
} Initial keys

B |= (S => A<-K-> B)
} Key server

B |= #(Nb)
B |= #(A<-K->B)
} Freshness

This is that hidden initial assumption

## What the idealized protocol says

2: S->A: {Na, (A<-Kab->B), #(A<-Kab->B), {A<-Kab>B}Kbs}Kas

First,

A <| {Na, (A<-Kab->B), #(A<-Kab->B), {A<-Kab>B}Kbs}Kas

which A decrypts using Kas. Since A knows Na to be fresh, we can apply:

$$\frac{P \mid\equiv \#(X),\ P \mid\equiv Q \mid\sim X}{P \mid\equiv Q \mid\equiv X} \quad \textit{(nonce verification)}$$

Leading to

A |= S |= A <-Kab-> B (good key)
A |= S |= #(A <-Kab -> B) (fresh key)

Applying

$$\frac{P \mid\equiv Q \Rightarrow X,\ P \mid\equiv Q \mid\equiv X}{P \mid\equiv X} \quad \textit{(jurisdiction)}$$

Gives

A |= A <-Kab-> B
A |= #(A <-Kab -> B)

# A repeats what it sees

A <| {A<-Kab->B}Kbs

3:A->B: {A<-Kab->B}Kbs

We apply the message meaning postulate:

$$\frac{P \mid\equiv Q \overset{K}{\leftrightarrow} P,\ P \triangleleft \{X\}_K}{P \mid\equiv Q \mid\sim X}$$

P |= Q <- K -> P, P <| {X}K / P |= Q |~ X

SHARED KEYS: If P believes that K is a good key for P and Q, and P sees X encrypted with K, then P believes that Q once said X.

To obtain

B |= S |~ A <-Kab-> B

In order to obtain B|= A<-Kab->B, we need to rely on nonce verification (recall, only N.V. promotes |~ to |=)

# Relying on an assumption now explicit

Since we assumed initially that

B |= #(A<-K->B)

We can promote

B |= S |~ A <-Kab-> B

to

B|= A <-Kab->B

using

$$\frac{P \mid\equiv \#(X),\ P \mid\equiv Q \mid\sim X}{P \mid\equiv Q \mid\equiv X}$$

P |= #(X), P |= Q |~ X / P |= Q |= X *(nonce verification)*

## Getting A to believe that B is on board

4:B->A: {Nb, (A<-Kab->B)}Kab

Since

A <| {Nb, (A<-Kab->B)}Kab}

and

A |= A <-Kab->B

then

A |= B |= (A <-Kab->B)

(since B said it, B believes it)

## The final step

5:A->B: {Nb, (A<-Kab->B)}Kab

Allows

B |= A |~ {Nb, (A<-Kab->B)}

Freshness distributes, so we can apply

$$\frac{P \mid\equiv \#(X),\ P \mid\equiv Q \mid\sim X}{P \mid\equiv Q \mid\equiv X} \quad \textit{nonce verification}$$

to get

B |= A |= A <-Kab->B

## At the end we have

A |= A <-Kab->B
B |= A <-Kab->B
A |= B |= A <-Kab> B
B |= A |= A <-Kab> B

which is the goal of an authentication protocol.

Had we not made the freshness assumption, we would have been stuck and could not have gotten here.

## Conclusions

- We need to make an awful lot of assumptions in designing authentication protocols.
- The assumptions are there, whether you state them or not.
- Only by stating them explicitly can we enter into a final acceptable state of mutual authentication.

## Authenticity and Integrity

- Sometimes we care about knowing messages authentic, but don't care about privacy.
- If only sender and receiver knew the keys we would be done … but that's often not the case
  - A pair of keys for each pair of communicating parties?
- In public key (RSA) systems the "encryption" key is potentially known by everyone
  - anyone could have sent us a confidential message by encrypting with our public key

43

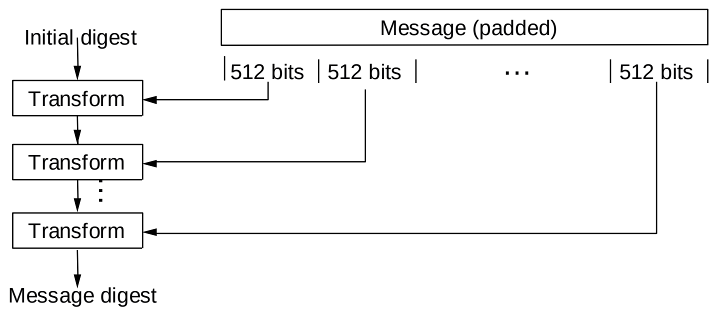
## A Faster "RSA Signature"

- Encryption can be expensive, e.g., RSA 1Kbps
- To speed up, let's sign just the checksum instead!
  - Check that the encrypted bit is a signature of the checksum
- Problem: Easy to alter data without altering checksum
- Answer: Cryptographically strong "checksums" called message digests where it's computationally difficult to choose data with a given checksum
  - But they still run much more quickly than encryption
  - MD5 (128 bits) is the most common example

44

## Message Digests (MD5, SHA)

- Act as a cryptographic checksum or hash
  - Typically small compared to message (MD5 128 bits)
  - “One-way”: infeasible to find two messages with same digest




45

## Cryptography in Protocols

- These techniques can be applied at different levels:
  - IP packets (IPSEC)
  - Web transfers or other transports (SSL/TLS, Secure HTTP)
  - Email (PGP)

- Next time ..

46

## Key Concepts

- Privacy, integrity, and authenticity
- Cryptographic mechanisms are used to support these properties: private key, public key and digests

47