CIS 551 / TCOM 401

Computer and Network Security

Spring 2009

Lecture 17

Announcements

- Plan for Today:
  - RSA continued
  - Dolev-Yao model of attackers
  - Authentication protocols

- Project 3 is due 6 April 2009 at 11:59 pm
  - Handout for SDES available by request…
  - Please read the project description *BEFORE* looking at the code

- Midterm 2 is Thursday, April 2nd (next week!) in class
- Final exam has been scheduled:
  Friday, May 8, 2009
  9:00am – 11:00am, Moore 216

RSA at a High Level

- Public and private key are derived from secret prime numbers
  - Keys are typically ≥ 1024 bits
- Plaintext message (a sequence of bits)
  - Treated as a (large!) binary number
- Encryption is modular exponentiation
- To break the encryption, conjectured that one must be able to factor large numbers
  - Not known to be in P (polynomial time algorithms)
  - Is known to be in BQP (bounded-error, quantum polynomial time – Shor's algorithm)

RSA Key Generation

- Choose large, distinct primes p and q.
  - Should be roughly equal length (in bits)
- Let $n = p*q$
- Choose a random encryption exponent e
  - With requirement: e and (p-1)*(q-1) are relatively prime.
- Derive the decryption exponent d
  - $d = e^{-1} \bmod ((p-1)*(q-1))$
  - d is e's inverse mod ((p-1)*(q-1))
- Public key: $K = (e,n)$ pair of e and n
- Private key: $k = (d,n)$
- Discard primes p and q (they're not needed anymore)

RSA Encryption and Decryption

- Message: m
- Assume m < n
  - If not, break up message into smaller chunks
  - Good choice: largest power of 2 smaller than n

- Encryption: $E((e,n), m) = m^e \bmod n$
- Decryption: $D((d,n), c) = c^d \bmod n$

Example RSA

- Choose p = 47, q = 71
- n = p * q = 3337
- (p-1)*(q-1) = 3220
- Choose e relatively prime with 3220: e = 79
  - Public key is (79, 3337)
- Find d = 79^{-1} mod 3220 = 1019
  - Private key is (1019, 3337)
- To encrypt m = 688232687966683
  - Break into chunks < 3337
  - 688 232 687 966 683
- Encrypt: E((79, 3337), 688) = 688^{79} mod 3337 = 1570
- Decrypt: D((1019, 3337), 1570) = 1570^{1019} mod 3337 = 688

Euler's *totient* function: $\phi(n)$

- $\phi(n)$ is the number of positive integers less than n that are relatively prime to n
  - $\phi(12) = 4$
  - Relative primes of 12 (less than 12): {1, 5, 7, 11}

- For p a prime, $\phi(p) = p-1$. Why?
- For p,q two distinct primes, $\phi(p*q) = (p-1)*(q-1)$
  - There's p*q-1 numbers less than p*q
  - Factors of p*q =
    - {1*p, 2*p, …, q*p} for a total of q of them
    - {1*q, 2*q, …, p*q} for another p of them
    - No other numbers
    - $\phi(p*q) = (p*q) - (p + q - 1) = pq - p - q + 1 = (p-1)*(q-1)$

q many multiples of p

All #s ≤ p*q

don't double count p*q

p many multiples of q

Fermat's Little Theorem

- Generalized by Euler.
- Theorem: If p is a prime then $a^p \equiv a \bmod p$.
- Corollary: If $\gcd(a,n) = 1$ then $a^{\phi(n)} \equiv 1 \bmod n$.
- Easy to compute $a^{-1} \bmod n$
  - $a^{-1} \bmod n = a^{\phi(n)-1} \bmod n$
  - Why? $a * a^{\phi(n)-1} \bmod n$
    $= a^{\phi(n)-1+1} \bmod n$
    $= a^{\phi(n)} \bmod n$
    $\equiv 1 \bmod n$

Chinese Remainder Theorem

- (Or, enough of it for our purposes…)

- Suppose:
  - p and q are relatively prime
  - $a \equiv b \pmod{p}$
  - $a \equiv b \pmod{q}$
- Then: $a \equiv b \pmod{p*q}$

- Proof:
  - p divides (a-b) (because a mod p = b mod p)
  - q divides (a-b)
  - Since p, q are relatively prime, p*q divides (a-b)
  - But that is the same as: $a \equiv b \pmod{p*q}$

Proof that D inverts E

$c^d \bmod n$

$= (m^e)^d \bmod n$ (definition of c)

$= m^{ed} \bmod n$ (arithmetic)

$= m^{k*(p-1)*(q-1) + 1} \bmod n$ (d inverts e mod $\phi(n)$)

$= m*m^{k*(p-1)*(q-1)} \bmod n$ (arithmetic)

$= m \bmod n$ (C. R. theorem)

$= m$ ($m < n$)

$e*d \equiv 1 \bmod (p-1)*(q-1)$

Finished Proof

- Note: $m^{p-1} \equiv 1 \mod p$ (if p doesn't divide m)
  - Why? Fermat's little theorem.
- Same argument yields: $m^{q-1} \equiv 1 \mod q$
- Implies: $m^{k*\phi(n)+1} \equiv m \mod p$
- And $m^{k*\phi(n)+1} \equiv m \mod q$
- Chinese Remainder Theorem implies:
  $m^{k*\phi(n)+1} \equiv m \mod n$
- Note: if p (or q) divides m, then $m^x \equiv 0 \mod n$
  - Since $m < n$ we must have $m = 0$.

How to Generate Prime Numbers

- Many strategies, but *Rabin-Miller* primality test is often used in practice.
  - $a^{p-1} \equiv 1 \bmod p$
- Efficiently checkable test that, with probability ¾, verifies that a number p is prime.
  - Iterate the Rabin-Miller primality test t times.
  - Probability that a composite number will slip through the test is $(1/4)^t$
  - These are worst-case assumptions.
- In practice (takes several seconds to find a 512 bit prime):
  1. Generate a random n-bit number, p
  2. Set the high and low bits to 1 (to ensure it is the right number of bits and odd)
  3. Check that p isn't divisible by any "small" primes 3,5,7,…,<2000
  4. Perform the Rabin-Miller test at least 5 times.

Rabin-Miller Primality Test

- Is n prime?
- Write n as $n = (2^r)*s + 1$
- Pick random number a, with $1 \le a \le n - 1$
- If
  - $a^s \equiv 1 \bmod n$ and
  - for all j in {0 … r-1}, $a^{2^j s} \equiv -1 \bmod n$
- Then return composite
- Else return probably prime

General Definition of “Protocol”

- A *protocol* is a multi-party algorithm
  - A sequence of steps that precisely specify the actions required of the parties in order to achieve a specified objective.

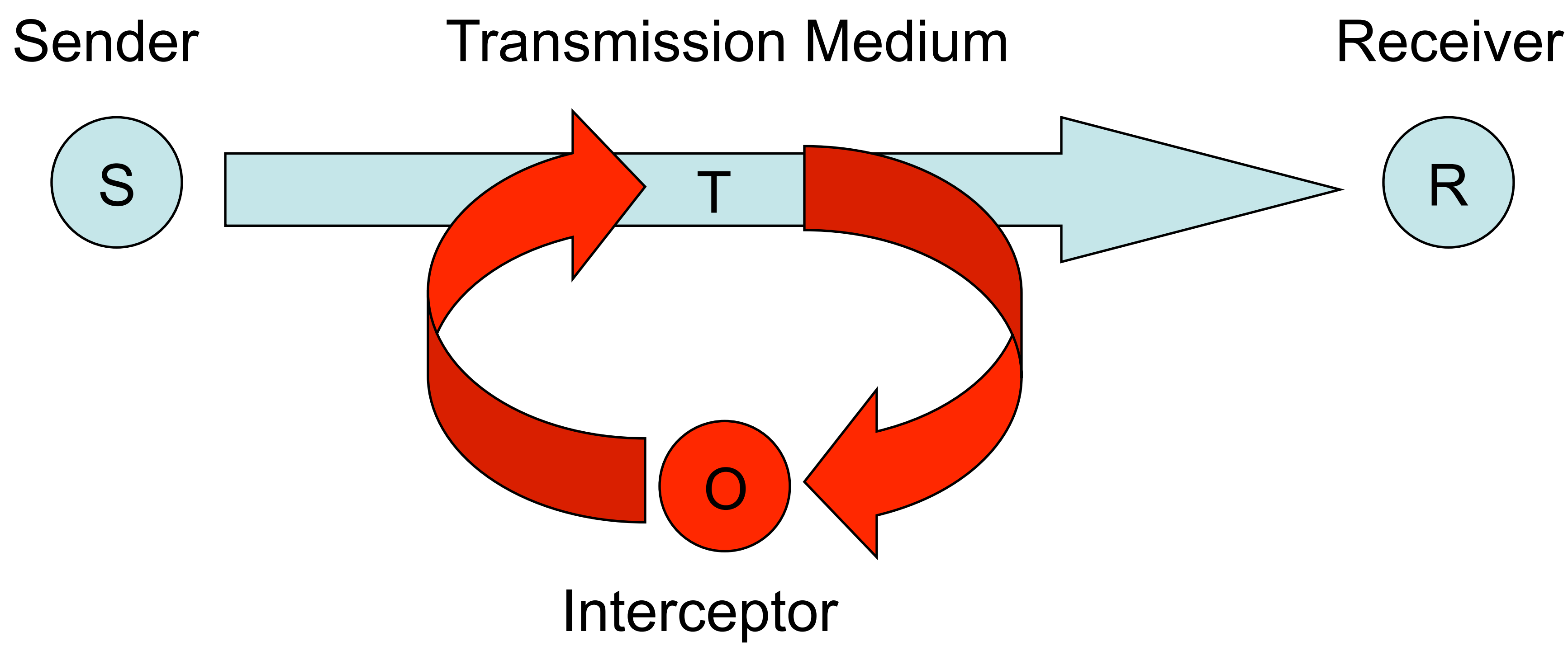
- Important that there are multiple participants
- Typically a situation of heterogeneous trust
  - Alice may not trust Bart
  - Bart may not trust the network

Characteristics of Protocols

- Every participant must know the protocol and the steps in advance.
- Every participant must agree to follow the protocol
  - *Honest participants*
- Big problem: How to deal with bad participants?

Cryptographic Protocols

- Consider communication over a network…
- What is the threat model?
  - What are the vulnerabilities?



What Can the Attacker Do?

- Intercept them (confidentiality)
- Modify them (integrity)
- Fabricate other messages (integrity)
- Replay them (integrity)

- Block the messages (availability)
- Delay the messages (availability)
- Cut the wire (availability)
- Flood the network (availability)

Dolev-Yao Model

- Simplifies reasoning about protocols
  - doesn't require reduction to computational complexity
- Treat cryptographic operations as "black box"
- Given a message M = (c1,c2,c3,…) attacker can deconstruct message into components c1 c2 c3
- Given a collection of components c1, c2, c3, … attacker can forge message using a subset of the components (c1,c2,c3)
- Given an encrypted object K{c}, attacker can learn c only if attacker knows decryption key corresponding to K
- Attacker can encrypt components by using:
  - fresh keys, or
  - keys they have learned during the attack

Formal Dolev-Yao Model

- A message is a finite sequence of :
  - Atomic strings, nonces, Keys (public or private), Encrypted Submessages

  $M ::= a \mid n \mid K \mid k \mid K\{M\} \mid k\{M\} \mid M,M$

- The attacker's (or observer's) state is a set S of messages:
  - The set of all message & message components that the attacker has seen -- the attacker's "knowledge"
  - Seeing a new message sent by an honest participant adds the new message components to the attacker's knowledge
  - If $M_1, M_2 \in S$ then $M_1 \in S$ and $M_2 \in S$
  - If $K_A\{M\} \in S$ and $K_A \in S$ then $M \in S$
  - If $K_A\{M\} \in S$ and $k_A \in S$ then $M \in S$
  - If $M \in S$ and $K \in S$ then $K\{M\} \in S$
  - If $M \in S$ and $k \in S$ then $k\{M\} \in S$
  - If k is a “fresh” key, then $k \in S$

S closed under these operations

Using the Dolev-Yao model

- Given a description of a protocol:
  - Sequence of messages to be exchanged among honest parties.
- "Simulate" an attacked version of the protocol:
  - At each step, the attacker's knowledge state is the (closure of the) knowledge of the prior state plus the new message
  - An active attacker can create (and insert into the communication stream) any message M composed from the knowledge state S:
    - $M = M_1, M_2, \ldots, M_n$ such that $M_i \in S$
- See if the "attacked" protocol leads to any bad state
  - Example: if K is supposed to be kept secret but $K \in S$ at some point, the attacker has learned the key.

Authentication

- For honest parties, the claimant A is able to authenticate itself to the verifier B. That is, B will complete the protocol having accepted A's identity.

Alice

Bart

Shared-Key Authentication

Alice

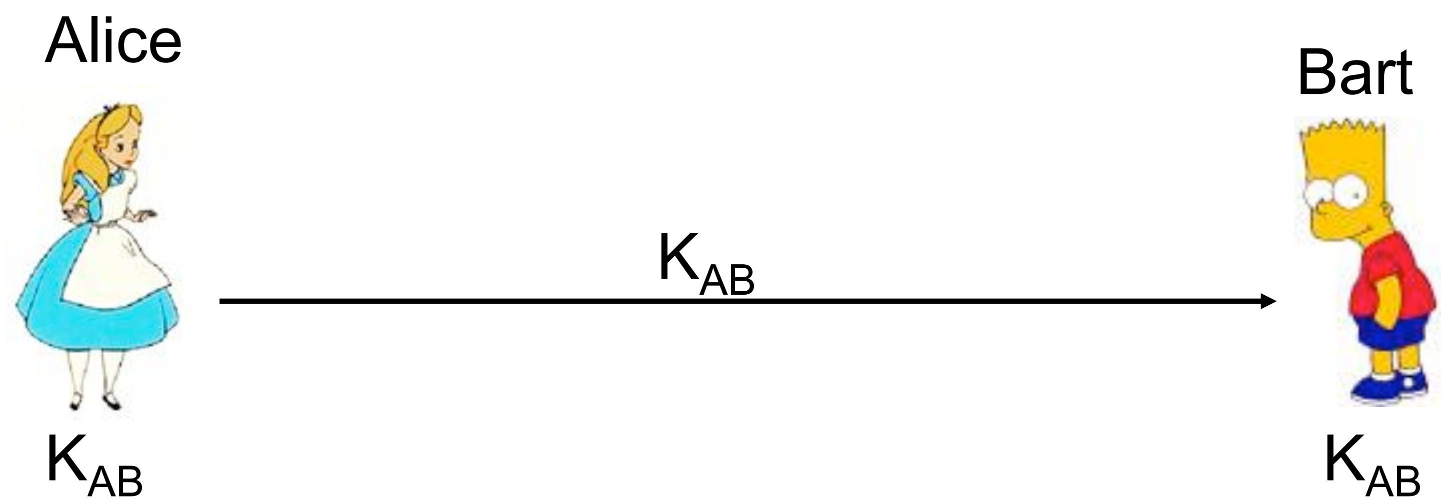
K_{AB}

Bart

K_{AB}

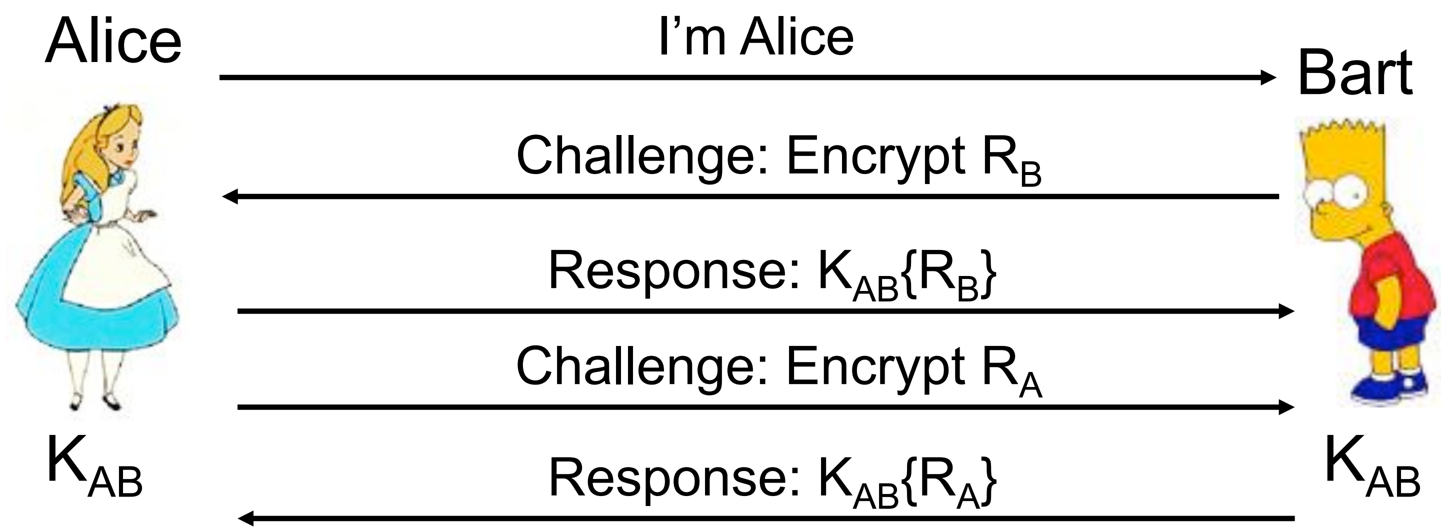
- Assume Alice & Bart already share a key K_{AB}.
  - The key might have been decided upon in person or obtained from a trusted 3rd party.
- Alice & Bart now want to communicate over a network, but first wish to authenticate to each other

Solution 1: Weak Authentication



- Alice sends Bart K_{AB}.
  - K_{AB} acts as a password.
- The secret (key) is revealed to passive observers.
- Only works one-way.
  - Alice doesn't know she's talking to Bart.

Solution 2: Strong Authentication

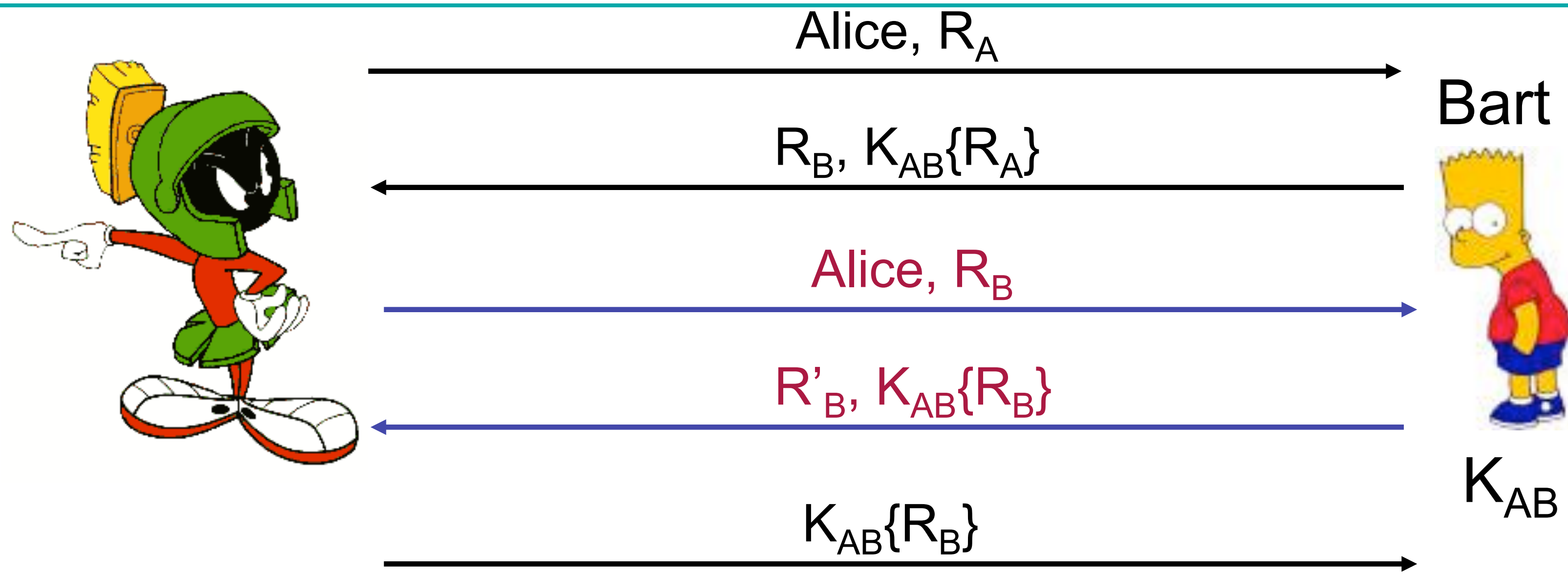


- Protocol doesn't reveal the secret.
- *Challenge/Response*
  - Bart requests proof that Alice knows the secret
  - Alice requires proof from Bart
  - R_A and R_B are randomly generated numbers

(Flawed) Optimized Version

- Why not send more information in each message?
- This seems like a simple optimization.
- But, it’s broken… how?

Attack: Marvin can Masquerade as Alice



- Marvin pretends to take the role of Alice in two runs of the protocol.
  - Tricks Bart into doing Alice's part of the challenge!
  - Interleaves two instances of the same protocol.

Lessons

- Protocol design is tricky and subtle
  - "Optimizations" aren't necessarily good
- Need to worry about:
  - Multiple instances of the same protocol running in parallel
  - Intruders that play by the rules, mostly
- General principle:
  - Don't do anything more than necessary until confidence is built.
  - Initiator should prove identity *before* responder takes action (like encryption)

Threats

- *Transferability:* B cannot reuse an identification exchange with A to successfully impersonate A to a third party C.
- *Impersonation:* The probability is negligible that a party C distinct from A can carry out the protocol in the role of A and cause B to accept it as having A's identity.

Assumptions

- A large number of previous authentications between A and B may have been observed.
- The adversary C has participated in previous protocol executions with A and/or B.
- Multiple instances of the protocol, possibly instantiated by C, may be run simultaneously.

Primary Attacks

- Replay.
  - Reusing messages (or parts of messages) inappropriately
- Interleaving.
  - Mixing messages from different runs of the protocol.
- Reflection.
  - Sending a message intended for destination A to B instead.
- Chosen plaintext.
  - Choosing the data to be encrypted
- Forced delay.
  - Denial of service attack -- taking a long time to respond
  - Not captured by Dolev Yao model

Primary Controls

- Replay:
  - use of challenge-response techniques
  - embed target identity in response.
- Interleaving
  - link messages in a session with chained *nonces*.
- Reflection:
  - embed identifier of target party in challenge response
  - use asymmetric message formats
  - use asymmetric keys.
- Chosen text:
  - embed self-chosen random numbers ("confounders") in responses
  - use "zero knowledge" techniques.
- Forced delays:
  - use nonces with short timeouts
  - use timestamps in addition to other techniques.

Replay

- *Replay*: the threat in which a transmission is observed by an eavesdropper who subsequently reuses it as part of a protocol, possibly to impersonate the original sender.
  - Example: Monitor the first part of a telnet session to obtain a sequence of transmissions sufficient to get a log-in.
- Three strategies for defeating replay attacks
  - Nonces
  - Timestamps
  - Sequence numbers.

Nonces: Random Numbers

- *Nonce*: A number chosen at random from a range of possible values.
  – Each generated nonce is valid *only once*.
- In a challenge-response protocol nonces are used as follows.
  – The verifier chooses a (new) random number and provides it to the claimant.
  – The claimant performs an operation on it showing knowledge of a secret.
  – This information is bound inseparably to the random number and returned to the verifier for examination.
  – A timeout period is used to ensure "freshness".

Time Stamps

- The claimant sends a message with a timestamp.
- The verifier checks that it falls within an acceptance window of time.
- The last timestamp received is held, and identification requests with older timestamps are ignored.
- Good only if clock synchronization is close enough for acceptance window.

Sequence Numbers

- Sequence numbers provide a sequential or monotonic counter on messages.
- If a message is replayed and the original message was received, the replay will have an old or too-small sequence number and be discarded.
- Cannot detect forced delay.
- Difficult to maintain when there are system failures.