User-side Forward-dating Attack on Time-stamping Protocol

IWAP 2004

Oct 4th, 2004

Shin'ichiro Matsuo and Hiroaki Oguro
NTT DATA Corporation

Outline

- Overview of time-stamping protocol
- Attacks on time-stamping protocol
  - Back-dating attacks
  - Forward-dating attacks
- User-side Forward-dating attack
  - Definition
  - Adversary models
- Countermeasures for each adversary model
  - Easy solutions for stand alone adversary
  - New time-stamping protocol secure against an adversary colluding wit TSA
- Analysis
- Conclusion

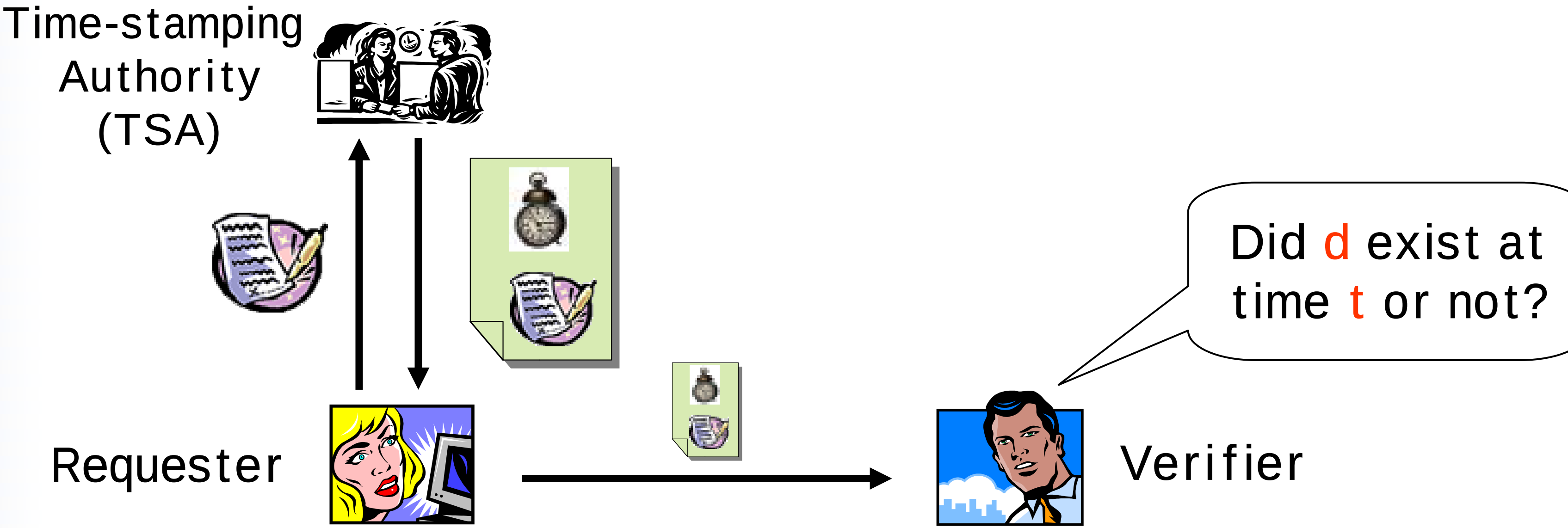
Background of this research

- Time-stamping services are widely organized to certify time of existence of certain document.
- Some secure protocols are proposed to realize such services.
  - Simple protocol [ACPZ01]
  - Linking protocol [HS91]
- There are many researches on security analysis against time-stamping protocol
  - Back-dating
  - Forward-dating by time-stamping authority [Just98]
- We focus on forward-dating attack by a malicious user.
  - Proposing models and countermeasure

Overview of Time-stamping Protocol

me-stamping protocol certifies a document d existed at certain time t.

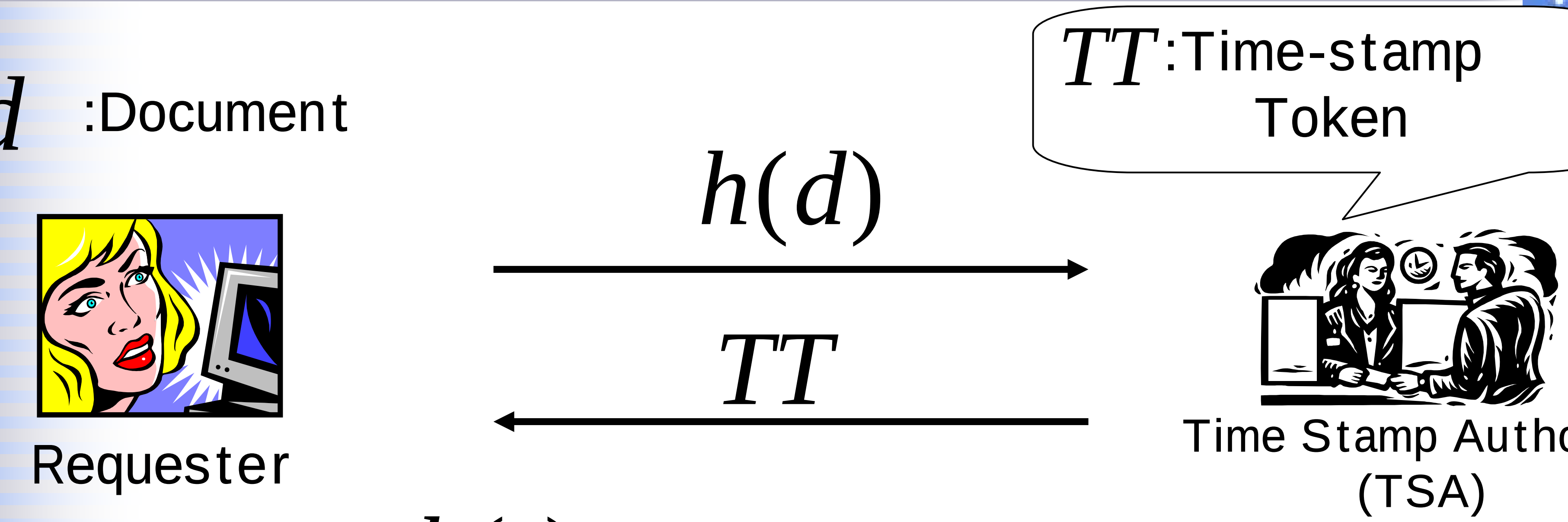
yer: Requester, Time-stamping Authority, Verifier



Application:

- Notary service
- Proving time of patent application (Which is earlier invention?)
- Extending valid period of digital signature …

suing sub-protocol of standard time-stamping protocol



d :Document

TT:Time-stamp Token

$h(d)$

TT

Requester

Time Stamp Autho
(TSA)

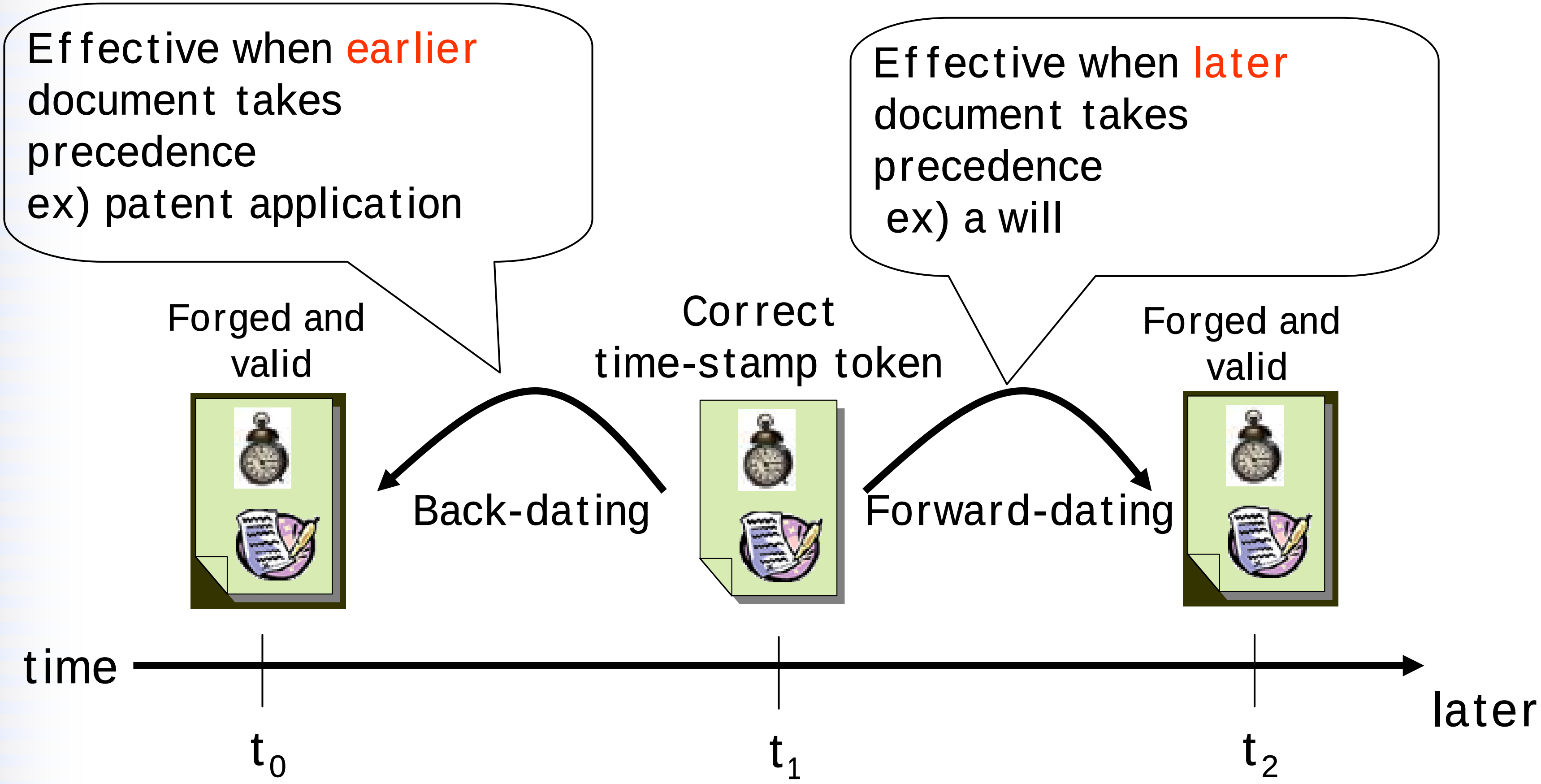
$h(\bullet)$: One-way Permutation

- Simple Protocol (Using digital signature)
- Linking Protocol (Using hash chain)

Supposed attacks on time-stamping protocol

Two major types of attacks

- Back-dating
- Forward-dating



Effective when earlier document takes precedence
ex) patent application

Effective when later document takes precedence
ex) a will

Forged and valid

Back-dating

Correct time-stamp token

Forward-dating

Forged and valid

time

later

t_0

t_1

t_2

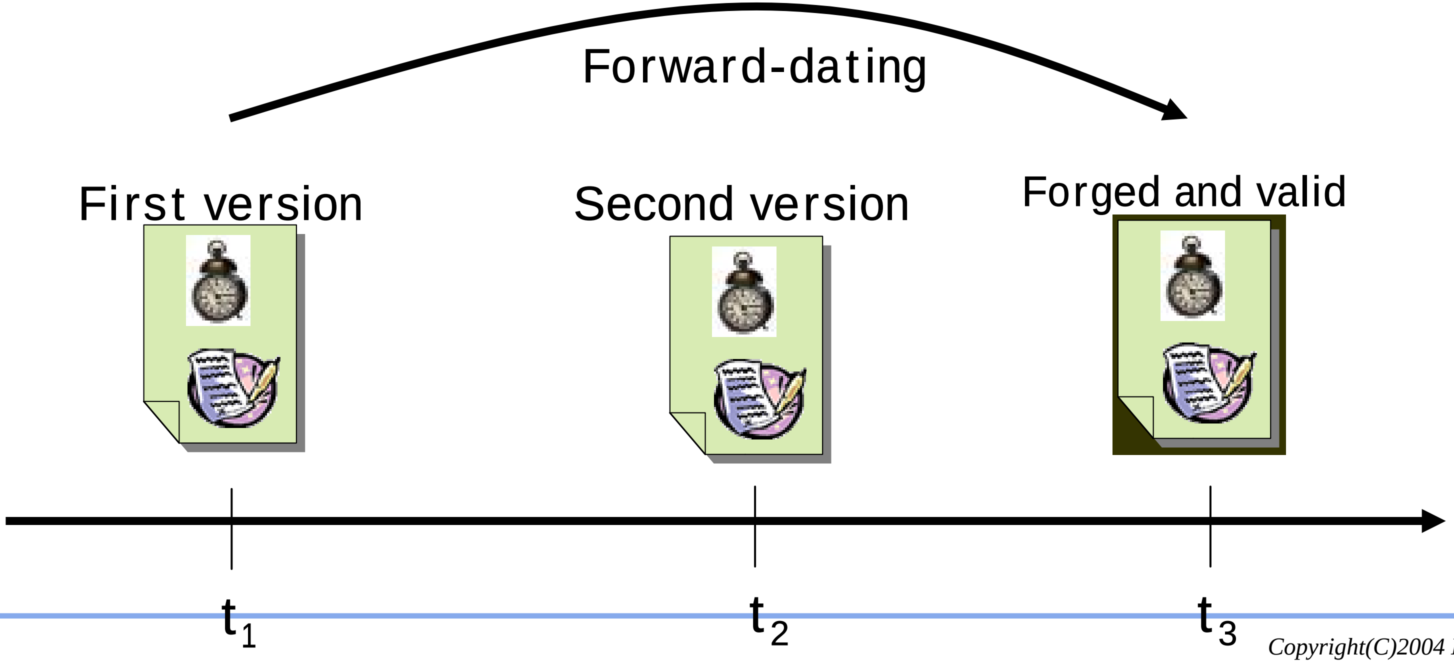
In the case of a will...

At first, original time-stamp requester creates the first version of a w

. Then she update the will to second version.

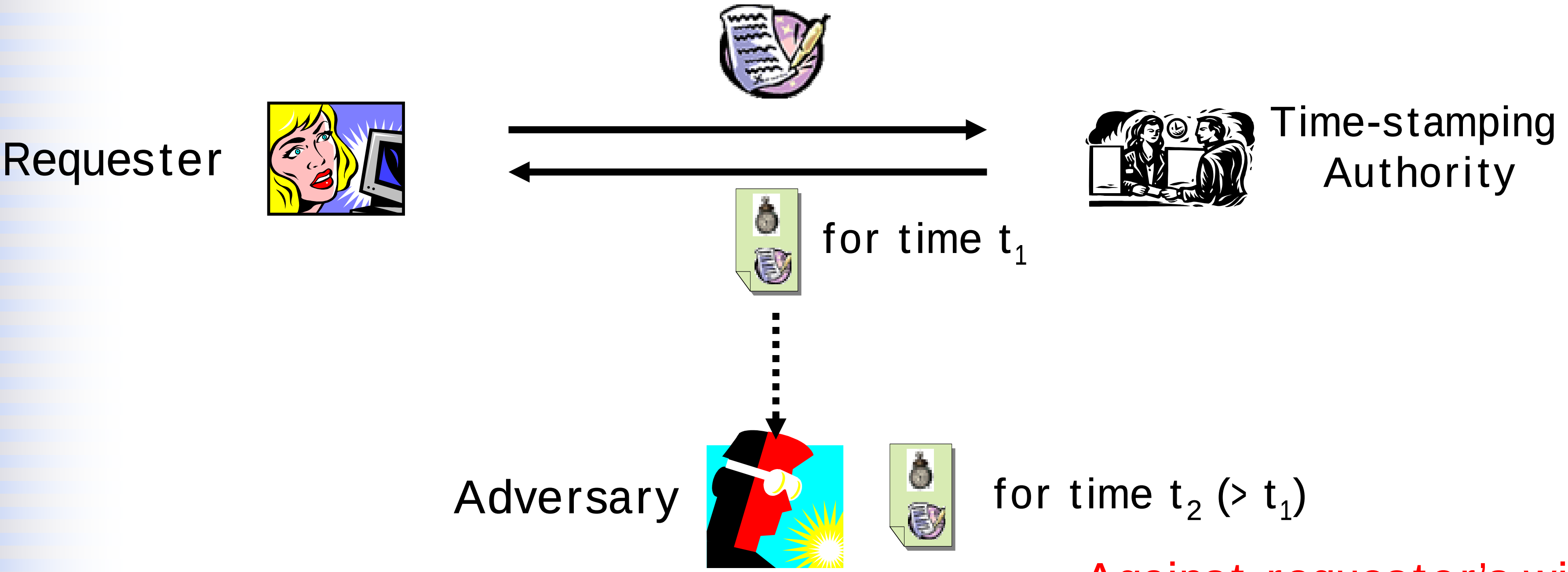
. The second version is worse than the first version for the adversary.

. The adversary intends to re-validate the first version by obtainin time-stamp token of the first version for later time.



User-side Forward-dating Attack

- Existing researches on forward-dating attack focus on the attack by only time-stamp authority. [Just98]
- We focus on the same attack originated by a malicious user.



Requester

Time-stamping Authority

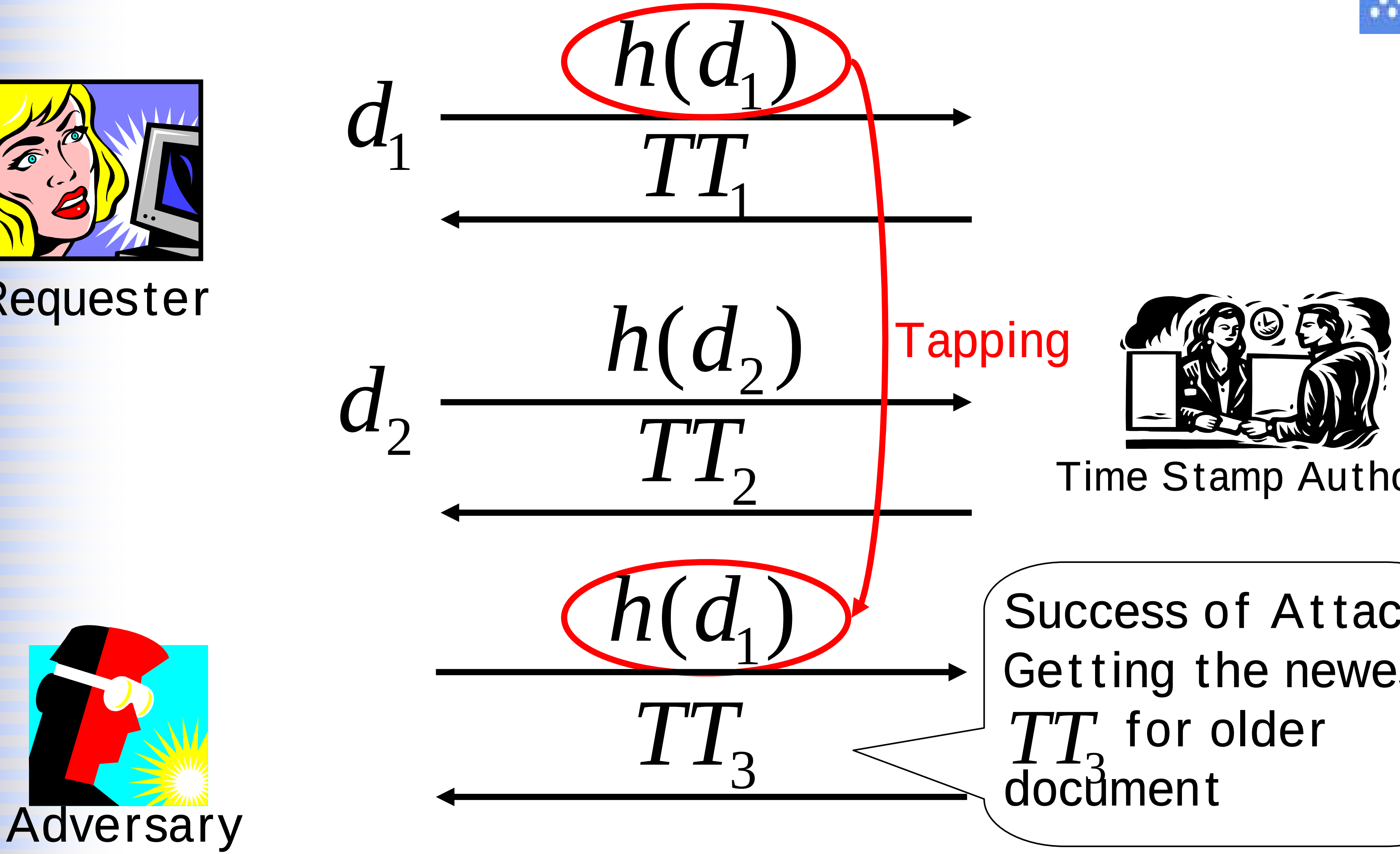
for time t_1

Adversary

for time t_2 (> t_1)

Against requester's wi

An example of this attack



Requester

d_1

$h(d_1)$

TT_1

d_2

$h(d_2)$

TT_2

Tapping

Time Stamp Autho

$h(d_1)$

TT_3

Adversary

Success of Attac
Getting the newe
TT_3 for older
document

Adversary models

Basic function

- Eavesdropping any message
- Requesting time-stamp token for any docume (including resending tapped time-stamp request)
- Receiving time-stamp token
- Poly-time Turing Machi

Subset of Delev-Yao model

We categorize additional setting as follows.

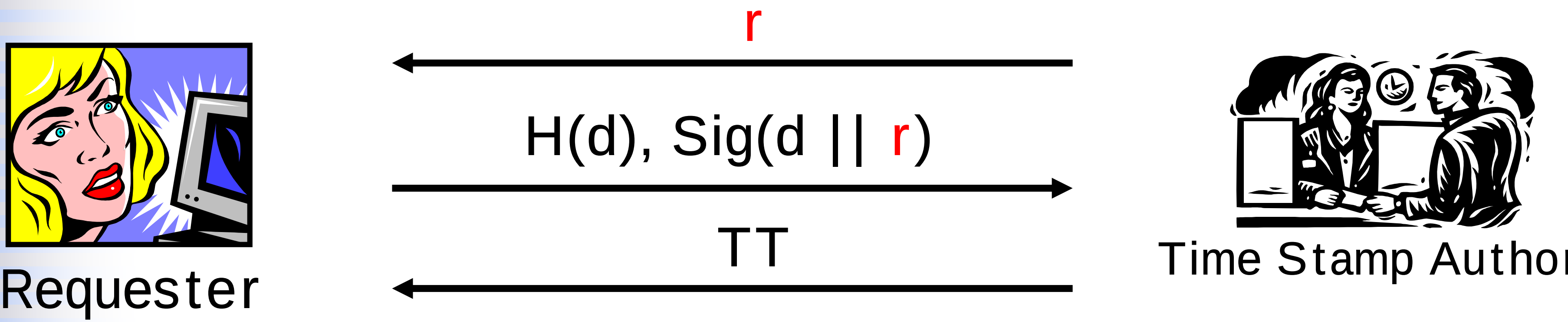
- Adversary can not collude with time-stamp authority
  - Adversary can obtain original document
  - Adversary can not obtain original document
- Adversary can collude with time-stamp authority

If adversary can not collude with TSA...

Point: How can the verifier confirm the requester's will?

If adversary can not know d ...

- Using challenge-and-response
  1. TSA sends a random r before time-stamp request.
  2. The requester calculates digital signature for d and r
  3. The adversary cannot calculate correct response



r

H(d), Sig(d || r)

TT

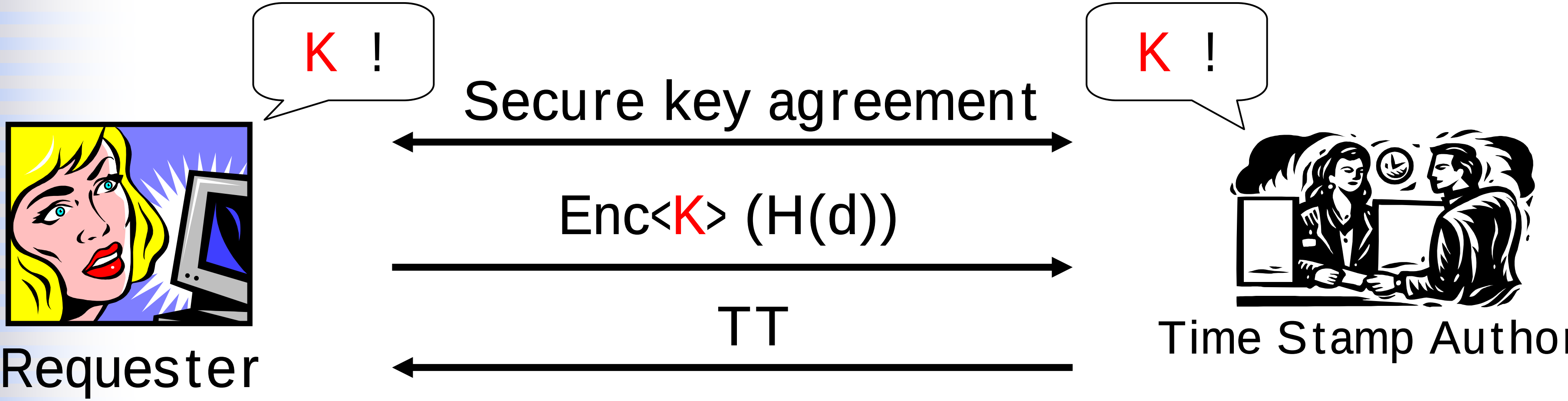
Requester

Time Stamp Author

If adversary can not collude with TSA... (cont.)

If Adversary can not know d ...

- Using hybrid-encryption scheme
  1. The requester encrypts the time-stamp request using random and one-time session key. (ex. SSL)
  2. The later adversary's time-stamp request is rejected by TSA unless key agreement scheme is secure.



K !

K !

Secure key agreement

Enc<K> (H(d))

TT

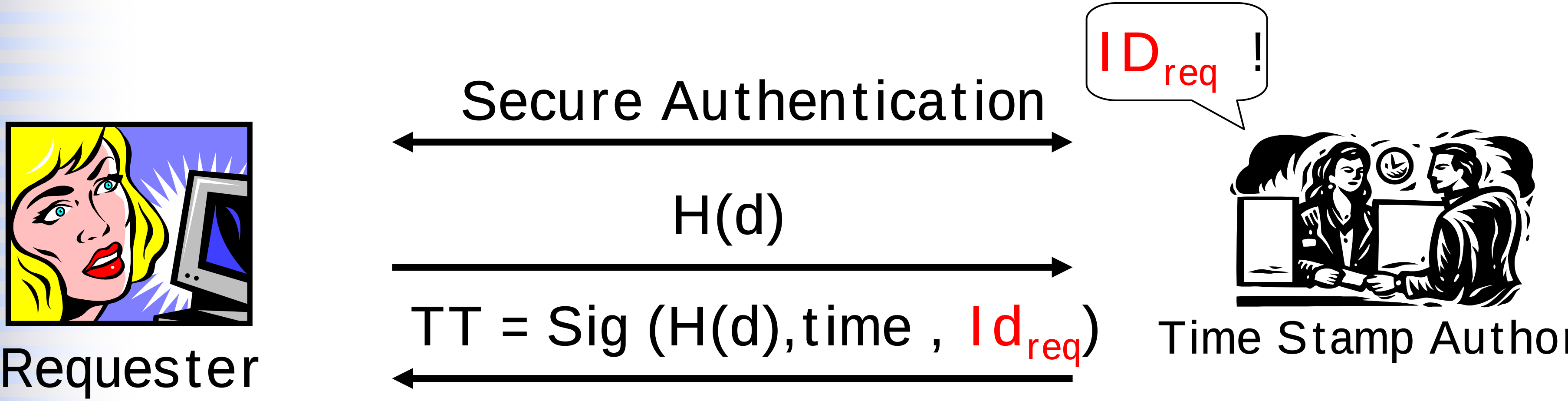
Requester

Time Stamp Author

If adversary can not collude with TSA… (cont.)

If adversary can know d …

- Authenticating then including identifier into time-stamp token
  1. The requester and TSA perform secure authentication.
  2. TSA includes identifier of the requester into the time-stamp token
  3. Adversary cannot obtain later time-stamp token with same Id_{req} unless authentication scheme is secure.



ID_{req} !

Secure Authentication

H(d)

TT = Sig (H(d),time , Id_{req})

Requester

Time Stamp Author

When the adversary can collude with TSA

- Adversary can obtain valid time-stamp token
  - For any document
  - For any time
- Adversary can obtain any secret information over the time-stamp protocol
  - Secret key for issuing
  - Challenge information …

Solutions in the previous slides do not work
to confirm the requester’s will.

Solution for this situation

1.The requester commits one-time secrets which ca prove

- Order of revision

- Consistency of revision

for each revision when she requests.

2. Add new procedure to verify which document is newer, when two documents are shown from different users.

Generating commitment using Hash-chain

- Generating initial value IV_d for each document
- Calculating size n hash-chain, where n is maximum revision number
- Keep IV_d and unused hash value secret

$$\begin{tikzcd}
IV \arrow[r, shift left, "h"] & h(IV) \arrow[l, dashed, shift left] \arrow[r, shift left, "h"] & \cdots\cdots \arrow[l, dashed, shift left] \arrow[r, shift left, "h"] & h^{n-1}(IV) \arrow[l, dashed, shift left] \arrow[r, shift left, "h"] & h^{n}(IV) \arrow[l, dashed, shift left]
\end{tikzcd}$$

n-th commitment

2nd commitment

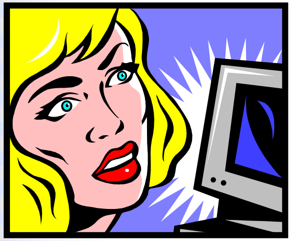
1st commitment

h: one-way permutation

Issuing sub-protocol

$= t_j$

d_j : j-th document



Requester

$$h(d_j \| h^{n-j+1}(IV_d))$$

$$TT_j$$

Same as existing procedure

Time Stamp Autho

$h(\bullet)$: One-way permutation

Verification of single time-stamp token

Almost same as existing time-stamping protocol !

d_j

TT_j

$h^{n-j+1}(IV_d)$

quester

Verifier

Verifies consistency all data.

- Verifying digital signature (Simple scheme)
- Verifying consistenc of hash-chain/tree (Linking scheme)

To compare ordinality of two documents

Requester

d	$h_1 = h^n(IV_d)$	TT_1
d'	$h_2 = h^{n-k+1}(IV_d)$	TT_2

Adversary

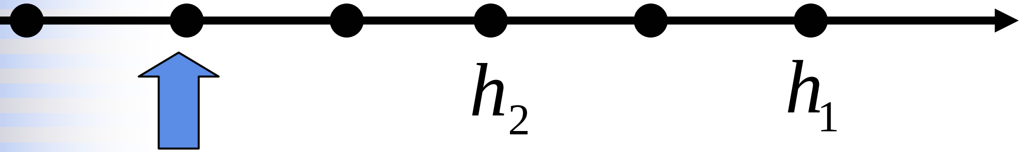
d	$\overset{?}{h}$	TT_3

Verifier

Verify TT_1 TT_2 TT_3

$$h_1 ?= h^{k-1}(h_2)$$

$$h_2 ?= h^m(\overset{?}{h}) \ (1 \le m \le$$



e

Early

h_2

h_1

$\overset{?}{h}$?

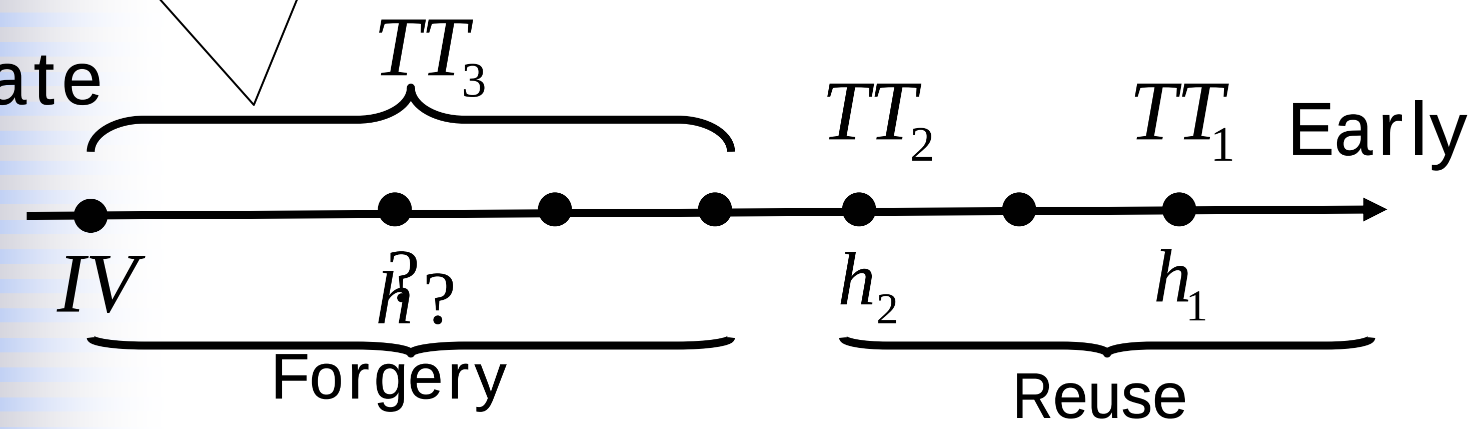
Security analysis

Sending d $\overset{?}{h}$ TT_3 which satisfies check equations in previous slide is required to prove TT_3 is newer than TT_2 and valid.

Check equations

$$h_1 ?= h^{k-1}(h_2)$$
$$h_2 ?= h^{m}(\overset{?}{h}) \quad (1 \leq m \leq$$

$\overset{?}{h}$ must be one of hash values in this range to fulfill the check equation. The probability of finding such value is 2^{-l_h}.



Performance analysis

Additional computation

Requester side

Calculating n hash values (maximum) for each document.

- In general n may be not so large.
- This give quite small impact to requester's procedure.

Verifier side

In ordinality verification,

- Three verifications of time-stamp tokens
- n+k-1 calculations of hash value
- Total computation cost in ordinality verification is not so large.

Compatibility with existing standard time-stamping scheme

The differences with existing scheme are

- Data to be time-stamped
  - In issuing procedure in requester side,
    - Calculating commitments using hash-chain
    - Asked to keep them secure
  - Issuing procedure in TSA side is same as existing schemes.
    - Calculating digital signature (Simple scheme)
    - Calculating hash-chain/tree (Linking scheme)
- Verification protocol for ordinality of two documents.
  - Additional procedure is required in verifier side.
  - Verification procedure of single time-stamp token is same as exiting scheme.

We can use existing TSA!

Conclusion

- Define user-side forward-dating attack
- Modeling adversary
- Solution when adversary can collude with TSA
  - ✓Using Hash-chain
  - ✓Committing the hash values into the time-stamp request
  - ✓Verification protocol for two different tokens
- Analysis
  - ✓Secure
  - ✓Low overhead
  - ✓Highly compatible with existing system