Addressing the Problem of Undetected Signature Key Compromise

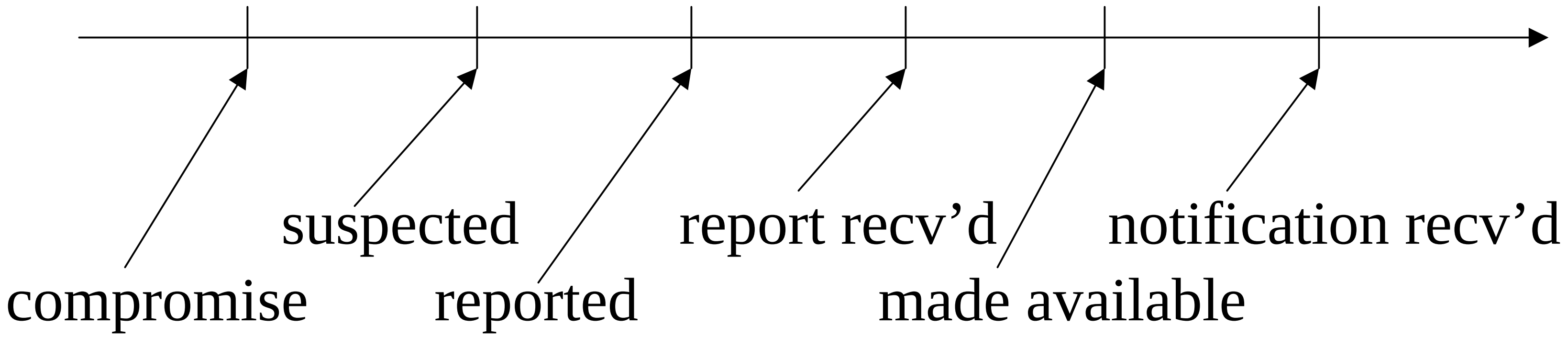
Mike Just, Paul Van Oorschot

Entrust Technologies

Outline

- Problem Definition
- Current Solutions
- Proposed Solution
  - second secret
  - synchronization
  - CHIP/COPE
- Concluding Remarks

Problem Definition



- Period of undetected key compromise
- Dilemma
  - Originator signed message during period?
  - Originator did not sign message during period?

Methods of Compromise

- Algorithmic attack
- Implementation failure
- Insider attack
- Brute-force attack

Current Solutions

- Time stamping of signature
  - message was signed before time *t*
- Revocation of verification certificate
  - occurs *after* compromise detection or suspicion
- Undetected compromise not resolved

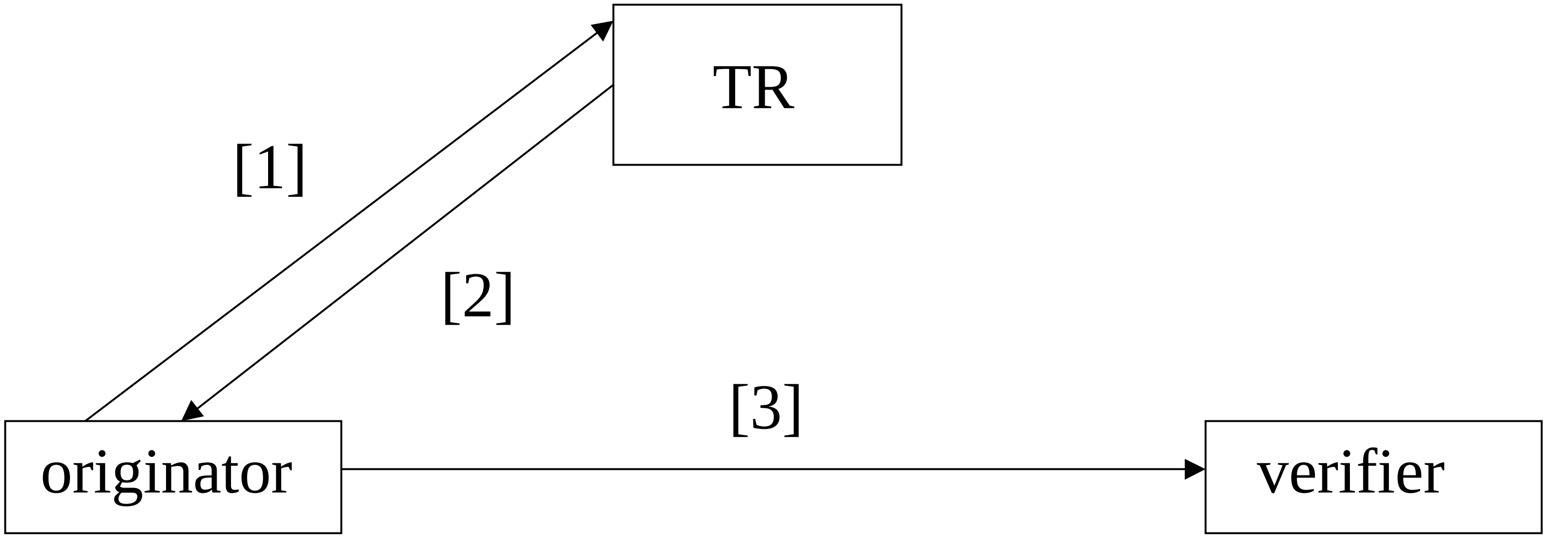
Current Solutions (cont’d)

- Redundant mechanism
  - threshold signatures
  - proactive signatures, certification
- multiple signers
- don’t necessarily preclude all attacks

Current Solutions (cont'd)

- Limit exposure from compromise
  - limit period of potential forgery, e.g., certificate expiry or revocation
  - proactive certification
  - limitation of signing privilege, i.e., type of signatures
  - limitation of number of signatures

Proposed Solution



- [1] request authentication for particular sig
- [2] receive second-level assurance
- [3] forward sig & second-level assurance

Properties

- Independence
  - – one attack doesn’t necessarily imply second
- binding
  - – signature bound to second-level request and response
- permits authentication
  - – allows identification of the originator

Second Secret Solution

- Shared key solution
  - – Setup: originator *u* and *TR* share a key *K*
  - – TR request: For signature *c*, *u* sends (c, $z=E_K(c)$) *to TR*
  - – TR response: $r=sig_T(c)$ is returned to *u* and verified
- Alternatives: secondary signature; Lamport keys

Synchronization Solution

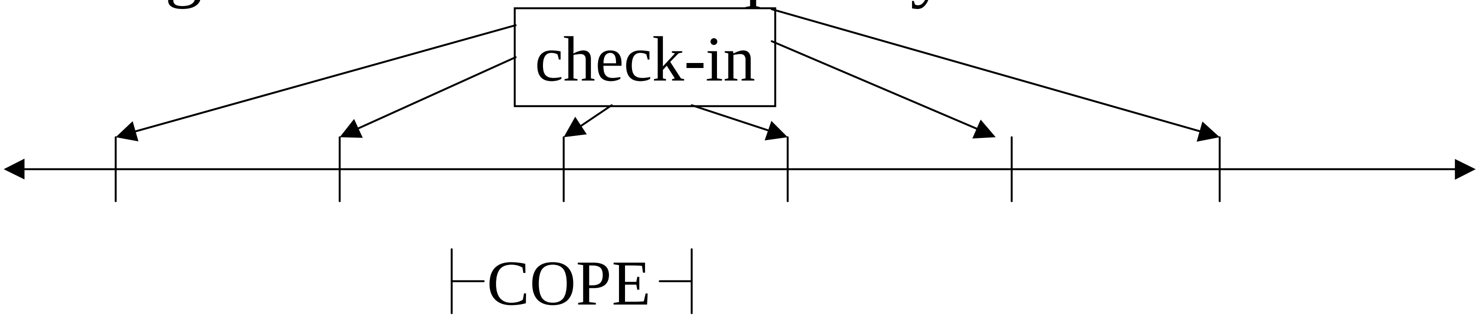
- Does not prevent signature production (first or second level) in case of signature key compromise, but
  - on its own, it allows *detection* by an honest user
  - with other measures, can *prevent* signature acceptance
- Alternatives: output of one-way function; time variant

Synchronization Solution (cont'd)

- Time of last signature
  - Setup: originator *u* and *TR* share a time t_0
  - TR request: For signature c_i, *u* sends $(c_{i,}\ t_{i-1})$ *to TR* (t_{i-1} *verified)*
  - TR response: $r=sig_T(c_i, t_{i-1}, t_i)$ is returned to *u* and verified

CHIP/COPE

- Cooling-off period (COPE)
- Check-in period (CHIP)
- Forgery is detectable prior to end of COPE
  - signatures can subsequently be rolled back



CHIP/COPE (cont'd)

- Time of last signature
  - TR request: For signature c_i, *u* sends *($c_{i,}$ t_{i-1}) to TR (t_{i-1} verified)*
  - CHIP verification: *TR* ensures that t_i - t_{i-1} < *t* where *t* = length(CHIP)
  - TR response: $r=sig_T(c_i, t_{i-1}, t_i)$ is returned to *u* and verified
  - Signature recipient waits till end of COPE, i.e., till time t_i+t

Concluding Remarks

- Proposed solution
  - – second-level authentication (independent secret; synchronization)
  - – increases likelihood of *detection*
  - – permits rollback of forged signatures
- Suitability
  - – applicable to automated, high-valued transactions, ...