1

Capture-Resilient Cryptographic Devices

Mike Reiter

University of North Carolina at Chapel Hill

2

Relevant Trends

- Proliferation of mobile devices
- Proliferation of networking
- Proliferation of security-relevant apps using these
  - In civilian applications, device may
    - Enable access to virtual private networks, corporate data
    - Enable modification of corporate assets (documents, code)
  - In military applications, device may
    - Send intelligence data, target coordinates, unit status
    - Enable access to mission data, status and readiness of other units, orders from unit commander

Capture and Misuse

3

- Physical capture a recognized problem
  - Four computers stolen from Wells Fargo, lost social security numbers of customers [Los Angeles Times, Nov 4, 2004]
  - General who heads Israel's foreign intelligence agency had cell phone stolen from car in Tel Aviv, leaking numbers of agents and secret service heads [Reuters, Mar 3, 2004]
  - Laptop stolen from UC Berkeley Graduate Division Offices that contained names and social security numbers of over 98,000 people [UC Berkeley News, Mar 28, 2005]
  - 478 laptops lost or stolen from the IRS between 2002–2006; 112 held sensitive taxpayer data, including SSNs [WTOP, Nov 15, 2006]
- Vulnerability due to device capture will grow as uses proliferate
  - Captor gains privileges (crypto keys) that device engenders (holds)

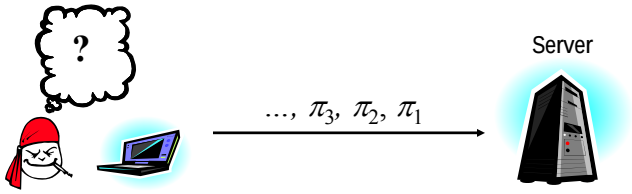
Capture-Resilient Cryptographic Devices

4

- A *capture-resilient device* is one that cannot be used by other than its rightful owner
  - No amount of reverse engineering exposes its cryptographic secrets
  - Does not rely on tamper-resistant hardware; a software-only solution
- Approach leverages networked nature of device
  - Most interesting uses of a key require network anyway
- Idea: The environment confirms that the device remains in its owner's possession before permitting its key to be used
  - Component in environment is called a "capture-protection server"

Security Goal #1

5



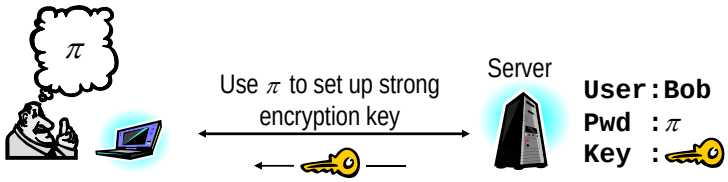
- If attacker captures device, then he can do no better than an *online* dictionary attack
  - Formally, can forge for device with probability $q/|D|$ after making q password guesses to server, where password is drawn from dictionary D
  - Server can detect and stop attack

Using the Network

6

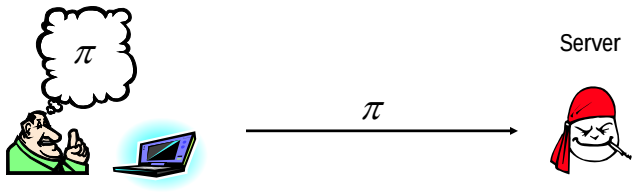
[Lomas et al. 1989; Bellovin & Merritt 1992; …; Perlman & Kaufman 1999]

- Store private key in a protected server that authenticates user before sending the private key



- Eavesdropper gains nothing to use in offline dictionary attack
- But … break-in at server leaks private key
  - Possibly after an offline dictionary attack

Security Goal #2

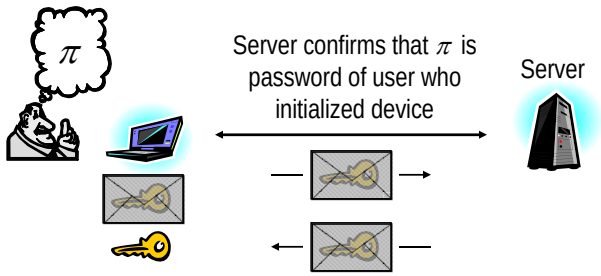


- If attacker compromises server (and password), then he still cannot forge signatures for the device
  - Server is *untrusted* in this sense

Reducing Trust in the Server

[w/ MacKenzie]

- Keep the key at the client, but in a disabled state



- Break-in at server leaks nothing
- Online dictionary attack possible only after device is captured
  - Server can again detect and stop the attack
  - Offline attack requires capture of both client device and server

Device Initialization

9

- Requires
  - secret key for the device SK_{dvc}
  - public key for the server PK_{svr}
  - one-way function h
  - pseudorandom function f
- Steps

$v \leftarrow_R \{0,1\}^\kappa$
$b \leftarrow h(\pi)$
$c \leftarrow f(v, \pi) \oplus SK_{dvc}$
$\tau \leftarrow E_{PK_{svr}}(<b, c>)$

save v, τ
(b, c, SK_{dvc} are deleted)

Key Retrieval Protocol

10

v, τ

$\beta \leftarrow h(\pi)$
$\rho \leftarrow_R \{0,1\}^\lambda$
$\gamma \leftarrow E_{PK_{svr}}(<\beta, \rho>)$

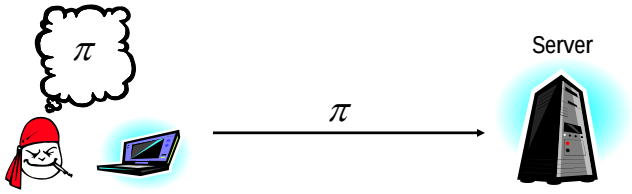
γ, τ →

$<\beta, \rho> \leftarrow D_{SK_{svr}}(\gamma)$
$<b, c> \leftarrow D_{SK_{svr}}(\tau)$
abort unless $b = \beta$
$\eta \leftarrow c \oplus \rho$

← η

$SK_{dvc} \leftarrow \eta \oplus \rho \oplus f(v, \pi)$

Security Goal #3

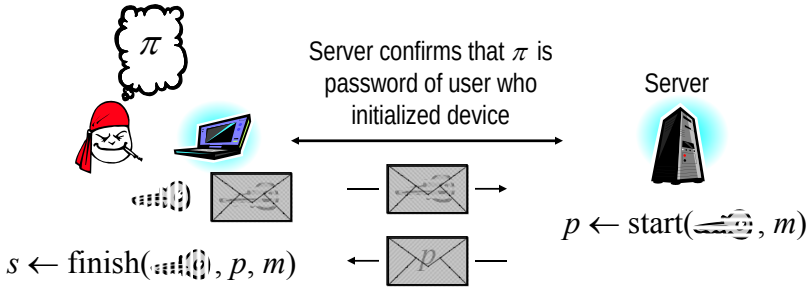


- If attacker compromises device and password, then he can forge only until the device key is *disabled* at the server.

Reducing Trust in the Client

[w/ MacKenzie; see also Boneh et al. 2001; c.f., Ganesan 1985]

- Can *disable* the device if stolen
  - Even if attacker knows the user's password



- Same properties as before, plus disabling
- Known techniques depend on particular form of private key
  - All use function sharing primitives

Example: RSA Signatures

13

- Private key is $<d, N>$
- Public key is $<e, N>$
- N is the product of two large primes
- $ed \equiv 1 \bmod \Phi(N)$
  - Φ is Euler's totient function
- Signature on m is $\sigma = H(m)^d \bmod N$
  - H is a collision-resistant hash function

RSA Signing Initialization

14

- Requires
  - secret key for the device $<d, N>$
  - public key for the server PK_{svr}
  - one-way function h
  - pseudorandom function f
- Steps

$v_0 \leftarrow_R \{0,1\}^\kappa$
$b \leftarrow h(\pi)$
$d_0 \leftarrow f(v_0, \pi)$
$d_1 \leftarrow d - d_0 \bmod \Phi(N)$
$\tau \leftarrow E_{PK_{svr}}(<b, d_1, N>)$

save v_0, τ
(b, d, d_0, d_1 are deleted)

RSA Signing Protocol

15

v_0, τ

$\beta \leftarrow h(\pi)$

$\rho \leftarrow_R \{0,1\}^\lambda$

$\gamma \leftarrow E_{PK_{svr}}(<m, \beta, \rho>)$

γ, τ →

$< m, \beta, \rho> \leftarrow D_{SK_{svr}}(\gamma)$

$<b, d_1, N > \leftarrow D_{SK_{svr}}(\tau)$

abort unless $b = \beta$

$\mu \leftarrow H(m)^{d_1} \bmod N$

$\eta \leftarrow \mu \oplus \rho$

← η

$\mu \leftarrow \eta \oplus \rho$

$d_0 \leftarrow f(v_0, \pi)$

$\sigma \leftarrow \mu(H(m)^{d_0}) \bmod N$

The Cost of Disabling

16

	Public key operation	Private key operation	Messages	Device exps	Server exps
RSA signatures	verify $s^e \bmod n = H(m)$	$s \leftarrow H(m)^d \bmod n$ return s	2	2	3
ElGamal encryption ($q = p-1$)	$r \leftarrow_R \mathbf{Z}_q$ $c_1 \leftarrow g^r \bmod p$ $c_2 \leftarrow my^r \bmod p$ return $<c_1,c_2>$	return $c_2/(c_1)^x \bmod p$	2	5	5
DSA signatures	$z \leftarrow (s_2)^{-1} \bmod q$ verify $s_1 \equiv_q g^{H(m)z} y^{s_1 z} \bmod p$	$r \leftarrow_R \mathbf{Z}_q$ $s_1 \leftarrow g^r \bmod p$ $s_2 \leftarrow (H(m)+xs_1)/r \bmod q$ return $<s_1 \bmod q, s_2>$	4	46	50

17

A Closer Look at Disabling

- Only the user should be able to disable her key
  - Can't require user to have the device to disable
  - Can't authenticate the user using only her password (too weak)
- Our approach adds two new values to initialization

  $t \leftarrow_R \{0,1\}^\kappa$

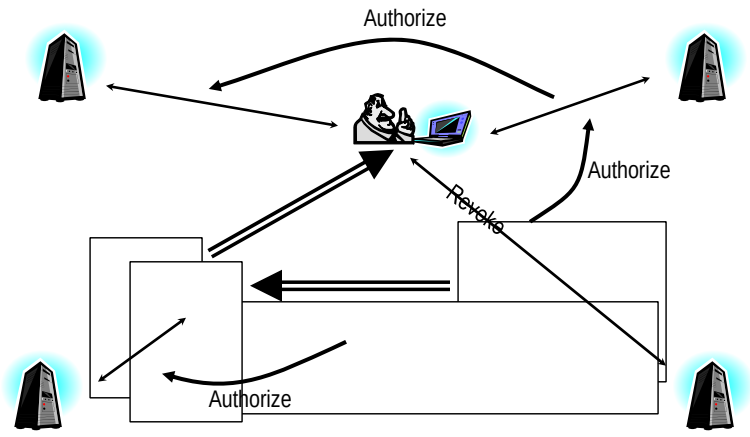
  $u \leftarrow h(t)$
- Value ***u*** is inserted into the ticket

  $\tau \leftarrow E_{PK_{svr}}(<b, u, d_1, N>)$
- User copies ***t*** off device, and disables by sending ***t*** to server
  - Server ignores any ticket containing $u = h(t)$

18

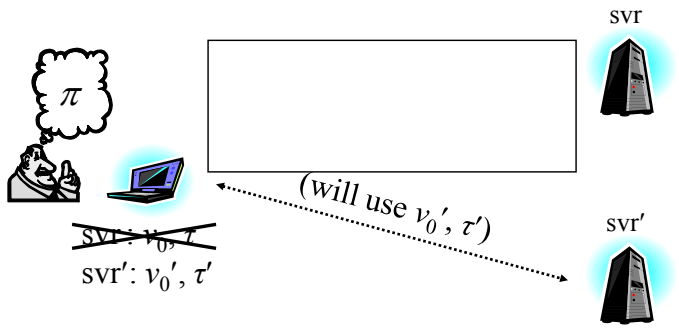
Delegation

[w/ MacKenzie]



19

Delegation Framework

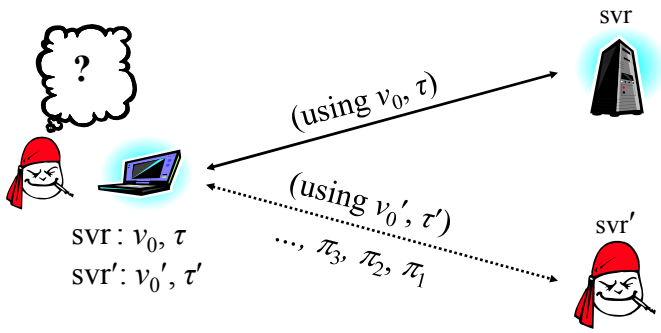


- Delegation: Interact with svr to create v_0', τ' for use with svr′
- Revocation: Unilaterally revoke (erase) v_0, τ for any svr

20

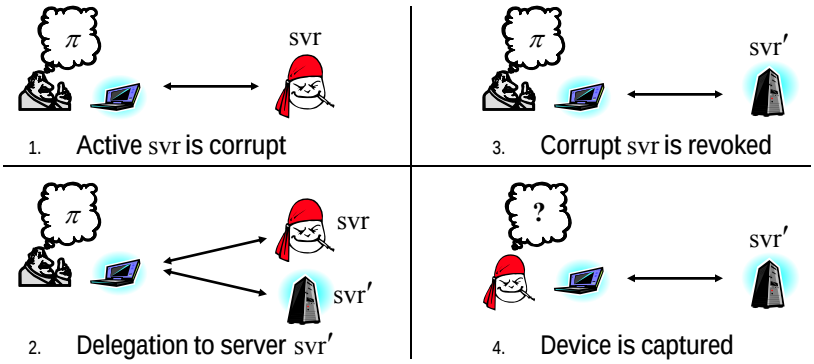
Delegation and Dictionary Attacks

- Delegation must be a password-protected operation
  - Otherwise, attacker can delegate to a server he controls to conduct an undetectable (offline) dictionary attack



21

Delegation and Dictionary Attacks (cont.)



- Can we ensure that attacker can forge with probability no better than q/D, after making q queries to $\mathbf{svr}'$?
  - Requires that attacker learned nothing about π in role as svr

22

Delegation and Dictionary Attacks (cont.)

- Unfortunately, some information about π must leak, namely the number of incorrect password guesses (mistypes)
  - Server presumably must count these, to limit online dictionary attacks
- But, we modify our protocol to make sure that the number of mistypes is the *only* information that leaks
  - Even hide the frequency of a given mistype
- We then prove security in a model where the adversary controls when mistypes occur (and so the number of mistypes he sees)

Limiting Password Information Leakage

- Initialization

  $v_1 \leftarrow_R \{0,1\}^\kappa$
  $b \leftarrow f_1(v_1, \pi)$
  …
  $\tau \leftarrow E_{PK_{svr}}(<b, ...>)$

- Protocol

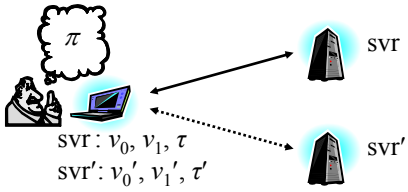
v_0, v_1, τ

$v_2 \leftarrow_R \{0,1\}^\kappa$
$\beta \leftarrow f_2(f_1(v_1, \pi), v_2)$
…
$\gamma \leftarrow E_{PK_{svr}}(<... v_2, \beta ...>)$

γ, τ →

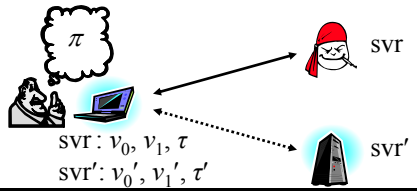
$< \ldots v_2, \beta ...> \leftarrow D_{SK_{svr}}(\gamma)$
$<b, ...> \leftarrow D_{SK_{svr}}(\tau)$
abort unless $f_2(b, v_2) = \beta$

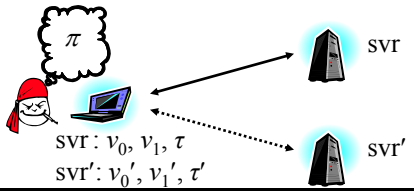
Who Creates Tickets?



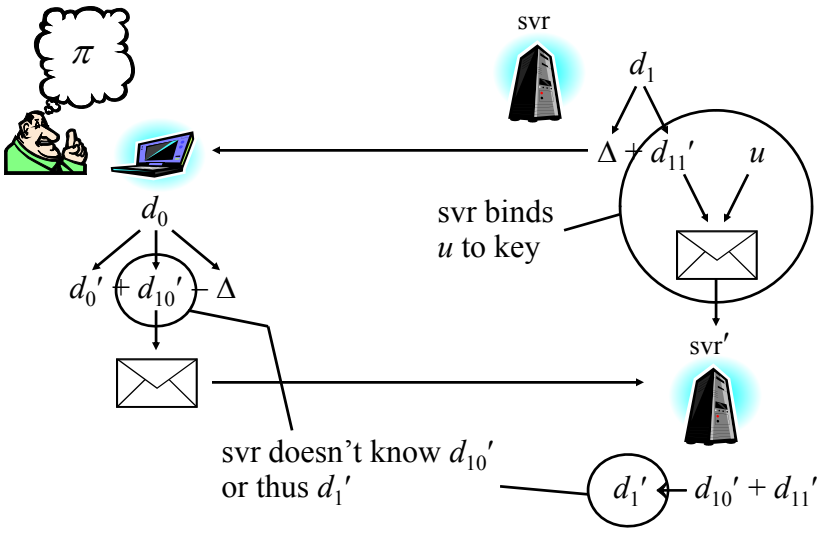
Question: When delegating from svr to svr′, who creates τ' ?

- If svr, then it knows share of svr′
  - Revoking svr still enables offline attack with device capture
- If device, then it can replace ***u*** with something else
  - Disabling won't be possible





Distributed Ticket Creation



Distributed Ticket Creation (cont.)

…

…

$\Delta \leftarrow_R \{0,1\}^{\lambda+\kappa}$
$d_{11}' \leftarrow d_1 - \Delta$
$\zeta' \leftarrow E_{PK_{svr'}}(<u, d_{11}'>)$
$\eta \leftarrow \rho \oplus <\Delta, \zeta'>$

η

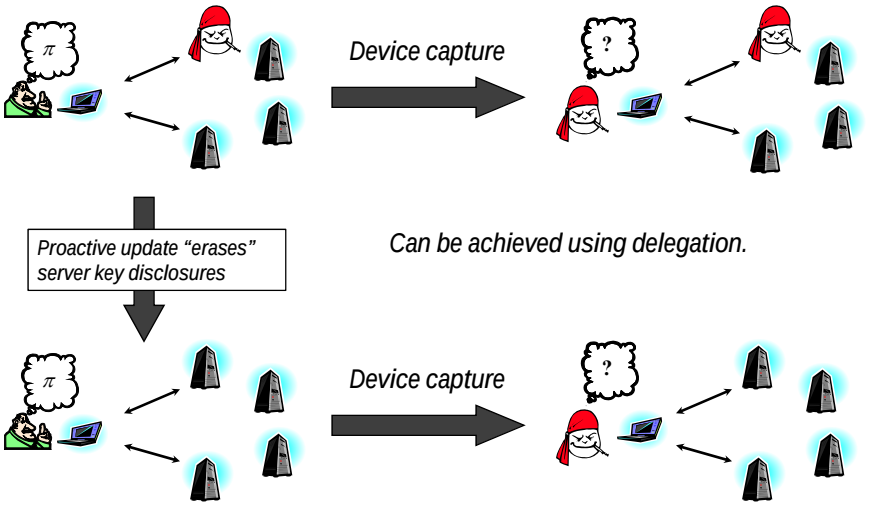
$<\Delta, \zeta'> \leftarrow \rho \oplus \eta$
$d_0 \leftarrow f(v_0, \pi)$
$d_0' \leftarrow f(v_0', \pi)$
$d_{10}' \leftarrow d_0 - d_0' + \Delta$
$\tau' \leftarrow E_{PK_{svr'}}(<b, d_{10}', \zeta', N>)$
save <svr′: v_0', v_1', τ'>

- **svr′** can recover d_1' by decrypting τ' and then ζ'

Proactivity

27

[Ostrovsky & Yung 1991; Canetti & Herzberg 1994; Herzberg et al. 1995; …]



Definitions

28

- Attackers (static)
  - A1: Does not compromise device.
  - A2: Compromises device, does not compromise π, and any server authorized when device is captured is never compromised.
  - A3: Compromises device, does not compromise π, and some server authorized when device is captured is compromised.
  - A4: Compromises device and password, but does not compromise any server.
- Other
  - D is the dictionary from which password is chosen.
  - n is number of servers.
  - Server encryption E is adaptive chosen ciphertext secure.
  - q_{svr} (q_{dvc}, …) are number of queries to servers (device, …).

29

Theorem 1

- A1: Does not compromise device.
- Assumptions
  - f_0 is a pseudorandom function family
- Thm (informal): If an A1 attacker forges with probability ε, then there is a forger that forges in the underlying the RSA signature scheme with probability $\varepsilon' \approx \varepsilon$.

30

Theorem 2

- A2: Compromises device, does not compromise π, and any server authorized when device is captured is never compromised.
- Assumptions
  - f_0, f_1,f_2 are pseudorandom function families
- Thm (informal): If an A2 attacker forges with probability

  $$\varepsilon = q_{\mathrm{svr}}/|D| + \psi$$

  then either there is an attacker that breaks encryption with probability

  $$\varepsilon' \approx \psi/(2n(2q_{\mathrm{dvc}}+1))$$

  or there is a forger that forges in the underlying the RSA signature scheme with probability

  $$\varepsilon' \approx \psi/2.$$

31

Theorem 3

- A3: Compromises device, does not compromise π, and some server authorized when device is captured is compromised.
- Assumptions
  - f_0, f_1, f_2 are random oracles
  - Mistypes are *diffuse*:

$$\forall \pi : \Pr[\pi_0 \leftarrow_R D;\ \pi \leftarrow \text{mistype}(\pi_0)] = 1/|D|$$

- Thm (informal): If an A3 attacker forges with probability

$$\varepsilon = (q_{f0} + q_{f1} + q_{f2})/|D| + \psi$$

then there is a forger for the underlying RSA signature scheme that forges with probability

$$\varepsilon' \approx \psi.$$

32

Theorem 4

- A4: Compromises device and password, but does not compromise any server.
- Assumptions
  - Underlying RSA signature scheme is deterministic.

- Thm (informal): If an A4 attacker forges after disabling with probability ε, then either there is an attacker that breaks encryption with probability

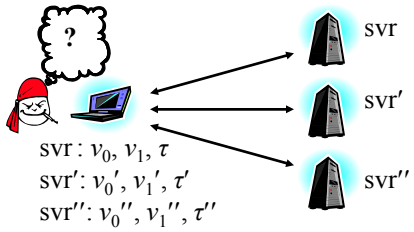
$$\varepsilon' \approx \varepsilon/(2n(q_{\text{dvc}}+1))$$

or there is a forger that forges in the underlying the RSA signature scheme with probability

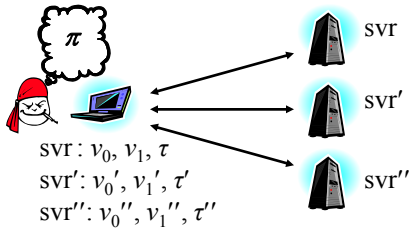
$$\varepsilon' \approx \varepsilon/2.$$

Security Costs of Delegation

33



- If each server permits q guesses, then k authorized servers permit kq guesses



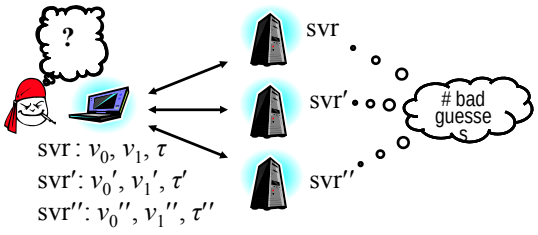
- *Every* admissible server must be disabled to ensure that the key will not be used

Regaining Control Using Shared Objects

34

[w/ Samar & Wang]

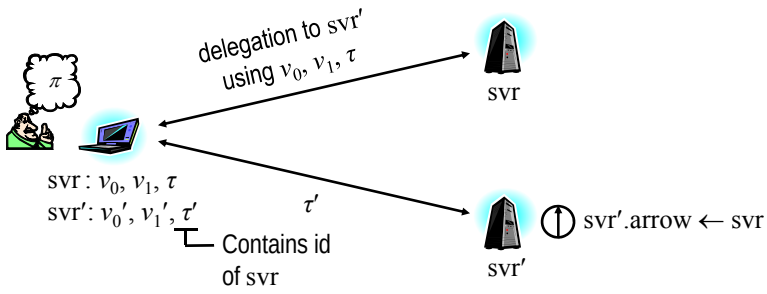
- Servers *share* a counter per device
  - Increment counter per wrong password guess
  - Set to max value for disabled device



- Challenges in storing and maintaining the counter
  - Naively storing it in one location undercuts the purpose of delegation
  - No global view of the authorized servers at any time
- Our goal: a "minimal" object sharing infrastructure that
  - Enables accurate maintenance of a shared counter despite compromises
  - Effectively utilizes locality of reference and adapts to delegations

35

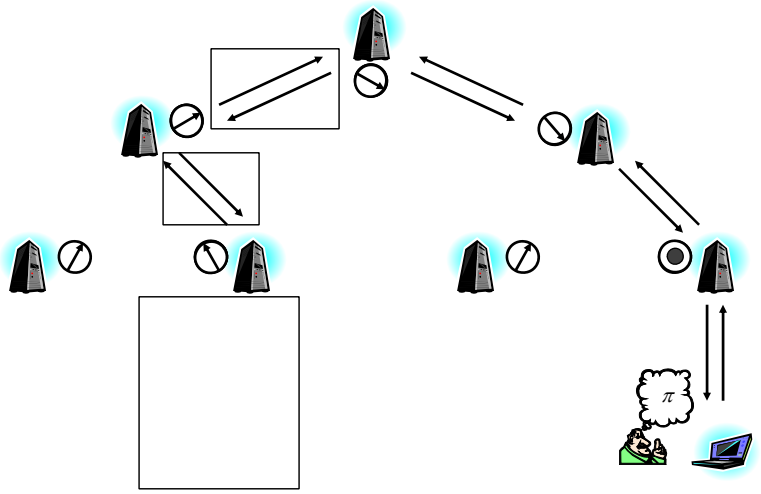
Arrow Initialization



- Each server maintains an "arrow" indicating the direction in which the counter lies
- Arrow for newly delegated server is initialized with consenting server for the delegation

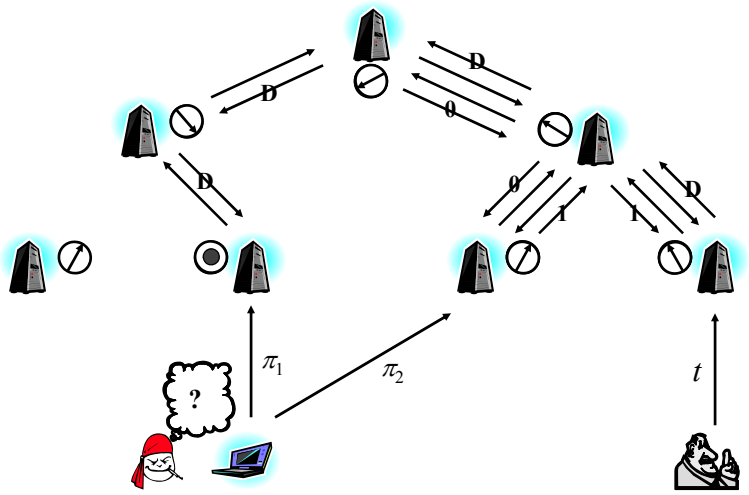
36

Retrieving the Counter



Mutual Exclusion

37



Conclusion

38

- An approach for defending a key from misuse even if device holding it is captured and reverse-engineered
  - Requires network connectivity to reach a "capture protection server"
  - Requires human operator to have (possibly low-entropy) secret
  - Does NOT expose key to server
  - Server can be used to disable device even if attacker knows password
- Delegation enables new servers to be authorized dynamically
  - Revocation enables authorization to be removed, with the security being essentially as if server was never authorized
  - Results in substantive changes to protocols
  - Significantly extends today's key management approaches

Bibliography

- P. MacKenzie and M. K. Reiter. Networked cryptographic devices resilient to capture. *International Journal of Information Security* 2(1):1–20, November 2003.
- P. MacKenzie and M. K. Reiter. Delegation of cryptographic servers for capture-resilient devices. *Distributed Computing* 16(4):307–327, December 2003.
- M. K. Reiter, A. Samar and C. Wang. The design and implementation of a JCA-compliant capture protection infrastructure. In *Proceedings of the 22nd IEEE Symposium on Reliable Distributed Systems*, October 2003.