INFORMATION SECURITY: Economics and Beyond

Hasan Cavusoglu

Hasan Cavusoglu

- Faculty, UBC – the Sauder School since 2003
- Background
  - B.Sc. in EE from Bogazici University (Istanbul, Turkey)
  - Ph.D. and M.Sc. in Management from UT Dallas
- Main area of research is Economics of IS
  - The economic consequences of IT advances and innovations

Economics

- Economics is the social science that analyzes the production, distribution, and consumption of goods and services
- Simply put, economics is the study of making choices
- Economics is the study of how people choose to use resources

Research Streams

- Information security & privacy management
- IT-enabled transformation
- Socio-organizational aspects of information security

Information security management

- Increasing information security risks
- How to deal with multifaceted information security risks by using multitude of tools
- Understanding the economic trade-offs
- Objective
  - Normative guideline for decision makers
  - Policy implications

Cavusoglu, Cavusoglu, Raghunathan (2007), *IEEE Transactions on Software Engineering*.

EFFICIENCY OF VULNERABILITY DISCLOSURE MECHANISMS TO DISSEMINATE VULNERABILITY KNOWLEDGE

Efficiency of Vulnerability Disclosure Mechanisms to Disseminate Vulnerability Knowledge

- What is security vulnerability in software?
- *"A flaw in a product that makes it infeasible—even when using the product properly—to prevent an attacker from usurping privileges on the user's system, regulating its operation, compromising data on it, or assuming ungranted trust"* --Microsoft

Motivations

- One of the reasons for increased rate of security breaches is vulnerabilities in software
  - Rapid propagation of attacks
  - Easy to launch an attack
  - Zero-day attacks
- Software has security vulnerabilities because
  - Poor software development
  - 20 flaws/KLOC (NIST, 2002)
- Recent increased awareness of software vendors

- Vulnerabilities are inevitable

Vulnerabilities

- Discovery of Vulnerability
  - Developers of the software vendor
  - Malicious Users (i.e., Hackers)
  - Benign Users
- How should a benign user should disseminate vulnerability information?
  - Should it be kept secret?
  - Should the public be informed immediately?

Vulnerability Disclosure Process

- Secrecy → Full Vendor Disclosure
  - Little intention of vendor to fix
  - No enforcement
- Transparency → Full Public Disclosure
  - Gives the vendor a strong incentive to fix
  - Allows vulnerable firms to take some immediate measure
- Intermediate → Hybrid Disclosure

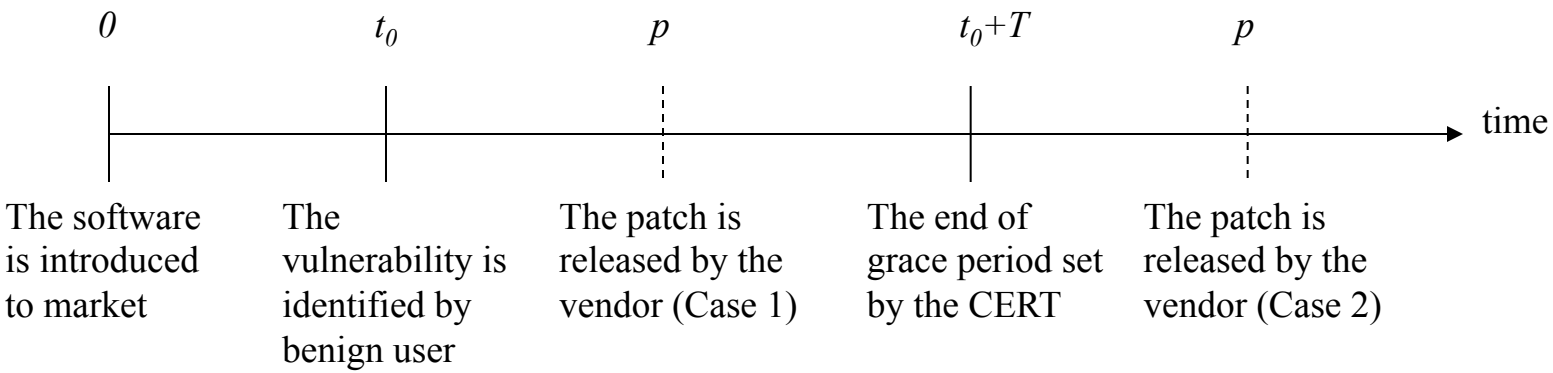
Vulnerability Disclosure Mechanisms

- A game theoretic model that captures the incentives of four main stakeholders
  - software developer (i.e., vendor), software deployers (i.e., firms), vulnerability identifier (benign user or hacker), and central coordinator)
- Two-stage game

Decision Makers

- CERT minimizes total expected social loss
  - Cost to software vendor (patch development cost)
  - Damage to software users
  - Workaround cost
- Vendor minimizes its cost
  - Patch development cost
  - Reputation cost

Model



- Attack rate
  - Before public disclosure: α
  - After public disclosure: αk where k>1
- Level of vulnerability
  - Before public disclosure: δ where 0<δ<1
  - After public disclosure: δγ where 0<γ<1

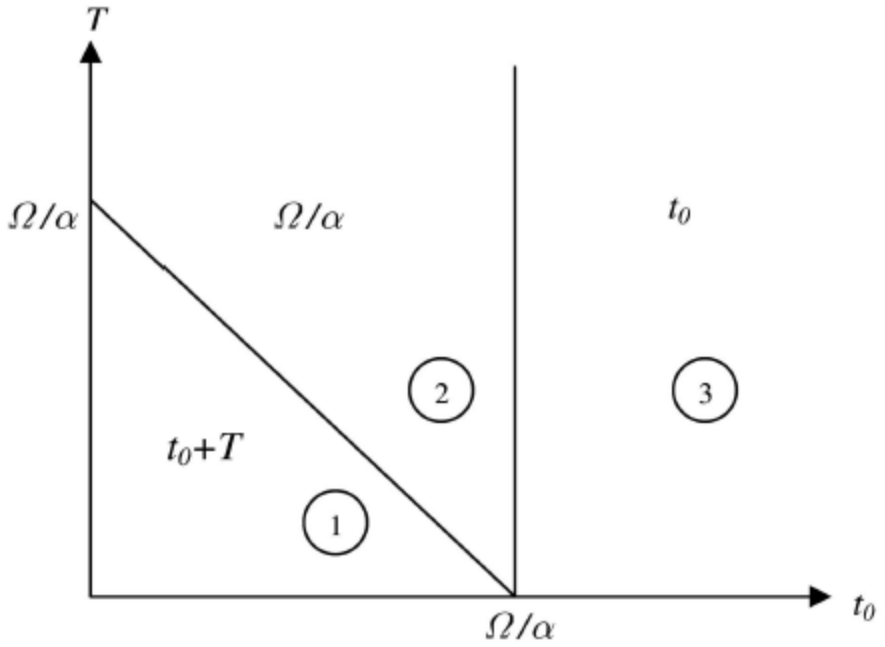
Objective Functions

- Vendor
  - Min (Patch development + Reputation)
    - Patch development: ε_1-$\varepsilon_2(p-t_0)$
    - Reputation cost: β*(Average # of users exploited)
- CERT
  - Min (Patch development + Total damage to users + Workaround cost)
    - Damage to a firm of type θ where $0<\theta<1$: $D\theta$
    - Workaround cost per firm: s

Proposition 1

- When $\gamma k < \Omega$, the vendor disregards the vulnerability when given a finite grace period.
  - After the public disclosure, savings in patch development cost outweighs the reputation loss associated with postponing the patch release per unit time

Vendor's Best Response (p) when γk>Ω



Proposition 2

- If the vendor releases a patch when it is given a finite grace period ($\gamma k > \Omega$), it intends to wait to release the patch if the discovery time of the vulnerability (t_0) is smaller than the tolerance level of the vendor (Ω / α), and release the patch immediately, otherwise.

Coordinator's Optimal Disclosure Policy

- After anticipating the vendor's best response for a given T (i.e. p*(T)), the coordinator decides on the optimal T which minimizes the social loss (i.e.
- $T^* = \arg_T \min [C(p^*(T), T)]$
- Equilibrium is $(T^*, p^*(T^*))$.

Optimal Disclosure Policy and Patch Release Time in the Single-Vendor Case

Extension: Multiple Vendors

- What if the vulnerability affects multiple vendors? Should the vendors be given a longer (shorter) grace period?
- Each vendor might have different preferences as to when to release a patch to address the vulnerability
  - Different set of customers
  - Different cost structure
- The patch release decision of one vendor of one can put others at a disadvantage

Results in Multiple Vendor Case

- Optimal disclosure policy of the coordinator may not guarantee the release of a patch from both vendors in the multiple vendor case
- When the optimal policy does elicit a patch from each vendor, the grace period in the multiple vendor case falls between the grace periods that it would be set individually for vendors in the single vendor case

Extension: Early Discovery

- If the discovery of vulnerability occurs earlier (t_0' < t_0), the vendor releases the patch no later than when it would release otherwise (p'*=< p*).
- If the discovery of vulnerability occurs earlier (t_0' < t_0), the coordinator does not shorten the grace period (i.e., T'*>= T*).
- The society is always better off with an early discovery of the vulnerability.

Cavusoglu, Cavusoglu, Zhang (2008), *Management Science*

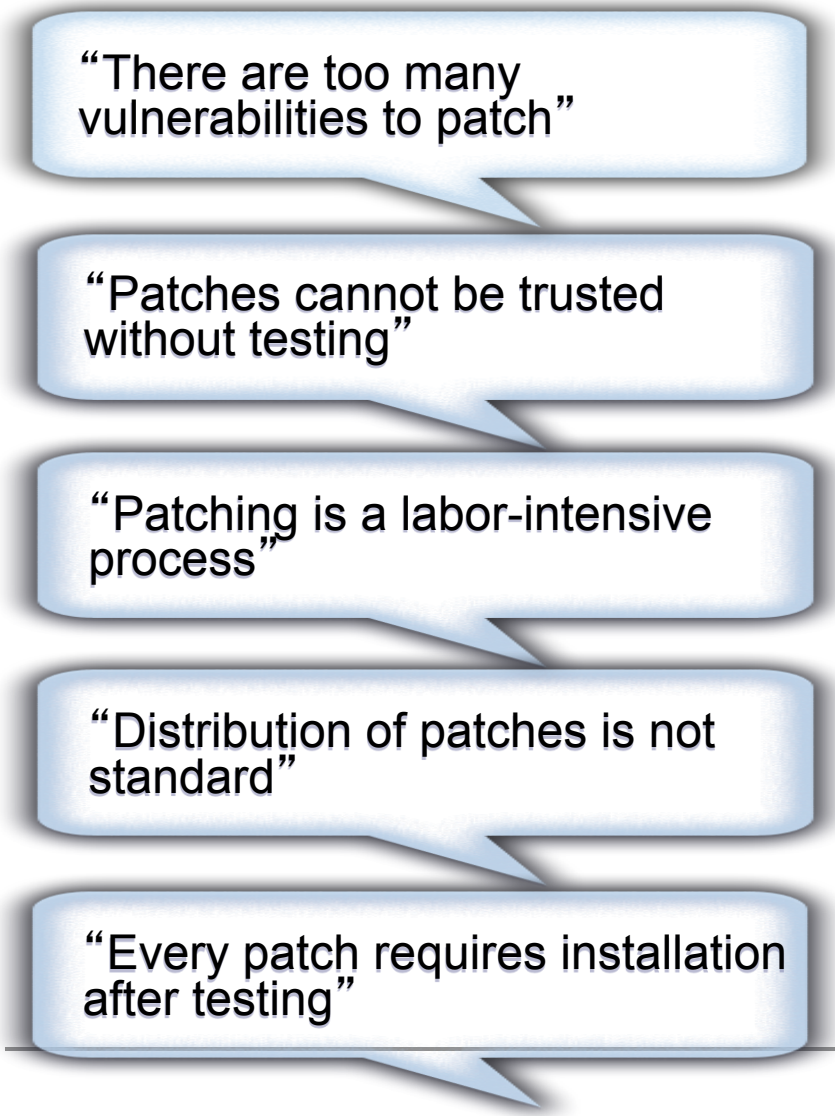
SECURITY PATCH MANAGEMENT:

SHARE THE BURDEN OR SHARE THE DAMAGE?

Path Management Problem

- The best solution to fix vulnerabilities is application of patches
- Firms do not apply patches promptly
  - 95% of breaches could be prevented by keeping systems up-to-date with patches (Dacey 2003)
    - Microsoft had released a patch to fix the vulnerability that *Slammer* exploited 6 months before the incident
    - Also *CodeRed*, *Nimda*, *Nanchi*, *Klez*, *SoBig*, *BugBear*
- Why don't firms apply patches as soon as vendors release them?

Problems with Patching



Around 150 vulnerabilities are announced per week

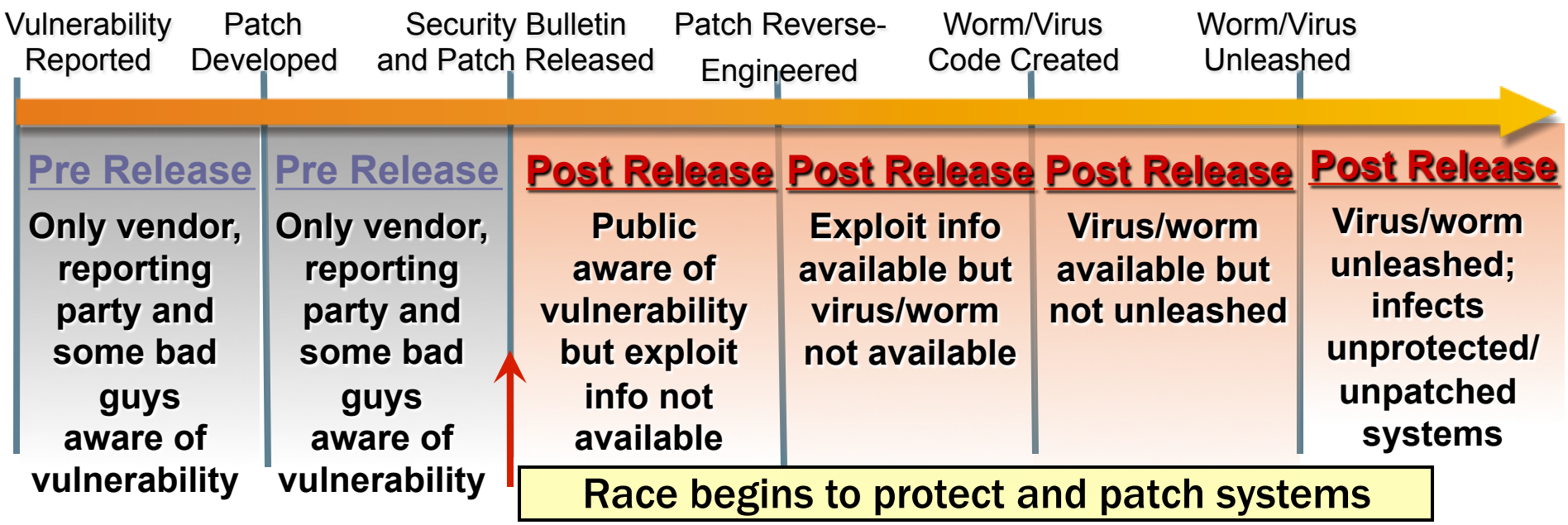
Some patches do not work properly and/or conflict with other applications

Cost of applying a patch is $900 per server $700 per client

Some patches are not available on the vendor site

Systems must be taken down and rebooted

Problems with Not Patching



- Once an exploit is released, it may be too late to consider patching
  - Slammer exploited 90 percent of vulnerable systems within 10 minutes of its release (Dacey 2003)
  - CodeRed infected a total of 359,000 computers within 14 hours of its release (McGhie 2003)

How to Update Systems?

- Vendor's Patch Release Policy is important
  - (Past) Release as soon as patches are ready
  - (Today) Release periodically
    - Monthly : Microsoft
    - Quarterly : CA, Oracle, PeopleSoft
- Different views about patching within firms
  - To system administrators, patching is risky (i.e., operational risk)
  - To security administrators, not patching is risky (i.e., security risk)
- Firms patch their systems periodically
  - Monthly: Novartis Pharma AG, Therma Electron

Research Questions

- How should patches be released and how should systems be updated with released patches?
  - Centralized Patch Management
  - Decentralized Patch Management
- How can we align the incentives of firms and vendor for effective patch management?
  - Cost Sharing Only
  - Liability Only
  - Cost Sharing Plus Liability

Model Basics

- One vendor and one firm
- Poisson process () for vulnerability identification
- Nested patch management
  - Patch update cycle is a multiple of patch release cycle

Time-Driven	Event-Driven
Firm's patch update cycle is T_f	Firm's patch update cycle is N_f
Vendor's patch release cycle is T_v	Vendor's patch release cycle is N_v
$T_f = kT_v$	$N_f = kN_v$

Model Basics (cont'd)

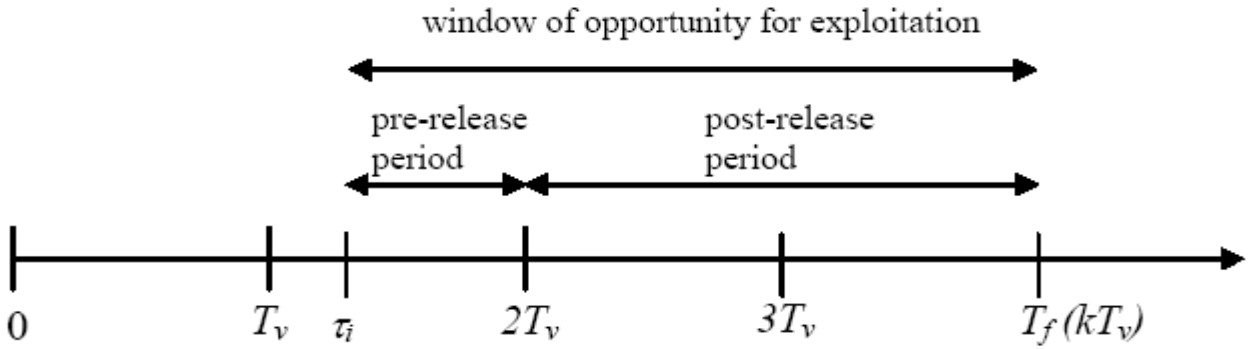
- Firm's Problem
  - (i) Damage Cost
    - vendor not releasing patches (pre-release) (c_b)
    - firm not updating with released patches (post-release) (c_a)
  - (ii) Patch Update Cost
    - identification of patches, downtime during update (K_f)
    - testing, configuration changes, installation (nc_f)
  - Firm chooses its *update cycle* to minimize its cost
    - In time-driven patch management
      - Update Cycle: Once in every T_f time units
    - In event-driven patch management
      - Update Cycle: Once after N_f patches are released

Model Basics (cont'd)

- Software Vendor's Problem
  - (i) Patch Release Cost
    - developing and testing patches (nc_v)
    - informing public about a release (K_v)
  - (ii) Cost of Loss in Reputation
    - loss in future sales
      - if patch is not available (pre-release) ($\$_b$)
      - if patch is available (post-release) ($\$_a$)
  - Vendor chooses its *release cycle* to minimize its cost
    - In time-driven patch management
      - Release cycle: Once in every T_v time units
    - In event-driven patch management
      - Release cycle: Once after N_v vulnerabilities are identified

The Integrated System

Time-Driven Patch Management



- Patch management cost (release and update)(1)

$$kK_v + c_v n + K_f + c_f n.$$

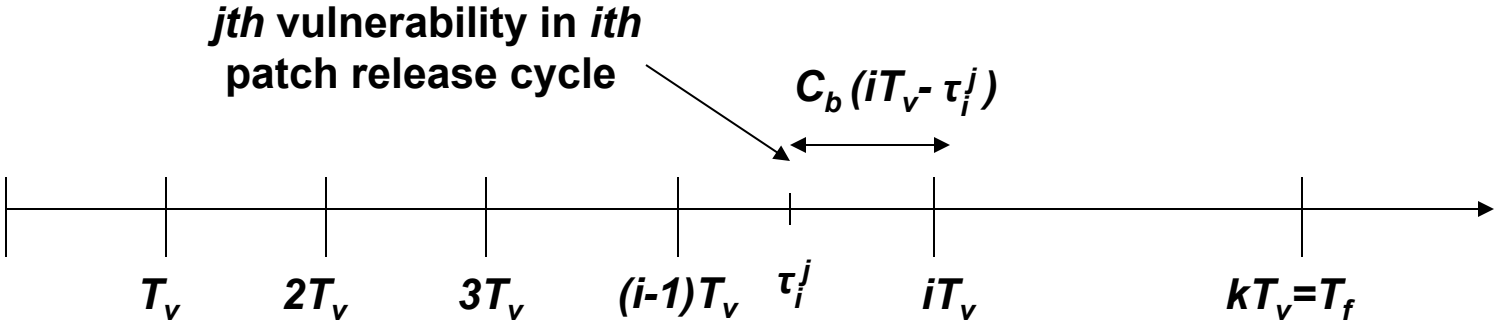
- Damage cost in pre-release periods (2)

$$c_b n T_v / 2.$$

- Damage cost in post-release periods (3)

$$c_a n(k-1) T_v / 2.$$

Damage Cost in Pre-Release Periods

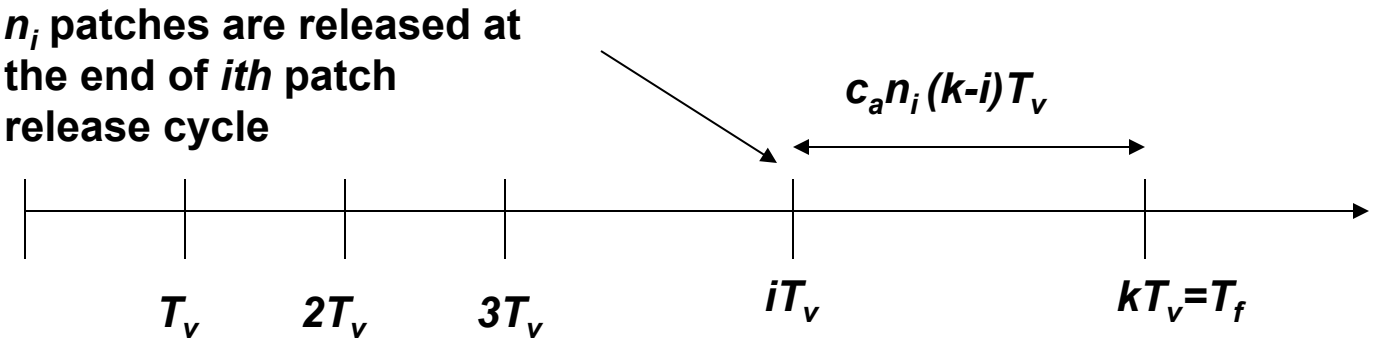


$$\sum_{i=1}^{k}\sum_{j=1}^{n_i} c_b(iT_v - \tau_i^j). \qquad (3)$$

- Total elapsed time in cycle *i* $E[\sum_{j=1}^{n_i}(iT_v - \tau_i^j)] = n_i T_v/2.$
- Expected value of (3)

$$E[\sum_{i=1}^{k}\sum_{j=1}^{n_i} c_b(iT_v - \tau_i^j)] = \sum_{i=1}^{k} E[\sum_{j=1}^{n_i} c_b(iT_v - \tau_i^j)] = \sum_{i=1}^{k} c_b n_i T_v/2 = c_b n T_v/2.$$

Damage Cost in Post-Release Periods



$$\sum_{i=1}^{k} c_a n_i (k-i) T_v = c_a T_v \sum_{i=1}^{k} n_i (k-i)$$

- Given *n*, $\{n_1, \ldots, n_k\}$ is multinomial
- Expected value of (2)

$$E_{\{n_1,\ldots,n_k\}}[c_a(n_1(k-1)T_v + \cdots + n_{k-1}T_v) | \sum_{i=1}^{k} n_i = n] = c_a n(k-1)T_v/2.$$

Integrated System Cost

- From (1), (2) and (3)

$$L^I(T_v, T_f|n) = K_f + c_f n + kK_v + c_v n + c_b n T_v/2 + c_a n(k-1)T_v/2.$$

- Total expected system cost during one update cycle

$$L^I(T_v, T_f) = \sum_{n=0}^{\infty} L^I(T_v, T_f|n) \frac{(\lambda T_f)^n}{n!} e^{-\lambda T_f}$$

$$= K_f + kK_v + [c_f + c_v + c_b T_v/2 + c_a(k-1)T_v/2]\lambda T_f$$

- Expected average system cost per unit time

$$C^I(T_v, T_f) = L^I(T_v, T_f)/T_f$$

$$= \frac{K_f}{T_f} + c_a \lambda T_f/2 + \frac{K_v}{T_v} + (\frac{c_b}{2} - \frac{c_a}{2})\lambda T_v + (c_f + c_v)\lambda.$$

The Integrated System Solution

Time-Driven Patch Management

- The central planner's problem

$$\min_{T_v \geq 0, T_f \geq 0} \{C^I(T_v, T_f) | T_f = kT_v \text{ for any } k\}$$

PROPOSITION 1. *Let T_v^* and T_f^* be the optimal patch-release and update cycles for the centralized system, respectively. Then,*

$$T_v^* = T_f^* = \sqrt{\frac{2(K_v + K_f^T)}{\lambda c_b}}, \tag{8}$$

and the minimum expected average system cost is

$$C^I(T_v^*, T_f^*) = \sqrt{2\lambda(K_v + K_f^T)c_b} + \lambda(c_f + c_v). \tag{9}$$

The Decentralized System

The Firm's Problem

- Average expected cost for the firm

$$C_f(T_v, k) = [c_b T_v^2 \lambda k/2 + c_a \lambda (k-1) k T_v^2/2 + K_f + c_f \lambda k T_v]/(kT_v)$$
$$= c_b T_v \lambda/2 + c_a \lambda (k-1) T_v/2 + K_f/(kT_v) + c_f \lambda.$$

- The firm's problem

$$\min_k \{c_b T_v \lambda/2 + c_a \lambda (k-1) T_v/2 + K_f/(kT_v) + c_f \lambda\}$$

$$s.t.\ k \text{ is an integer.}$$

Lemma 3 *For a given patch release cycle T_v of the vendor, k^* satisfies*

$$k^*(k^*-1) < \frac{2K_f}{\lambda c_a T_v^2} \text{ and } k^*(k^*+1) \geq \frac{2K_f}{\lambda c_a T_v^2}.$$

The Decentralized System
The Vendor's Problem

- Average expected cost for the vendor

$$C_v(T_v, k^*) = \frac{k^* K_v + [\alpha_b T_v/2 + \alpha_a T_v(k^* - 1)/2 + c_v]\lambda k^* T_v}{k^* T_v}$$

$$= K_v/T_v + (\alpha_b - \alpha_a)\lambda T_v/2 + \alpha_a \lambda k^* T_v/2 + c_v\lambda.$$

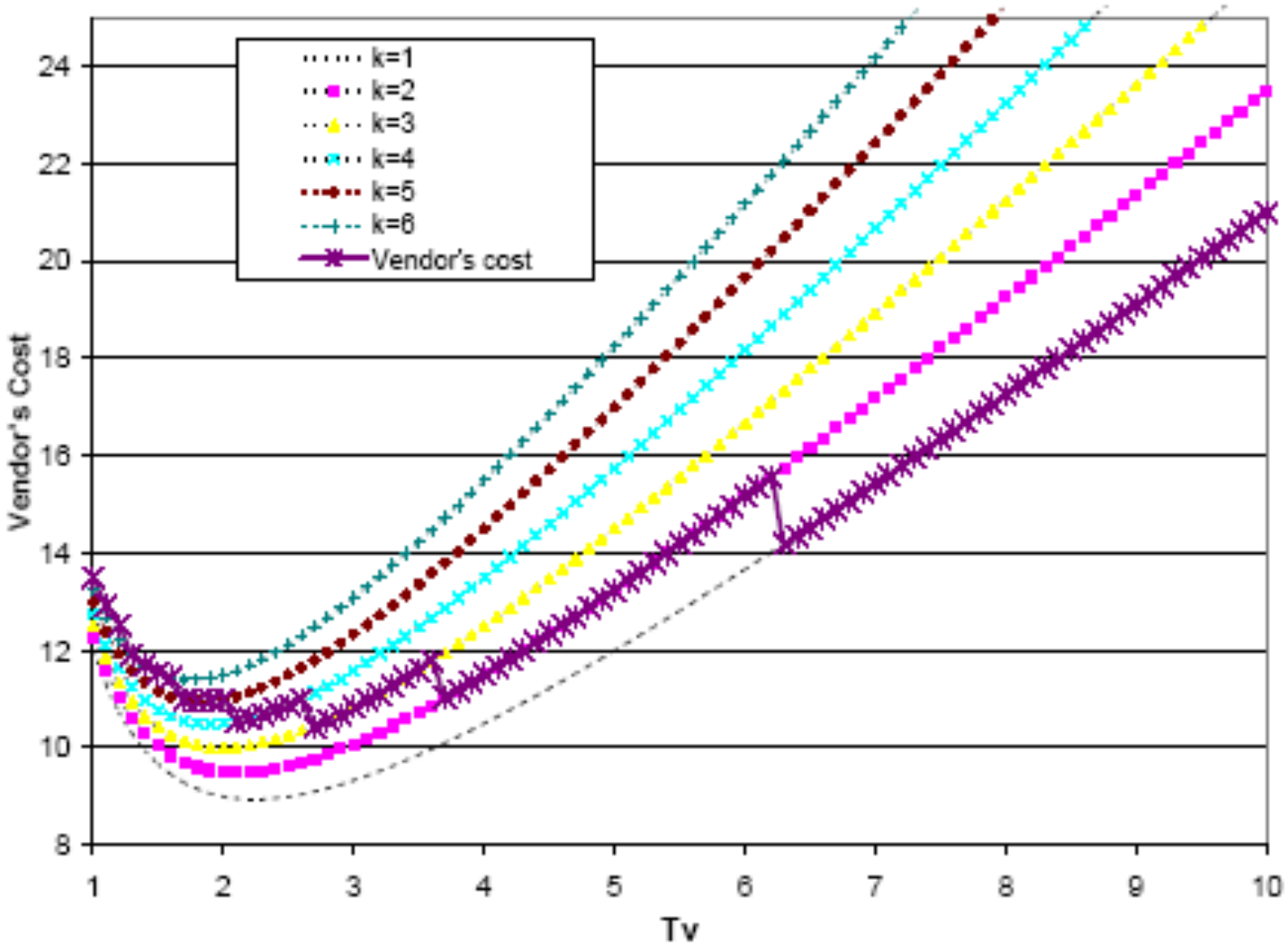
- The vendor's problem

$$\min_{T_v,k} K_v/T_v + (\alpha_b - \alpha_a)\lambda T_v/2 + \alpha_a \lambda k T_v/2 + c_v\lambda$$

$$s.t. k(k-1) < \frac{2K_f}{\lambda c_a T_v^2} \text{ and } k(k+1) \geq \frac{2K_f}{\lambda c_a T_v^2}.$$

Lemma 4 *If $K_f/c_a \leq 2K_v/\alpha_b$, then the optimal patch release cycle is $T_v^{p*} = \sqrt{2K_v/(\lambda\alpha_b)}$ and the corresponding patch update cycle T_f^{p*} is equal to T_v^{p*}.*

Vendor's Cost Function



Coordination Schemes

- How to align the incentives of vendor and firm to achieve the socially optimal patch release and patch update in a decentralized setting?
- Cost Sharing
  - Vulnerabilities are defects caused by vendors
  - Firms bear the full cost of patching
- Liability
  - Vendor's release policy may prevent firms from closing the vulnerability
  - Firms bear the full cost of damage
- Cost Sharing plus Liability

Coordination Schemes

Cost Sharing

- A portion of the fixed cost for path update is charged to vendor (θ^c)
- Patch release and update cycles are identical

$$\frac{\alpha_b}{2c_a}(1-\theta^c)K_f \leq K_v + \theta^c K_f \leq (K_v + \theta^c K_f/2)\frac{\alpha_b + \alpha_a}{\alpha_b}$$

- Synchronized cycle is socially optimal

$$(K_v + \theta^c K_f)/\alpha_b = (K_v + K_f)/c_b$$

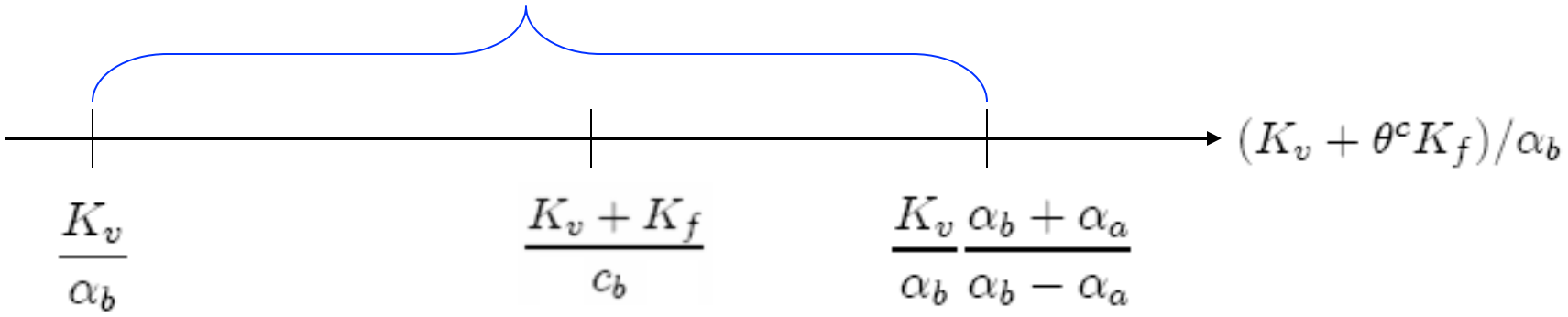
Proposition 2 *Cost sharing by itself can achieve the socially optimal patch update and release if*

$$\frac{K_v}{\alpha_b} \leq \frac{K_v + K_f}{c_b} \leq \frac{K_v}{\alpha_b}\frac{\alpha_a + \alpha_b}{\alpha_b - \alpha_a}. \tag{16}$$

Then, the required level of cost sharing is $\theta^c = (K_v + K_f)\alpha_b/(K_f c_b) - K_v/K_f$.

Interpretation of the Result for Cost Sharing

Feasible Region to Achieve Coordination Using Cost Sharing Only



Vendor's tolerance level with *no cost sharing*

System's tolerance level

Vendor's tolerance level with *maximum cost sharing*

$$\frac{K_v + K_f}{c_b} = (K_v + \theta^c K_f)/\alpha_b \quad \theta^c = (K_v + K_f)\alpha_b/(K_f c_b) - K_v/K_f$$

Coordination Schemes

Liability

- A portion of the pre-release damage cost is charged to vendor (θ^l)
- Patch release and update cycles are identical

$$\frac{K_f}{c_a} \leq \frac{2K_v}{\alpha_b + \theta^l c_b}$$

- Synchronized cycle is socially optimal

$$\frac{K_v}{\alpha_b + \theta^l c_b} = \frac{K_v + K_f}{c_a + c_b}.$$

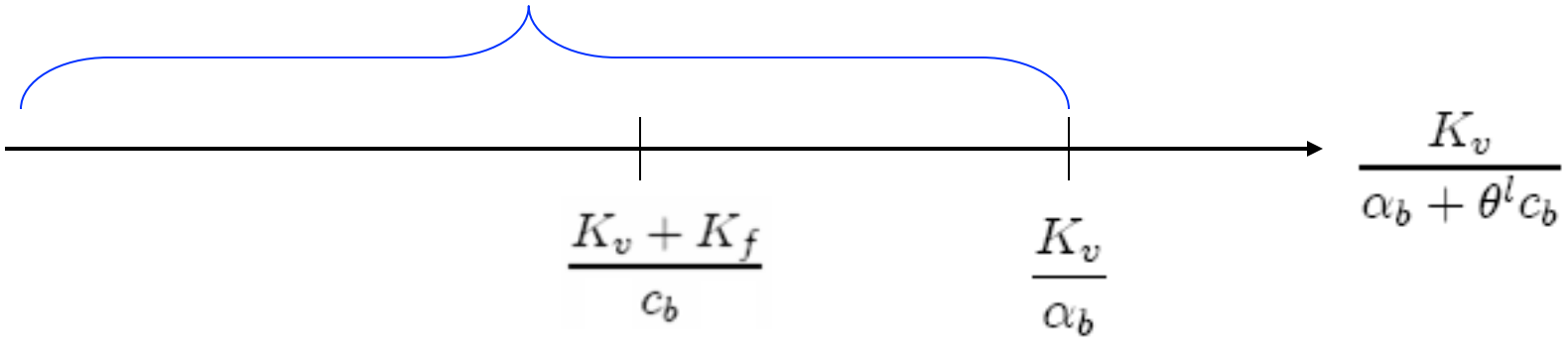
Proposition 3 *Liability can achieve the socially optimal patch release and update if*

$$\frac{K_v + K_f}{c_b} \leq \frac{K_v}{\alpha_b}.$$

Then, the required level of liability is $\theta^l = K_v/(K_v + K_f) - \alpha_b/c_b$.

Interpretation of the Result for Liability

Feasible Region to Achieve Coordination Using Liability Only



System's tolerance level

Vendor's tolerance level with *no liability*

$$\frac{K_v + K_f}{c_b} = \frac{K_v}{\alpha_b + \theta^l c_b} \quad \Leftrightarrow \quad \theta^l = K_v/(K_v + K_f) - \alpha_b/c_b$$

Coordination Schemes

Cost Sharing Plus Liability

- Does the joint use of cost sharing and liability
  - increase the size of the feasible region? and/or
  - give more flexibility in choosing levels of θ^c and θ^l ?
- Patch release and update cycles are identical

$$\frac{\alpha_b + \theta^l c_b}{2c_a}(1-\theta^c)K_f \leq K_v + \theta^c K_f \leq (K_v + \theta^c K_f/2)\frac{\alpha_b + \theta^l c_b + \alpha_a}{\alpha_b + \theta^l c_b}$$

- Synchronized cycle is socially optimal

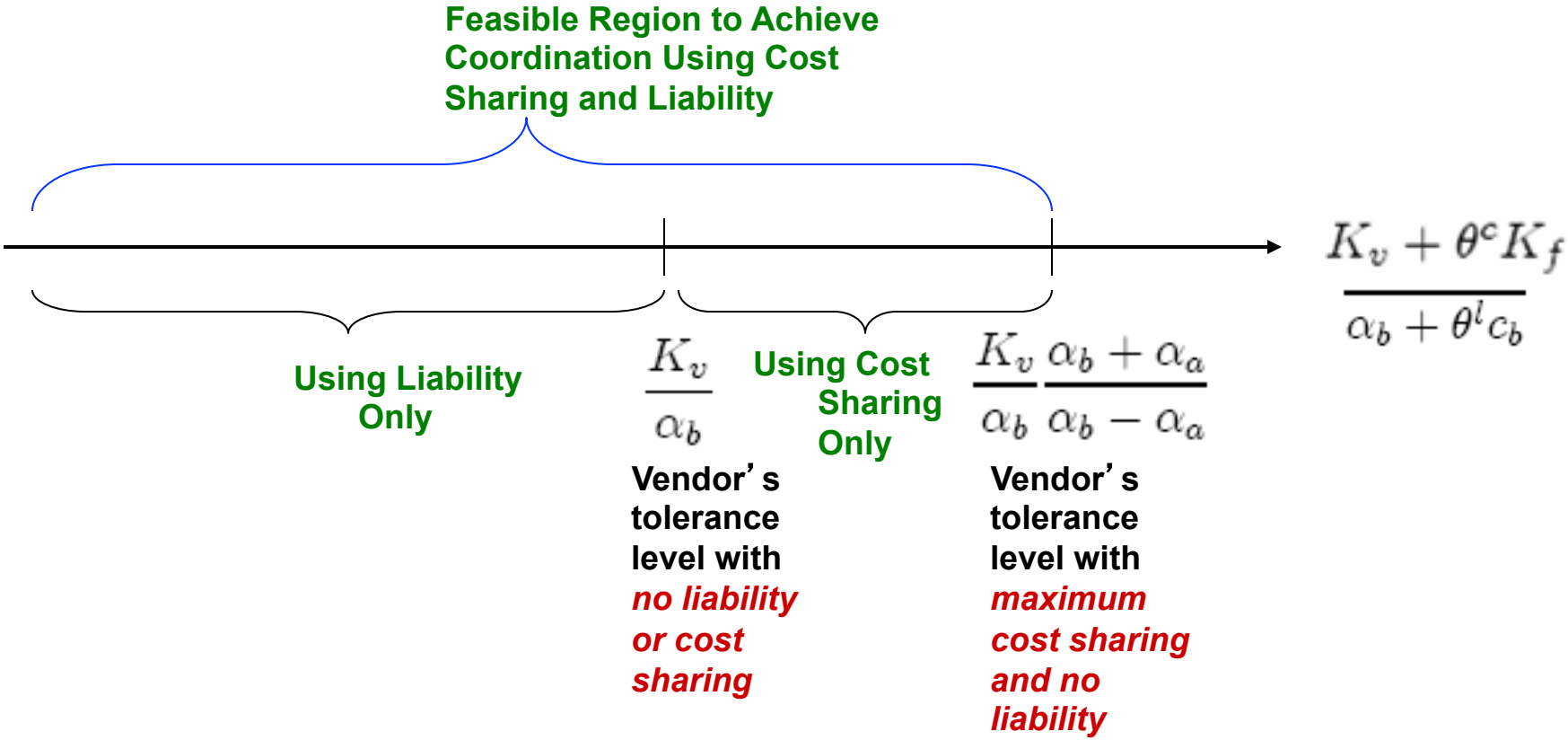
$$(K_v + \theta^c K_f)/(\alpha_b + \theta^l c_b) = \frac{K_v + K_f}{c_b}$$

Proposition 4 *Cost sharing together with liability can achieve the socially optimal patch release and update if*

$$\frac{K_v + K_f}{c_b} \leq \frac{K_v}{\alpha_b}\frac{\alpha_b + \alpha_a}{\alpha_b - \alpha_a}.$$

And the levels of cost sharing and liability satisfy $(K_v + \theta^c K_f)/(\alpha_b + \theta^l c_b) = (K_v + K_f)/c_b$.

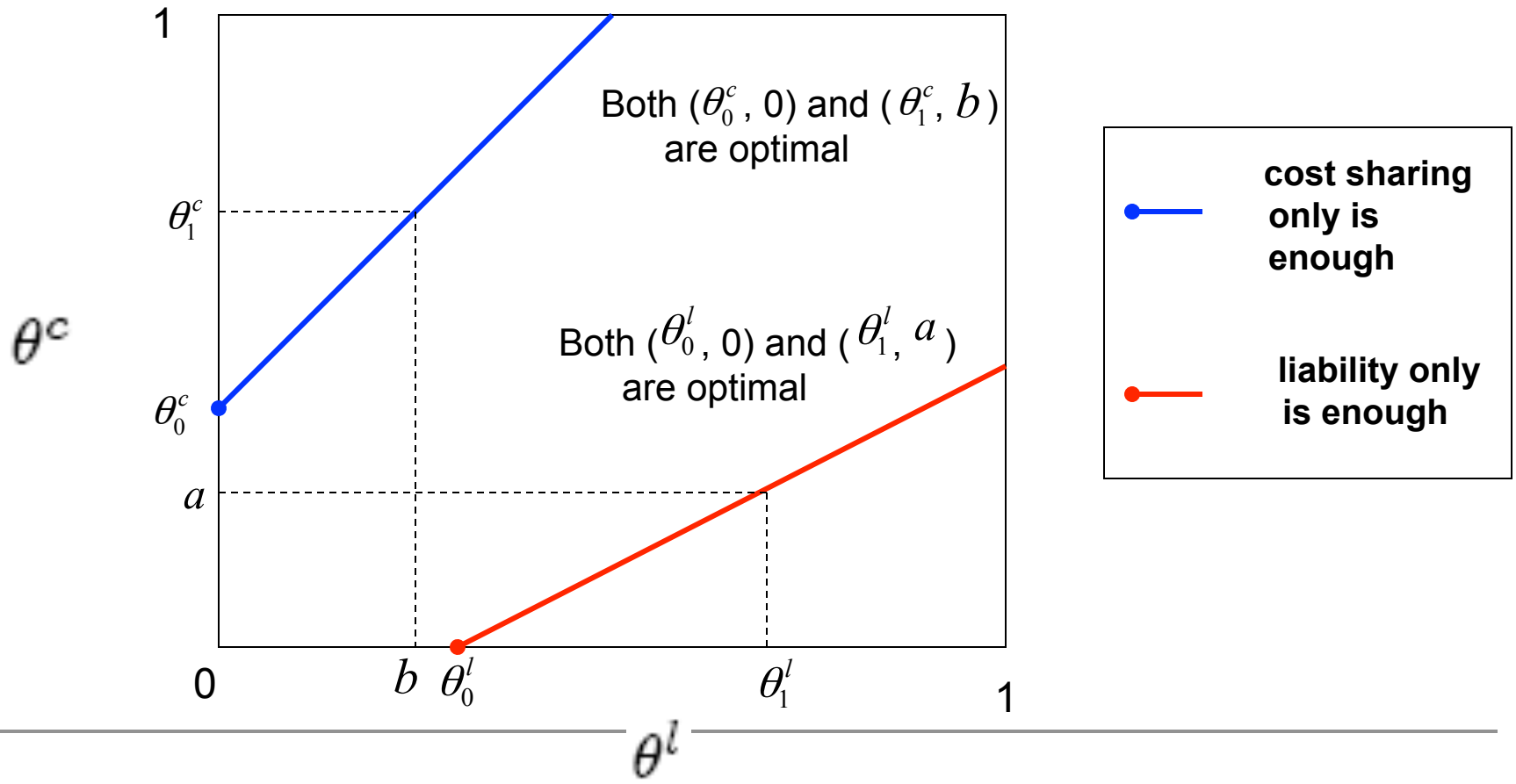
Interpretation of the Result for Cost Sharing Plus Liability



$$\frac{K_v + K_f}{c_b} = \frac{K_v + \theta^c K_f}{\alpha_b + \theta^l c_b}$$

Combinations of cost sharing and liability levels required

Optimal Levels of Cost Sharing and/or Liability



Summary of Main Results

- Socially optimal patch management requires *synchronization* of patch release and update cycles
- Individual decision making may result in *suboptimal* choices for patch release and update cycles
- Cost sharing or liability may *coordinate* patch release and update decisions in a non-cooperative setting
- Cost sharing and liability are *not substitutes*

Implications of Results

- Vendors have no incentive to fix their software since the cost of insecurity is borne by firms. Unless vendors bear some cost, security of their customers will not in the vendors' best financial interest (Schneier 2004)

- **Liability** helps if vendors are releasing less frequently than what the social optimality requires (i.e., **share the damage**)

- **Cost sharing** helps if vendors are releasing more frequently than what the social optimality requires (i.e., **share the burden**)

Implications of Results

- Do not use **cost sharing** and **liability** together if the purpose is to achieve coordination with minimum additional cost on the vendor side
- Patch management tools are available to improve the effectiveness of patch management process
  - BigFix Enterprise Suite, Patch Link Update, HFNetChkPro, Microsoft Update, MBSA, WSUS, SMS
  - Reducing both fixed and variable parts of patch update cost
    - Firms update their systems more frequently
    - Liability becomes more attractive than cost sharing

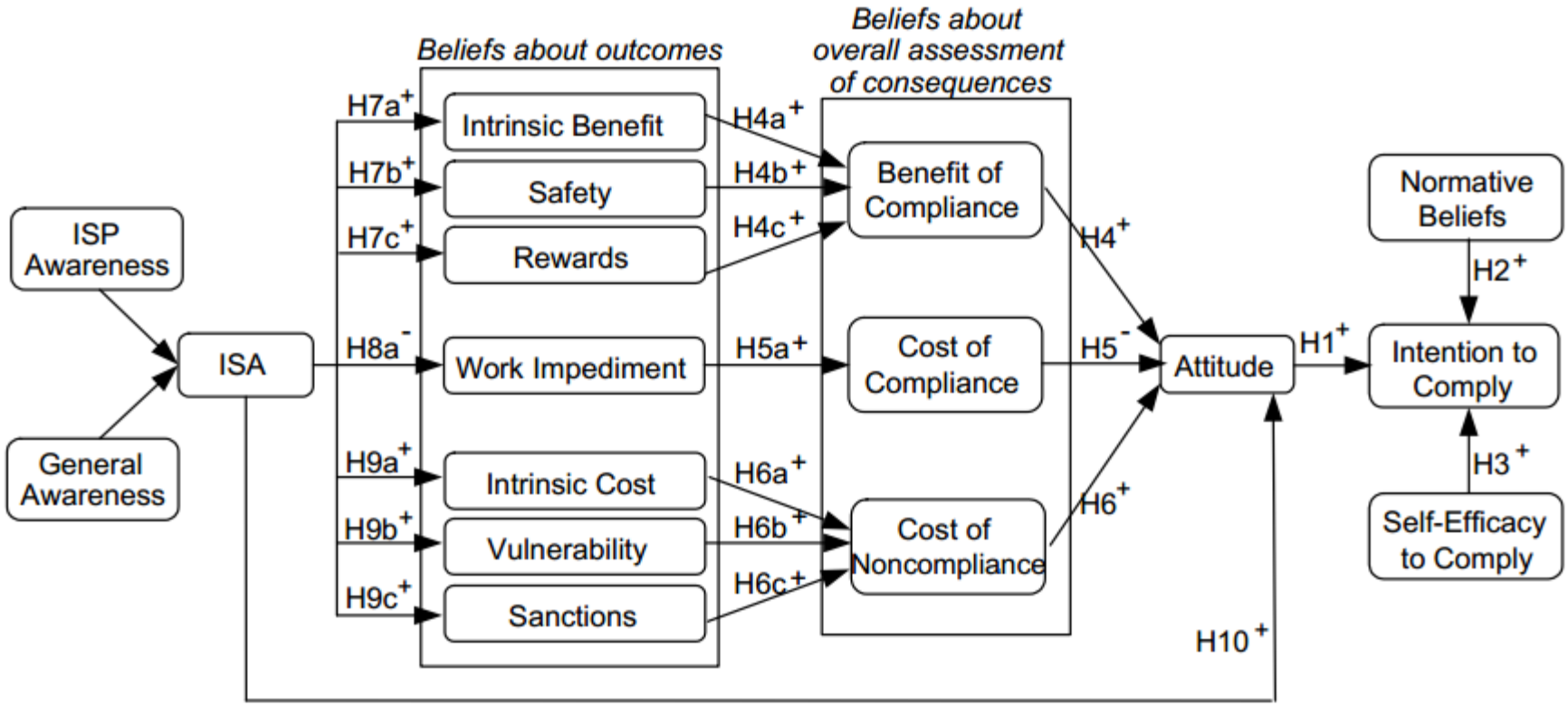
Socio-organizational Aspects of Information Security

- Increasing investment in information security
- Increasing number of security incidents
- Technology-based solutions are crucial but not enough
- Human weaknesses
- Emphasis in security management should go beyond the technical means

Information Security Policy Compliance: An Empirical Study of Rationality-Based Beliefs and Information Security Awareness

- While employees are the weakest link, they can be an asset in security management
- identifying antecedents of their compliance with the information security rules and regulations
- rationality-based beliefs
  - *benefit of compliance*, *cost of non-compliance*, and *cost of compliance*
  - rewards and sanctions are not necessarily the only leverage available to get their employees to attain favorable beliefs about compliance

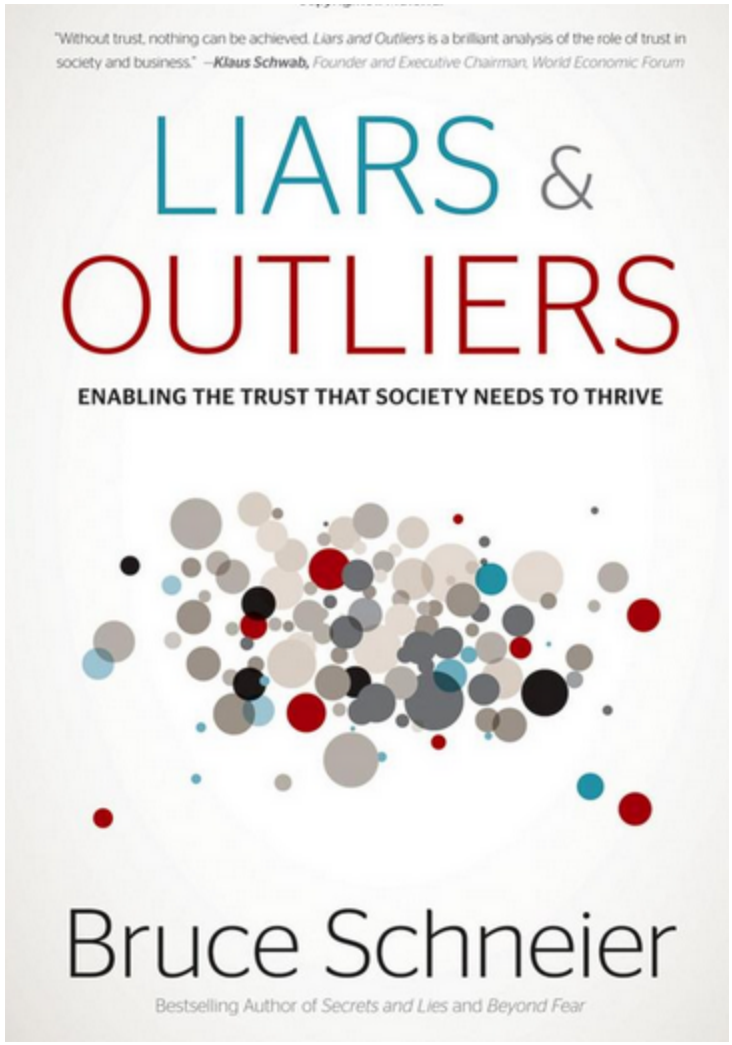
Information Security Policy Compliance: An Empirical Study of Rationality-Based Beliefs and Information Security Awareness



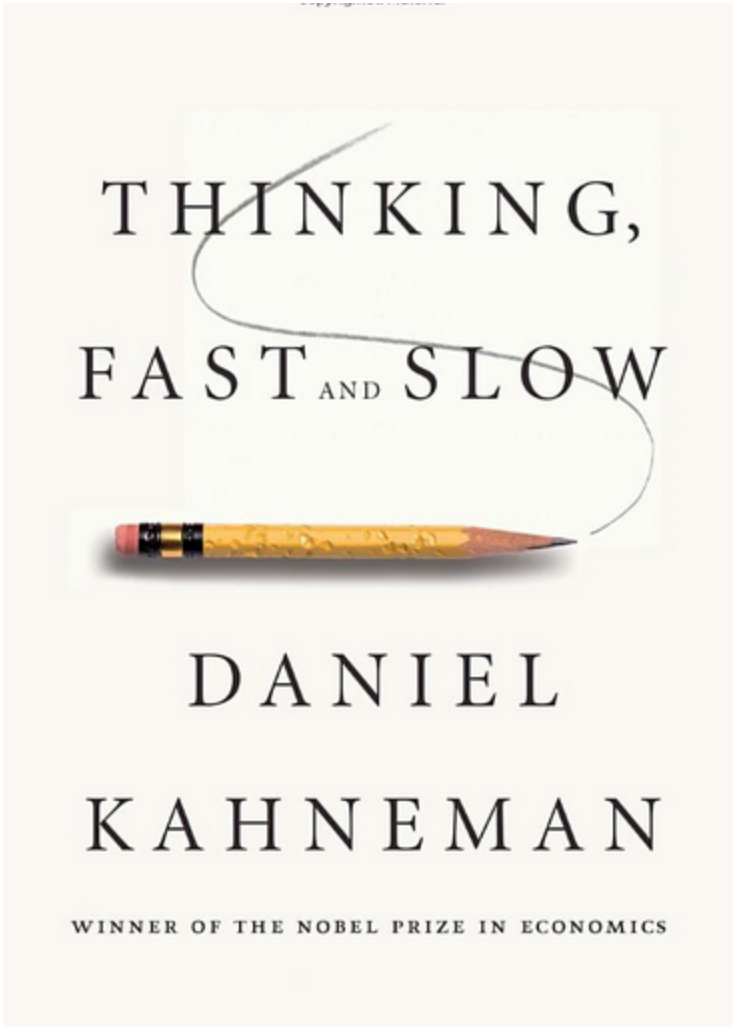
Ongoing Facebook Privacy Studies

- Why do we use apps that potentially violate our privacy?
  - Privacy calculus? Yes, we consider benefits, privacy risks, and control cost. However, higher the benefits, the *less* emphasis that put on privacy risks.
- How do we deal with situations where an app is likely to be quite beneficial but it is likely to pose privacy risks?
  - We deploy coping strategies: problem focused coping (withholding, distortion, safeguard..), emotional focused coping (denial, distancing, divert attention..)

Reading Suggestions



Liars and Outliers
Bruce Schneier



Thinking, Fast and Slow
Daniel Kahneman