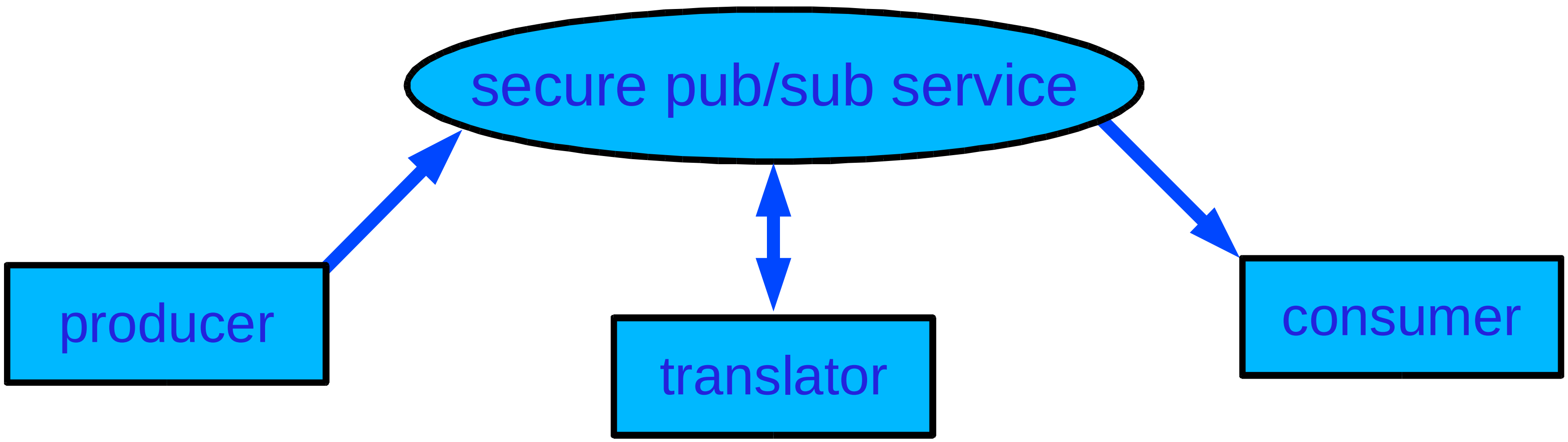
# CODEX – An Application of Distributed Trust

Michael A. Marsh
Information Assurance Institute
Department of Computer Science
Cornell University

joint work with Fred B. Schneider and Lidong Zhou

This work is funded in part by DOD CIPIAF grant F49620-01-1-0312

# Motivating Example




need access control

Common access privileges for publishing or subscribing:

- encrypt with (symmetric) keys
- key distribution equivalent to access control
- check access privileges iff new key issued

# The Problem

critical systems on Internet

- commerce
- power grids
- military

trustworthiness

- unreliable network
- faulty (compromised) processors

$\Longrightarrow$ Distributed Trust

# Outline of Talk

- Distributed Trust
- COrnell Data EXchange (CODEX)
- Composing Systems
- Distributed Blinding

# Outline of Talk

- Distributed Trust
- COrnell Data EXchange (CODEX)
- Composing Systems
- Distributed Blinding

# Assumptions

reasonable to assume:

- fair links

  repeated sends $\Rightarrow$ eventual delivery

  allow wiretapping, message delays

  (can build secure links with PKI)

- asynchronous

  free of assumptions about timing

- compromised processors (fewer than 1/3)

  Byzantine (arbitrary) failures

  all might collude with adversary

# Replication

availability – often overlooked

- unavailable system/data not useful
- potentially dangerous

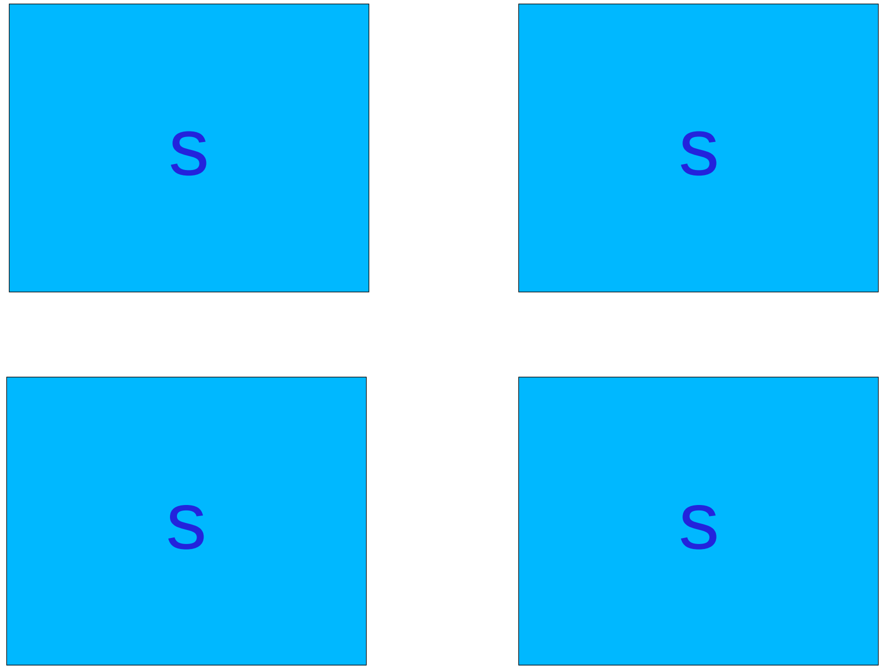
replication

- data in multiple locations
- lose one replica, others still available

design system to tolerate up to $t$ failures

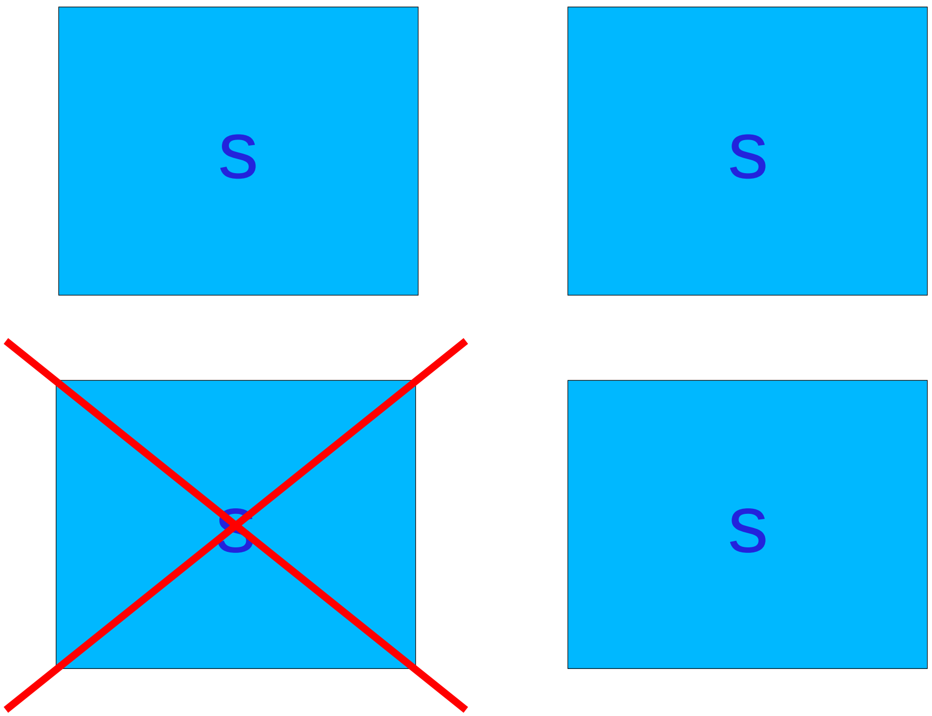
# Secret Sharing

replication of secrets:



# Secret Sharing

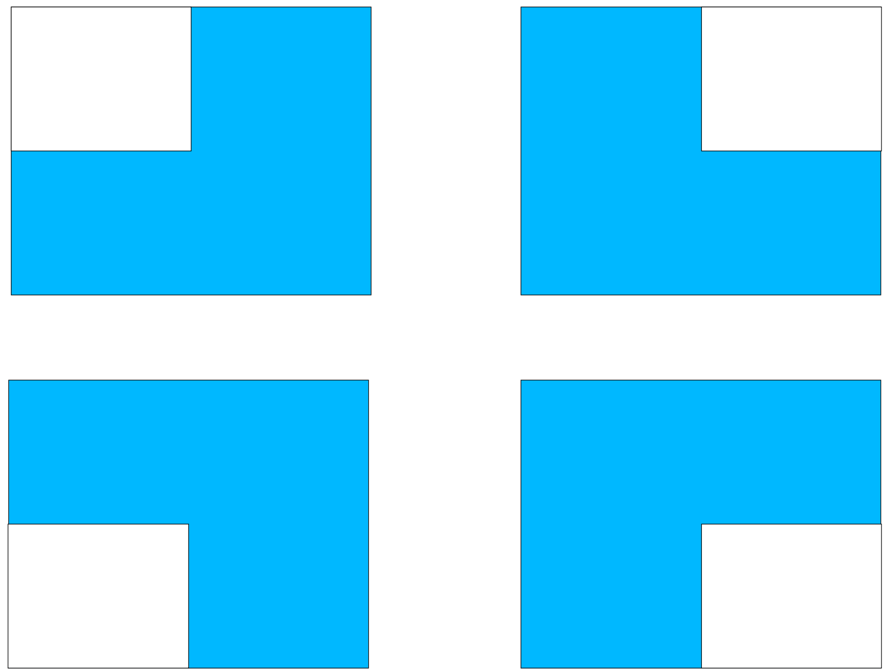
replication of secrets:




one compromise ⇒ secret leaked

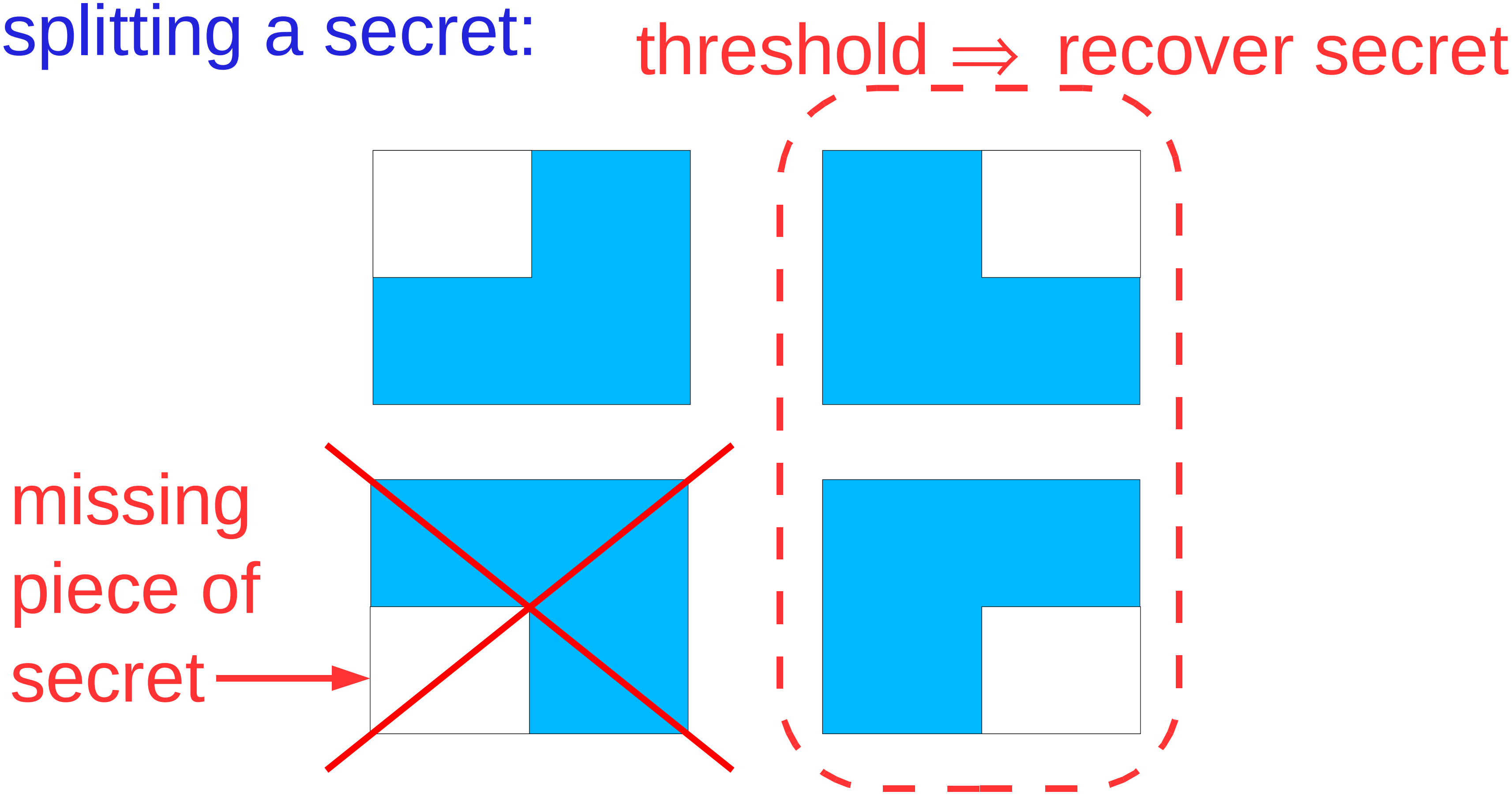
# Secret Sharing

splitting a secret:



# Secret Sharing

splitting a secret:



# Mobile Adversaries

recover compromised processors

$\Longrightarrow$ disclosed shares still disclosed

mobile adversary [OY91]

- changes targets
- never more than t compromises
- eventually collect >t shares

# Proactive Recovery

intrusion detection difficult, periodically assume:

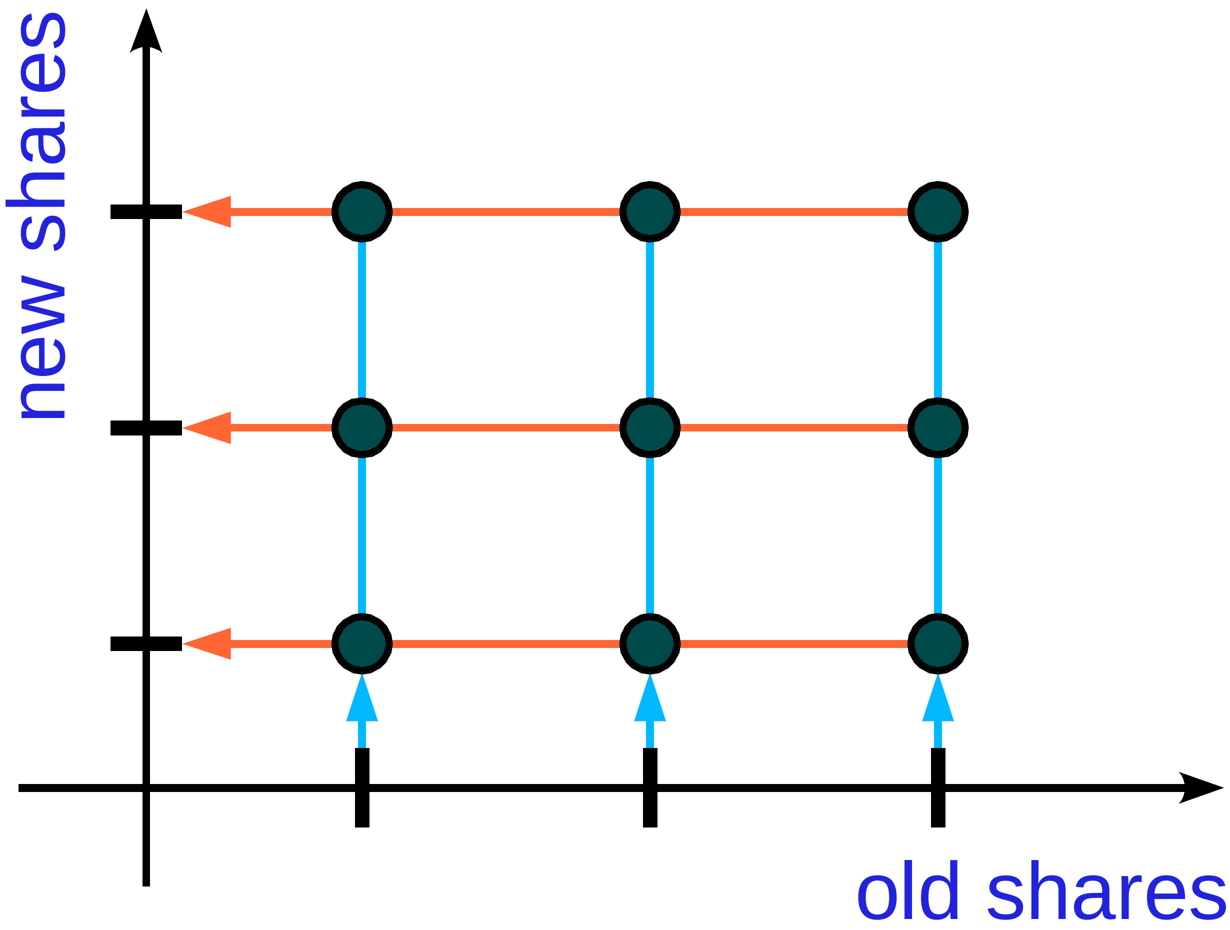
processor might be in corrupted state

- reboot from up-to-date clean media

secrets might have been disclosed

- processor's private key
  - $\Longrightarrow$ generate new public/private key pair
- shares of split secrets (mobile adversary)
  - $\Longrightarrow$ new shares for same secret
    (proactive secret sharing) [HJKY95]

# Proactive Secret Sharing



# Distributed Trust

building trustworthy systems

- replication for availability
- secret sharing for confidentiality
- proactive recovery for long-term security
  (PSS, rekey, reboot from clean media)

existing implementations:
COCA [ZSvR02], SINTRA [CP02]

we have added:
data storage, distribution

# Outline of Talk

- Distributed Trust
- COrnell Data EXchange (CODEX)
- Composing Systems
- Distributed Blinding

# CODEX – Distributed Trust in a Key Distribution Service

stores clients' secrets (keys)

access control

confidentiality of secrets

transparent – client sees single "server", does not see *changes* to service

could also layer on top of OceanStore [KBC+00]

# Maintaining Confidentiality of Client Data

unauthorized clients

- access control lists

compromised servers

- encryption with service public key
- private key $\rightarrow$ shared secret
    - threshold decryption/signature
    - does not use private key explicitly

retrieval requires decryption

# CODEX Operations

create_key: associates ownership and access policies with a name

create_key,name,owner,policies

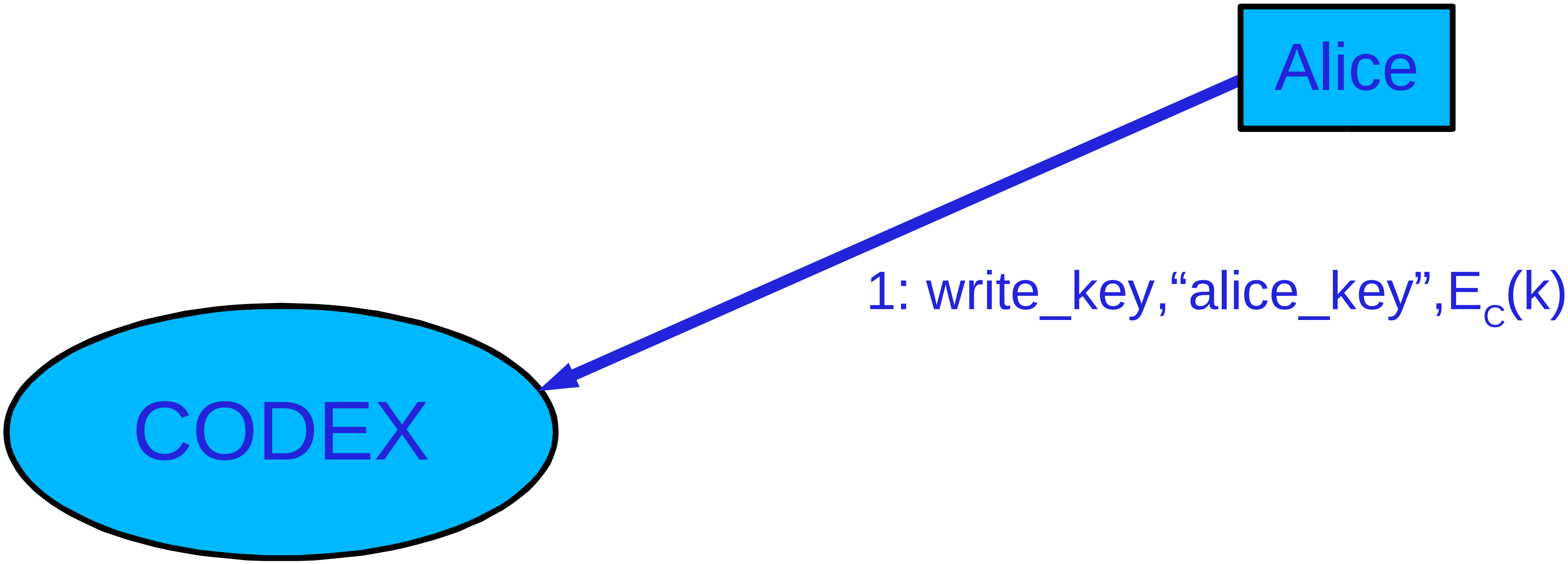
write_key: stores a value for a name, if access policy satisfied

write_key,name,$E_C$(k),auth

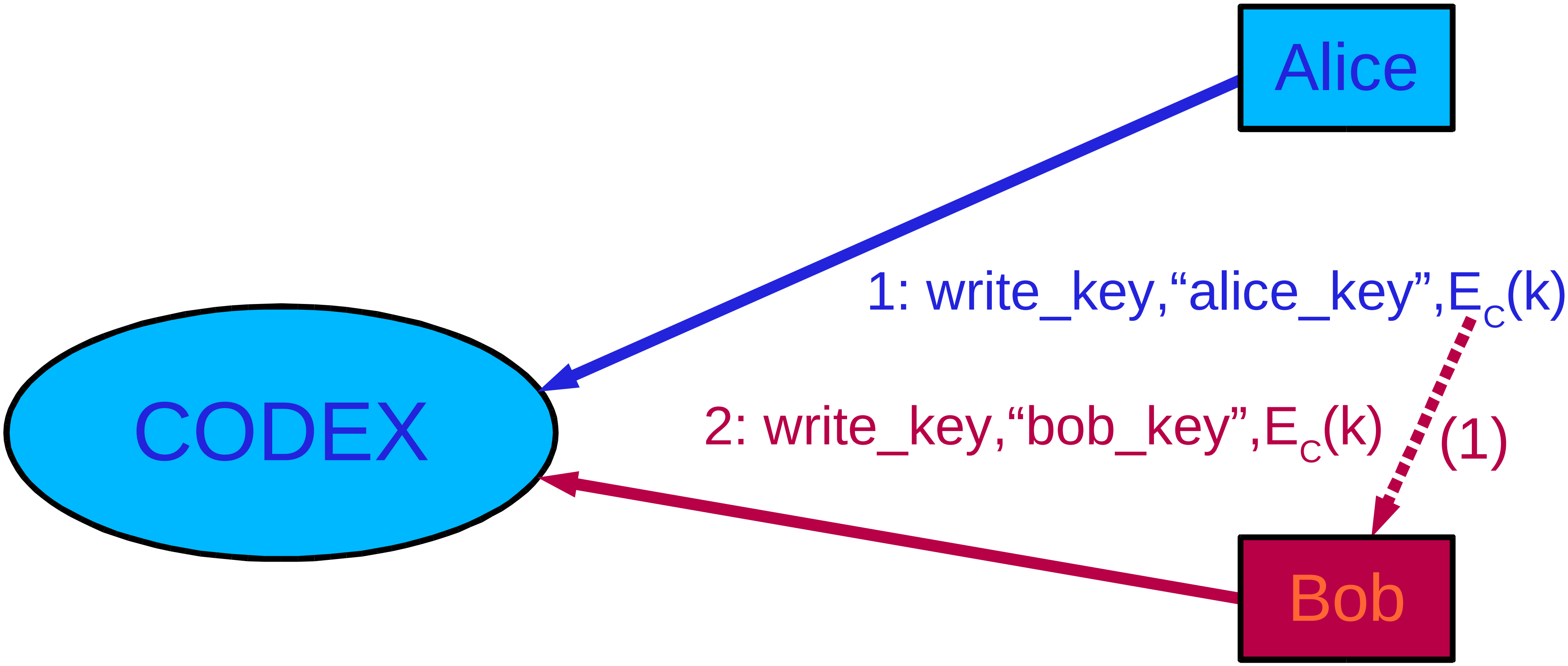
read_key: retrieves the value for a name, if access policy satisfied

read_key,name,auth

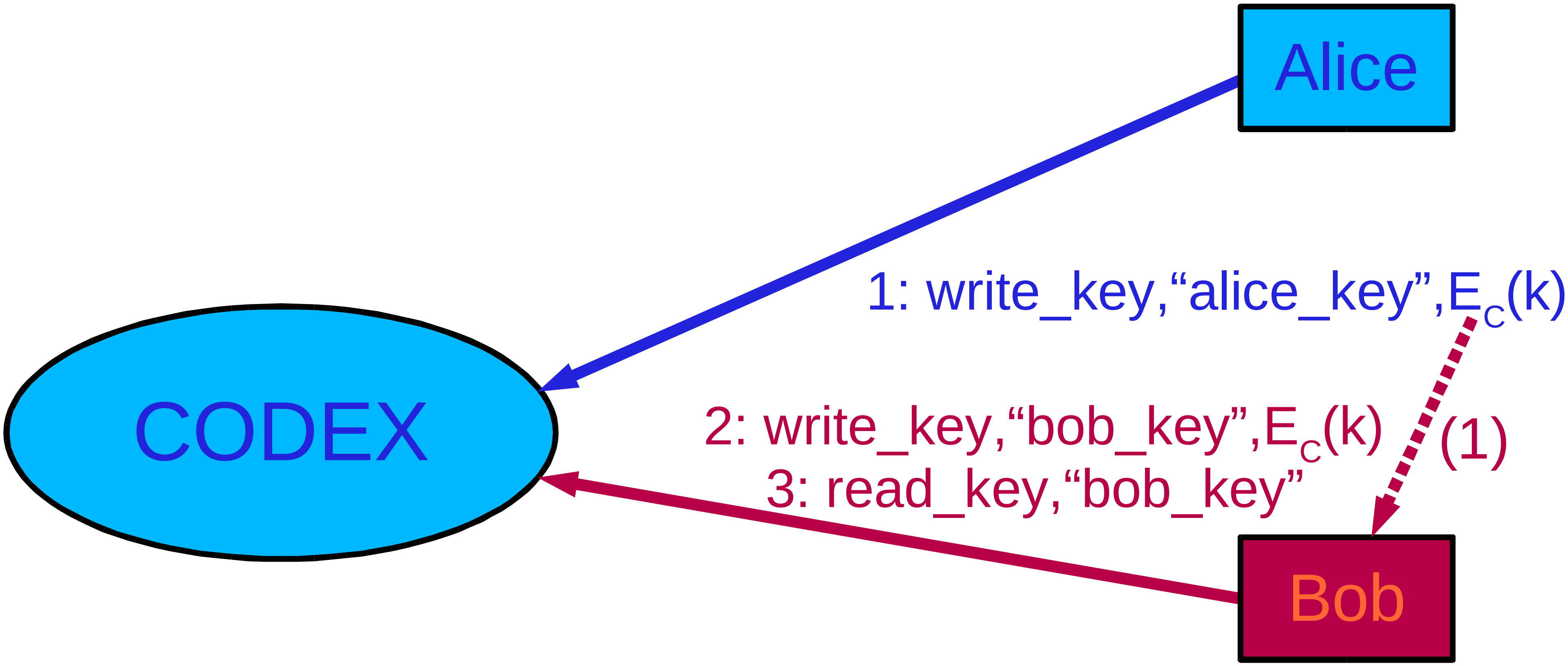
# Preventing Known Ciphertext Attacks



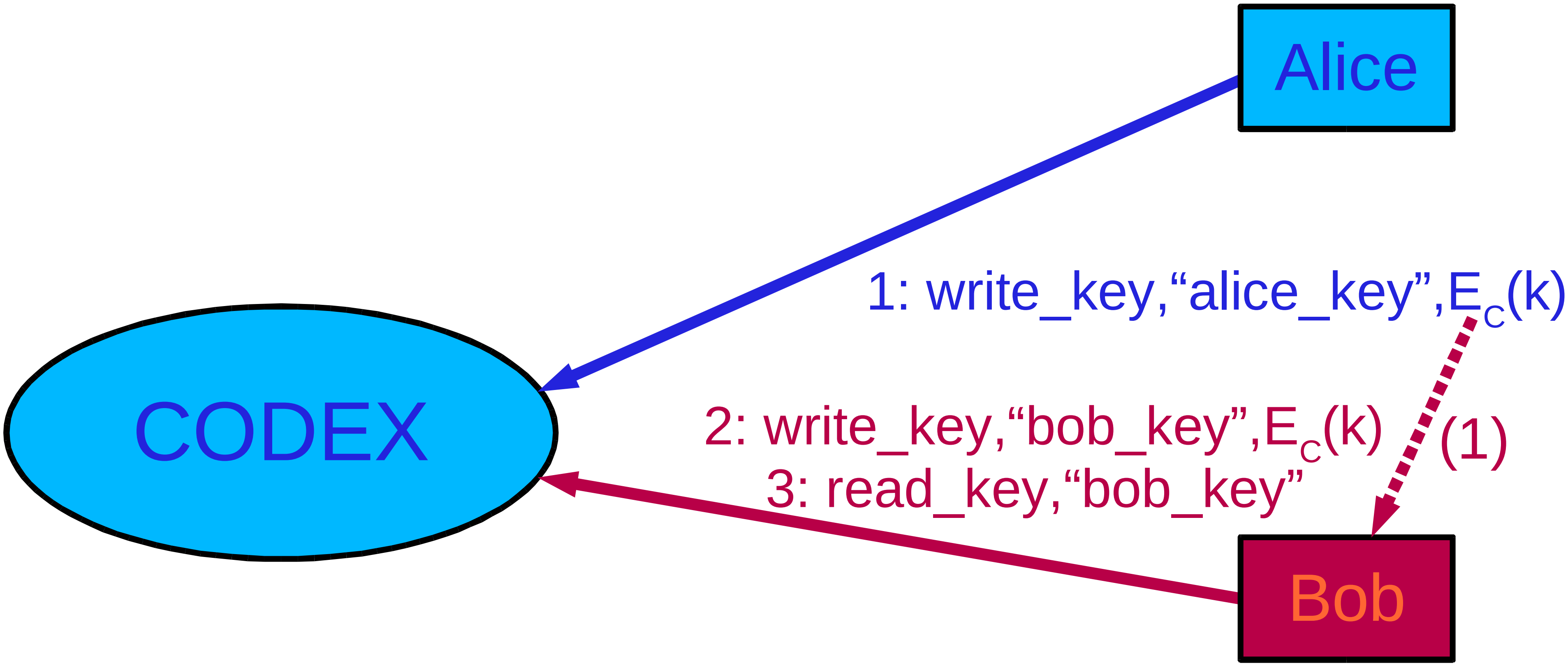
# Preventing Known Ciphertext Attacks



# Preventing Known Ciphertext Attacks



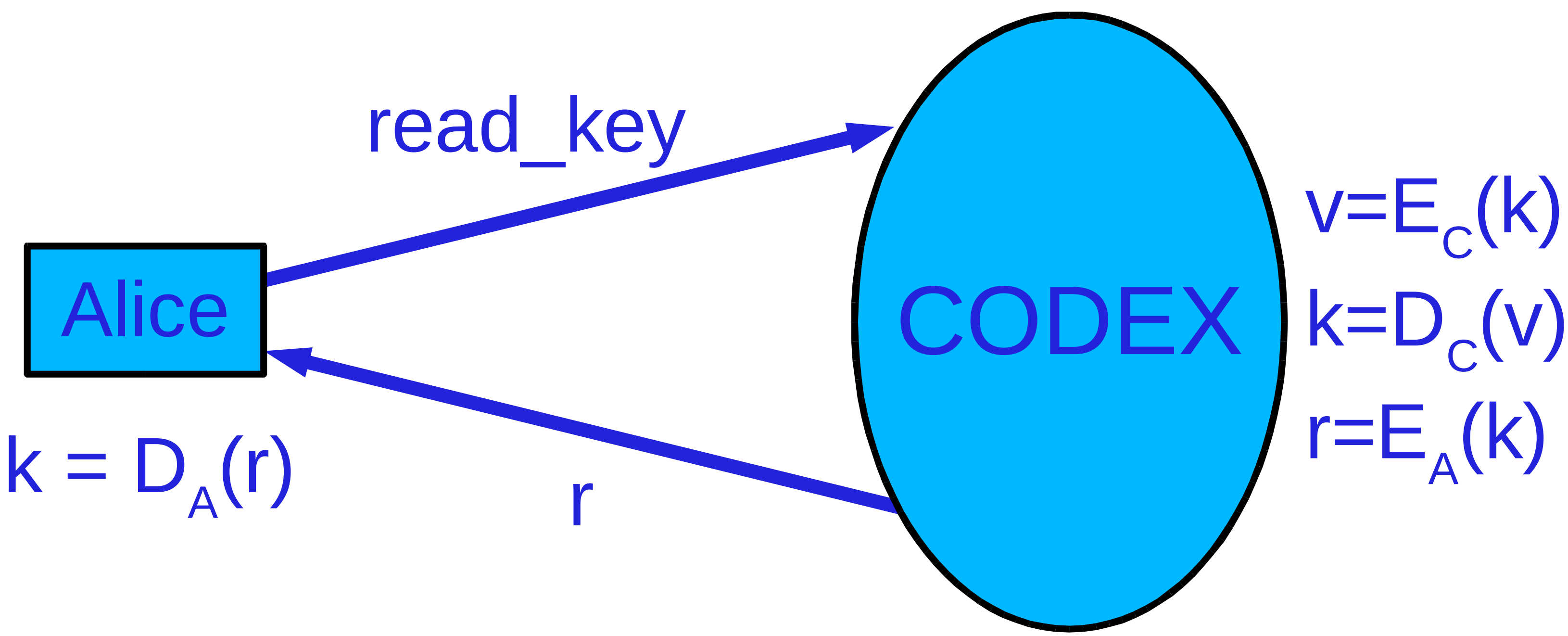
# Preventing Known Ciphertext Attacks



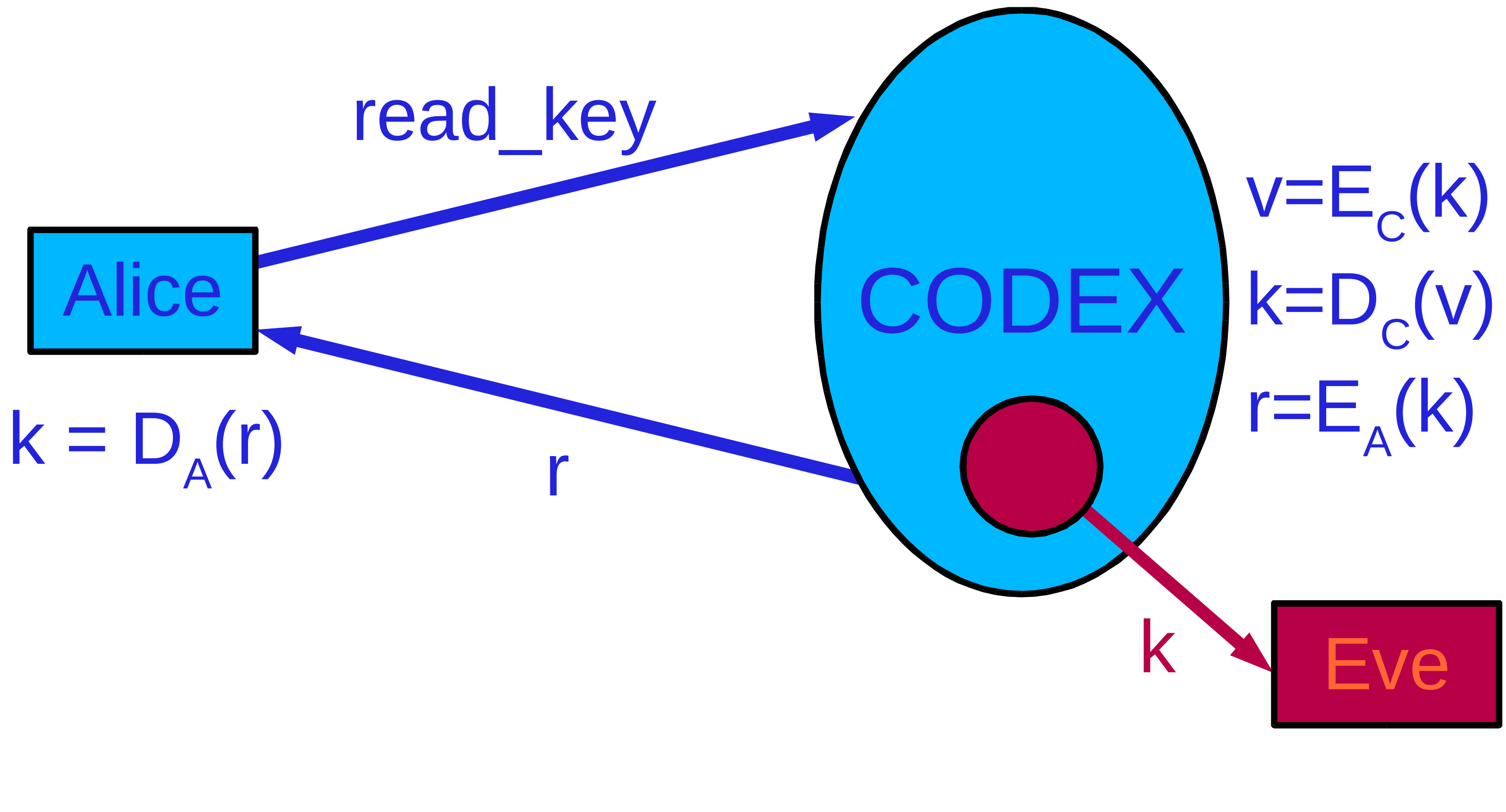

non-interactive, non-reusable ZKP that k known

(eg, Schnorr signature on ciphertext)

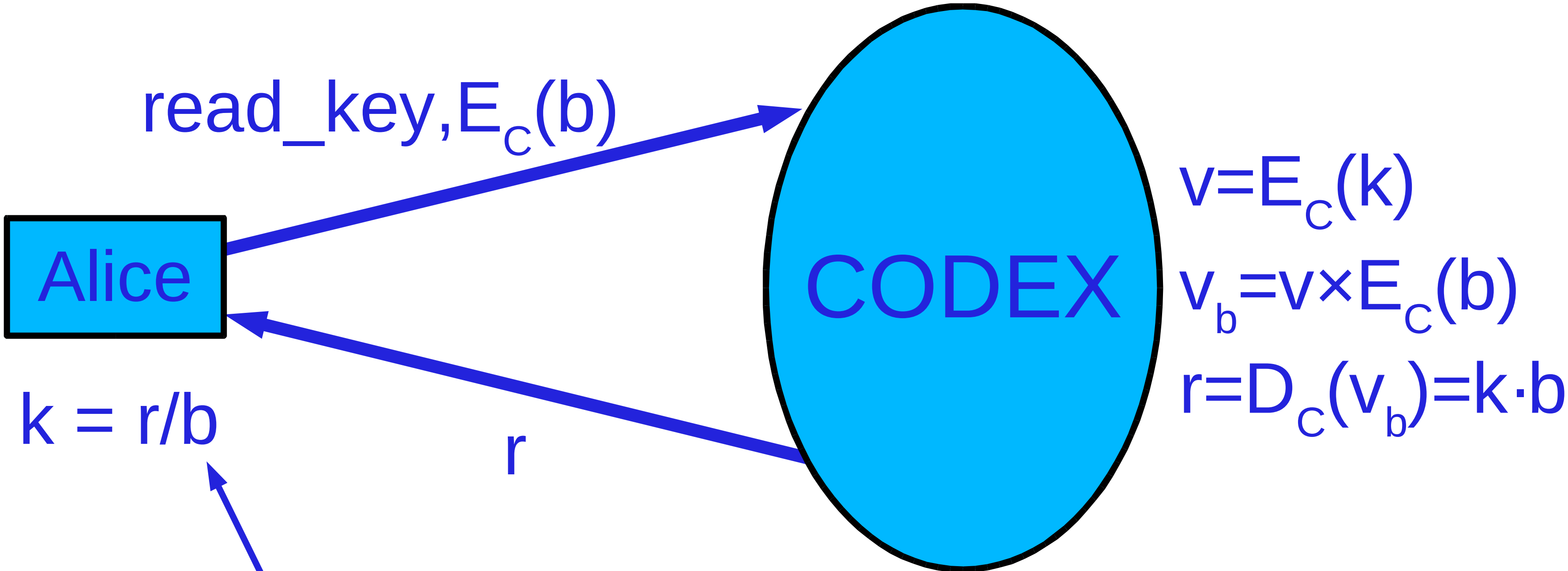
# Protecting Secrets During Decryption



# Protecting Secrets During Decryption



# Protecting Secrets During Decryption




known only to Alice

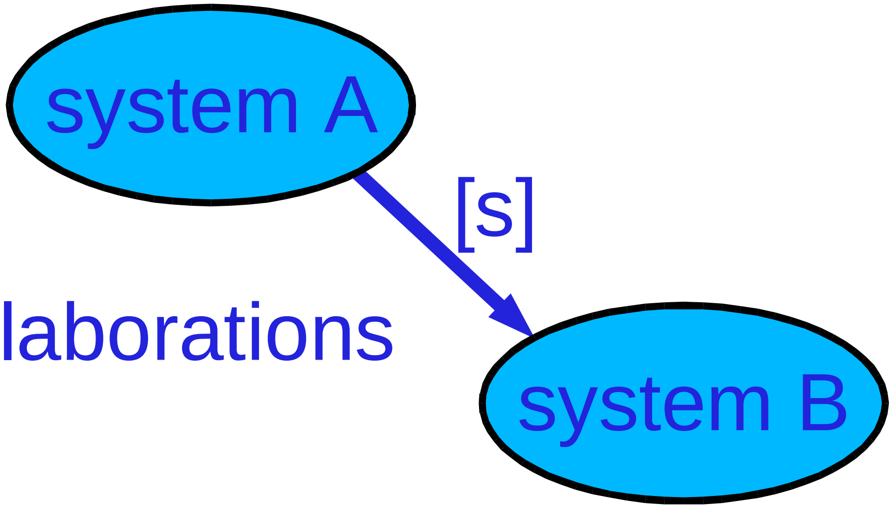
blind decryption

[Chaum83]

# Outline of Talk

- Distributed Trust
- COrnell Data EXchange (CODEX)
- Composing Systems
- Distributed Blinding

# Composing Systems with Distributed Trust




- business/military collaborations
- exploiting locality
- “trusted” hosts in peer-to-peer

each system employs distributed trust

transparent vs. non-transparent

# Tradeoffs: Non-Transparent vs. Transparent

non-transparent:

simple data propagation (secure links)

clients need public keys of all servers

exposes fault tolerance structure

# Tradeoffs: Non-Transparent vs. Transparent

transparent:

one public key

looks like single server

threshold signatures (extra communications)

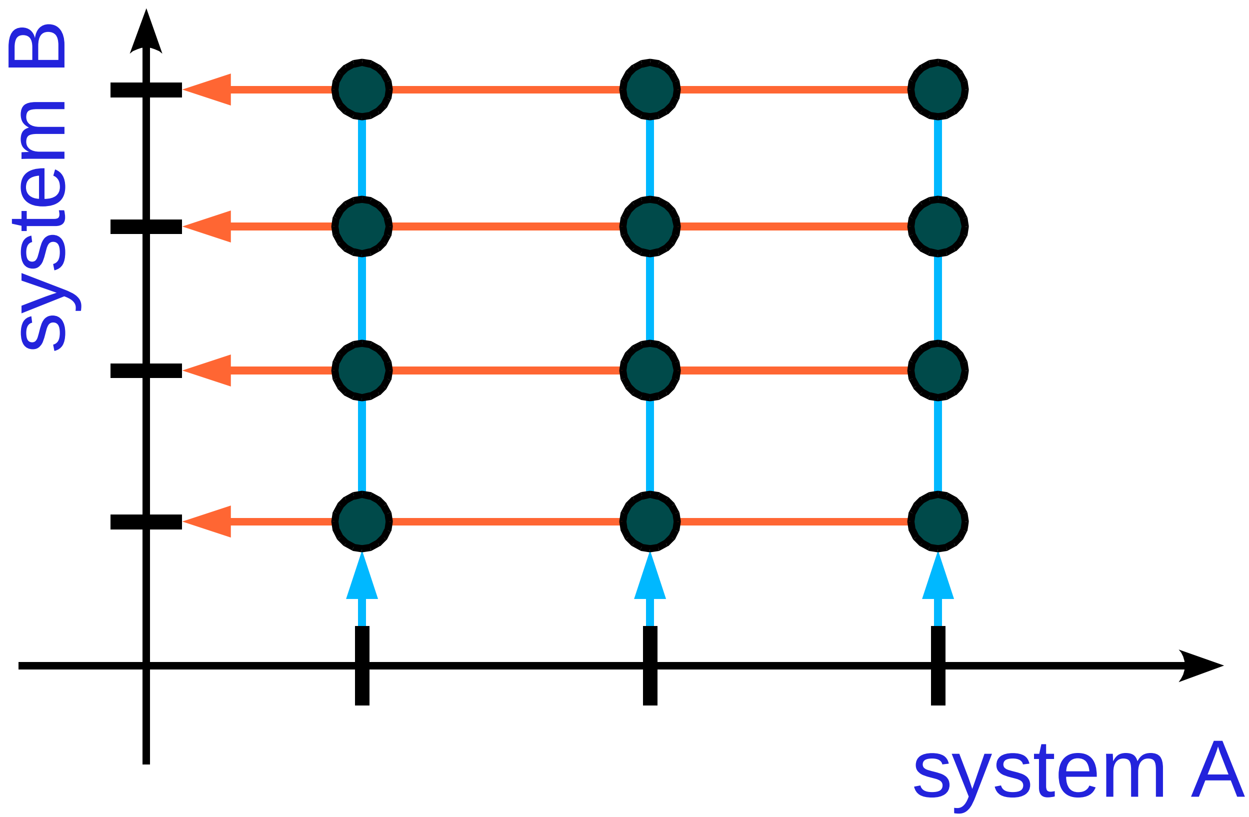
compromised delegates

transferring secrets more complicated

# Non-Transparent System Composition

data $\rightarrow$ shared secrets

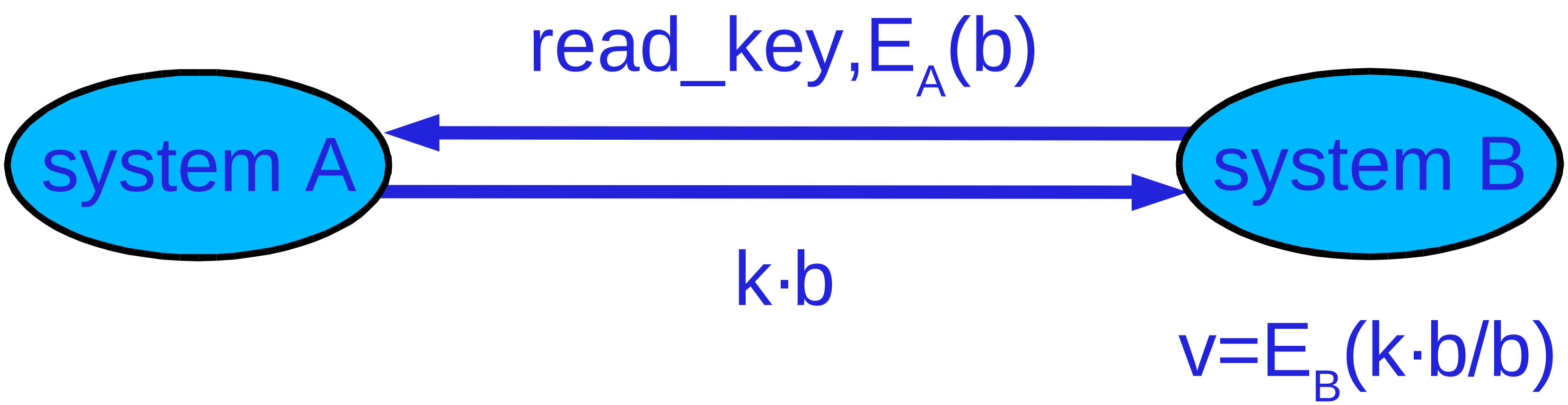
redistribution similar to PSS [DJ97,WWW02]



# Transparent System Composition

data $\rightarrow$ public-key encrypted

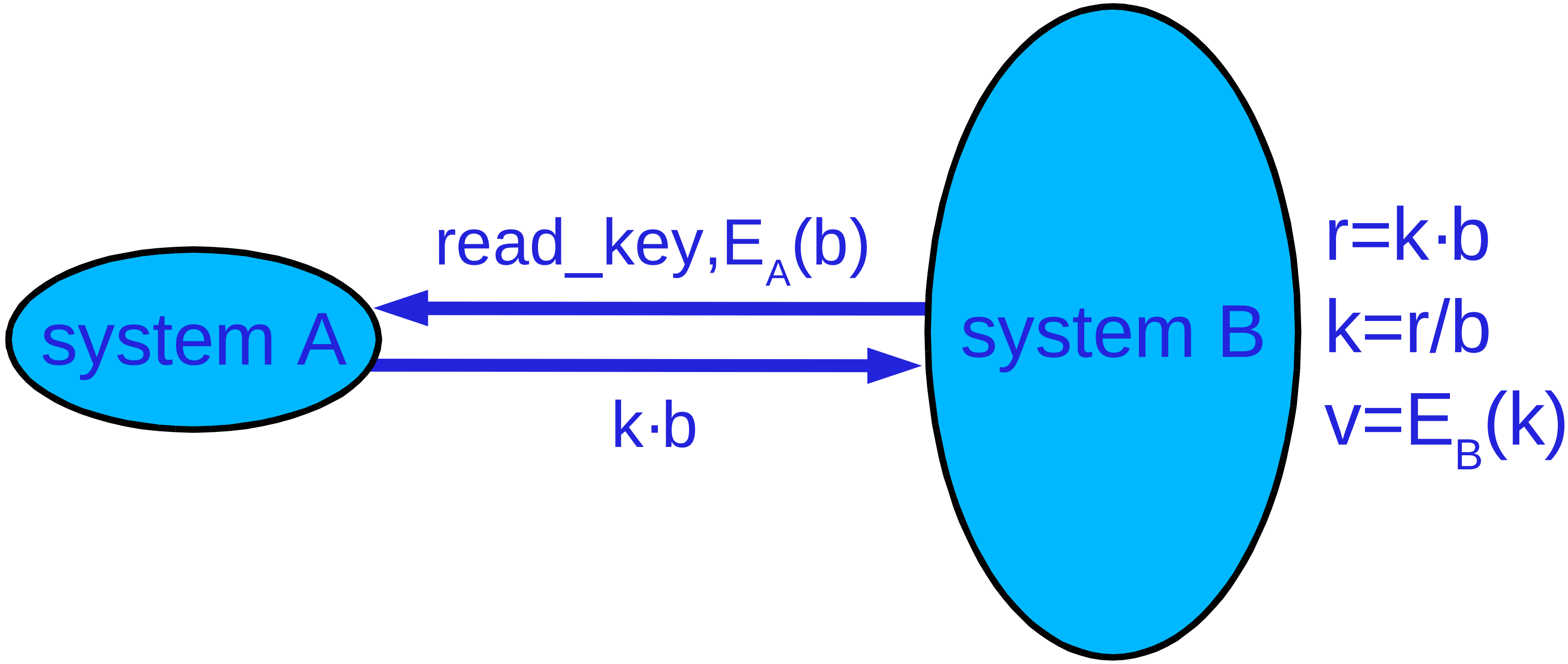
redistribution via blinding, re-encryption



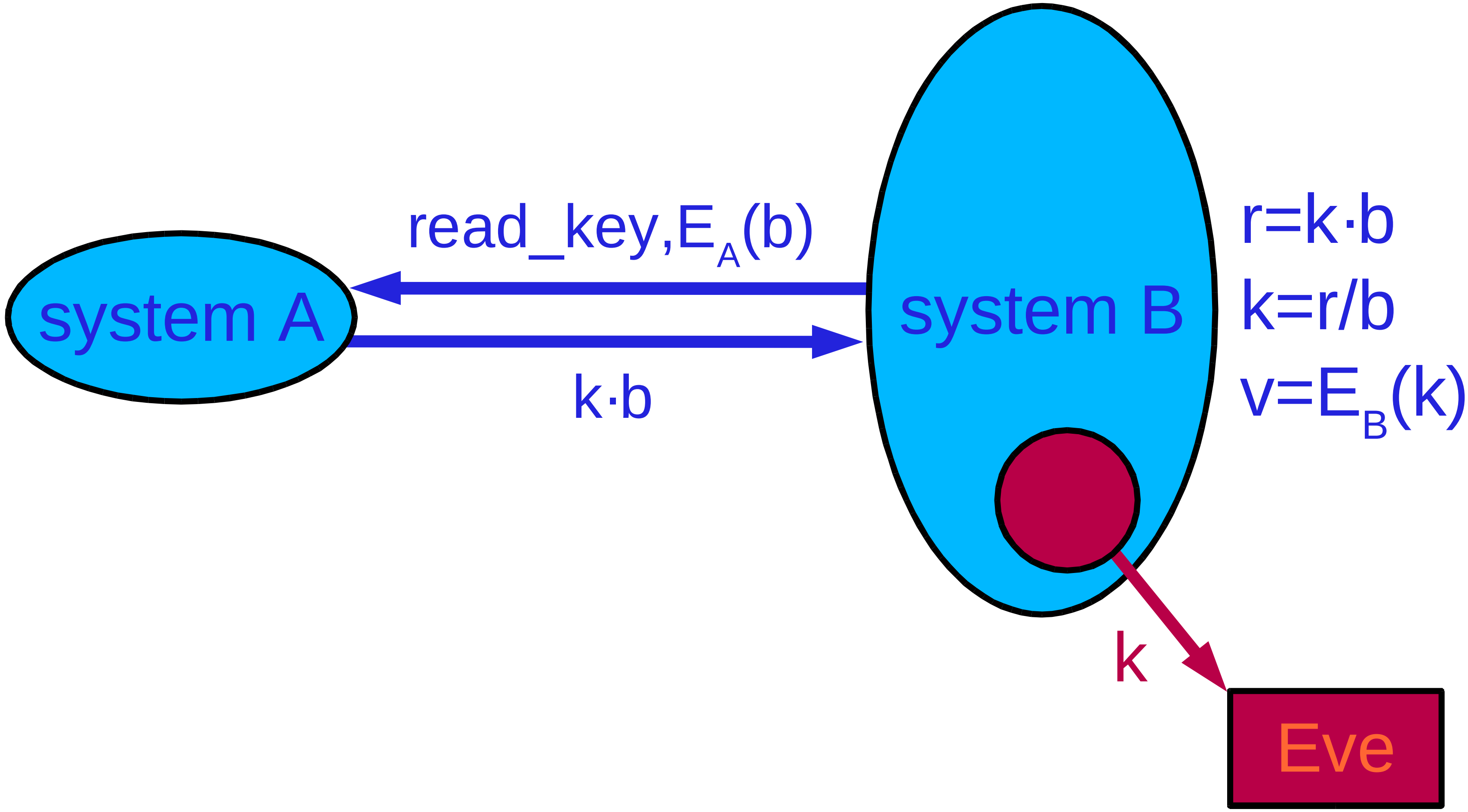
# Outline of Talk

- Distributed Trust
- COrnell Data EXchange (CODEX)
- Composing Systems
- Distributed Blinding

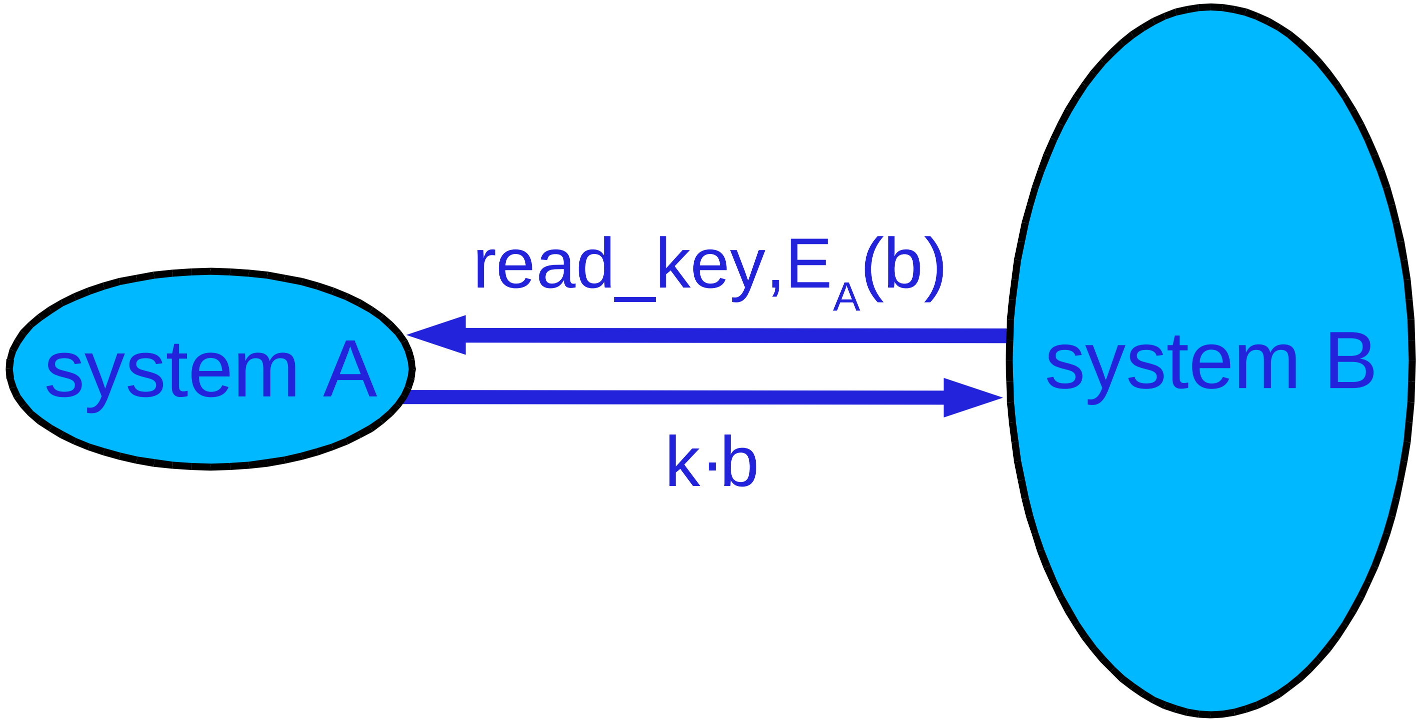
# Protecting Secrets During Unblinding



# Protecting Secrets During Unblinding



# Protecting Secrets During Unblinding




$r = k \cdot b$

$v = r \otimes \underbrace{E_B(b^{-1})}_{E_B(k)}$

distributed blinding

many cryptosystems $\Rightarrow$ simple to compute $E_B(b^{-1})$ from $E_B(b)$

# Distributed Blinding

goal: Generate ciphertexts $c_A = E_A(b)$, $c_B = E_B(b)$

requirements:

- consistency
- confidentiality
- randomness

solution: construct $c_A$ and $c_B$ simultaneously from encrypted partial blinding factors $b_i \rightarrow E_A(b_i), E_B(b_i)$

$b = \prod_i b_i \Rightarrow c_A = \prod_i E_A(b_i)$, $c_B = \prod_i E_B(b_i)$

# Maintaining Consistency of b

faulty processor:

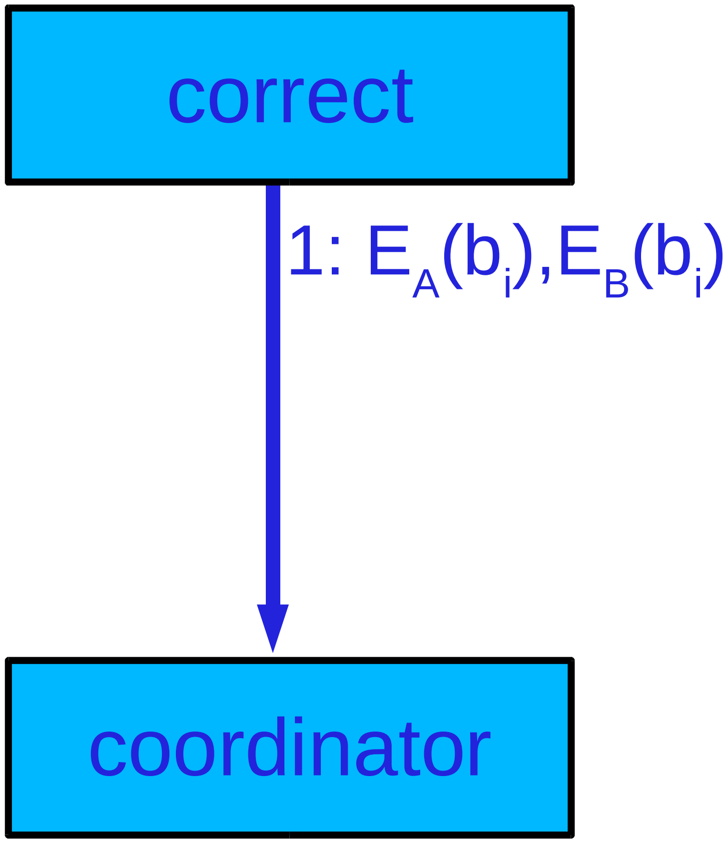
$E_A(b_i)$, $E_B(b_i')$ inconsistent ($b_i \neq b_i'$)

proof of correctness

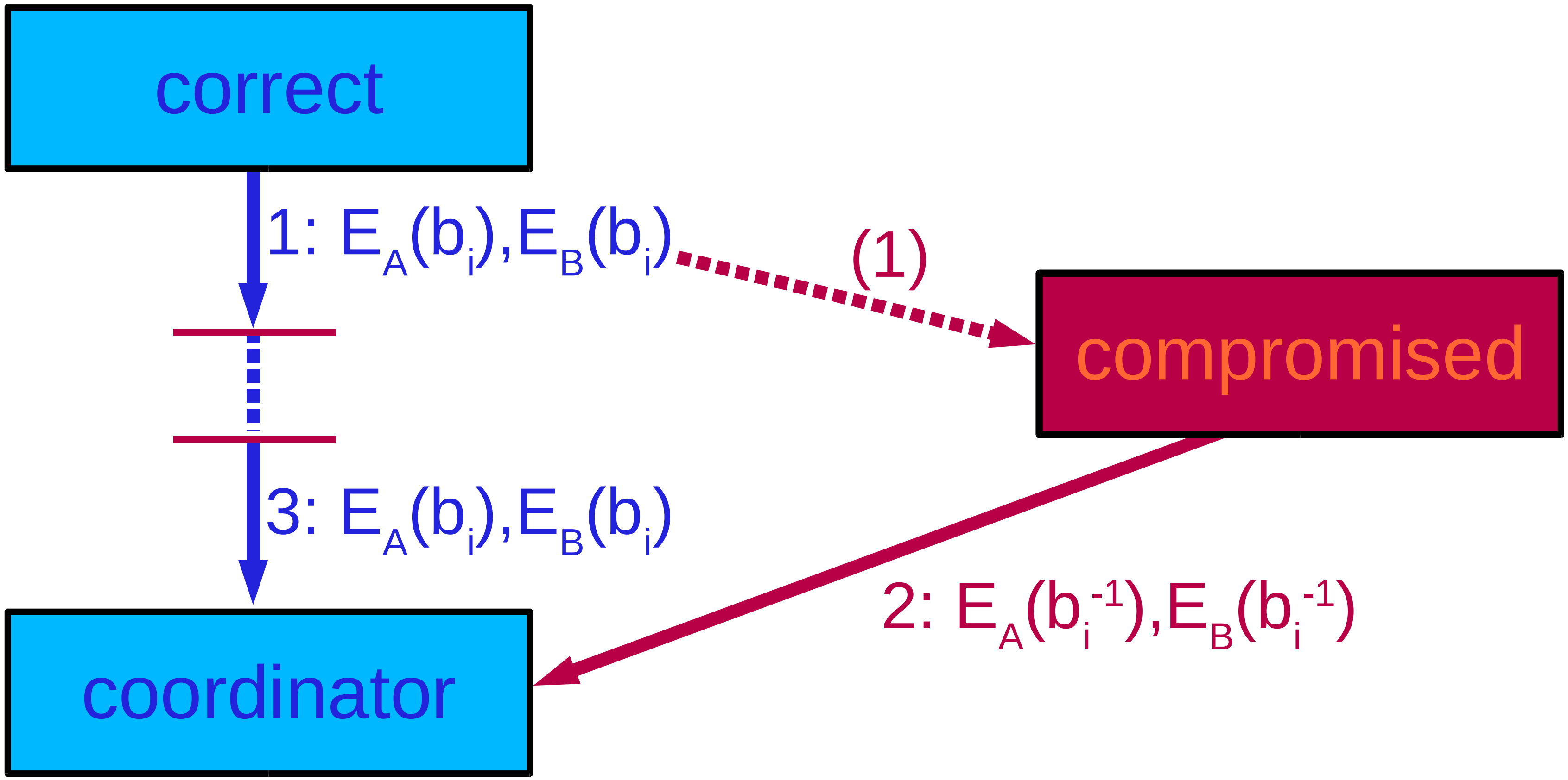
translation certificates [Jakobsson99]

self-verifying contributions

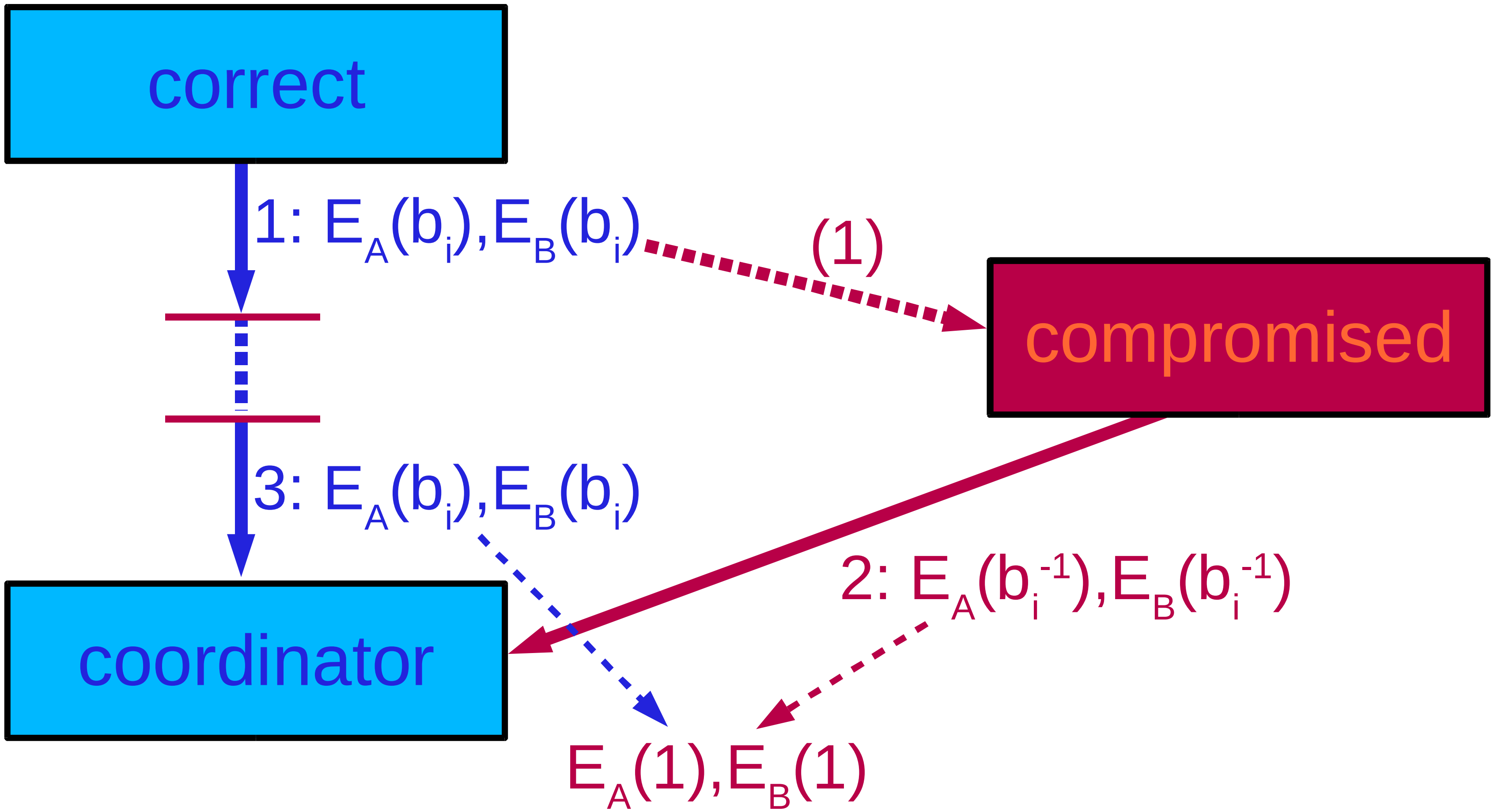
# Maintaining Confidentiality and Randomness of b



# Maintaining Confidentiality and Randomness of b

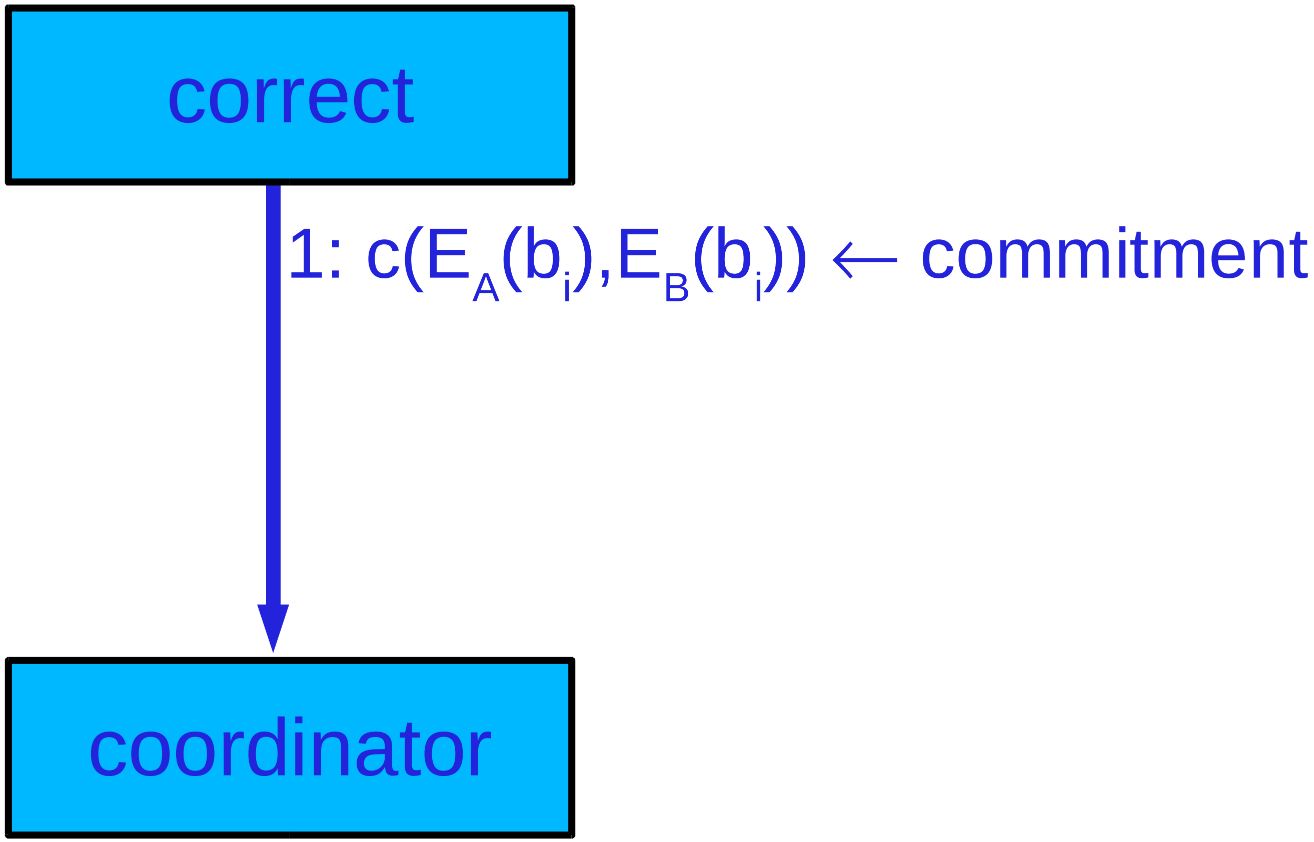


# Maintaining Confidentiality and Randomness of b

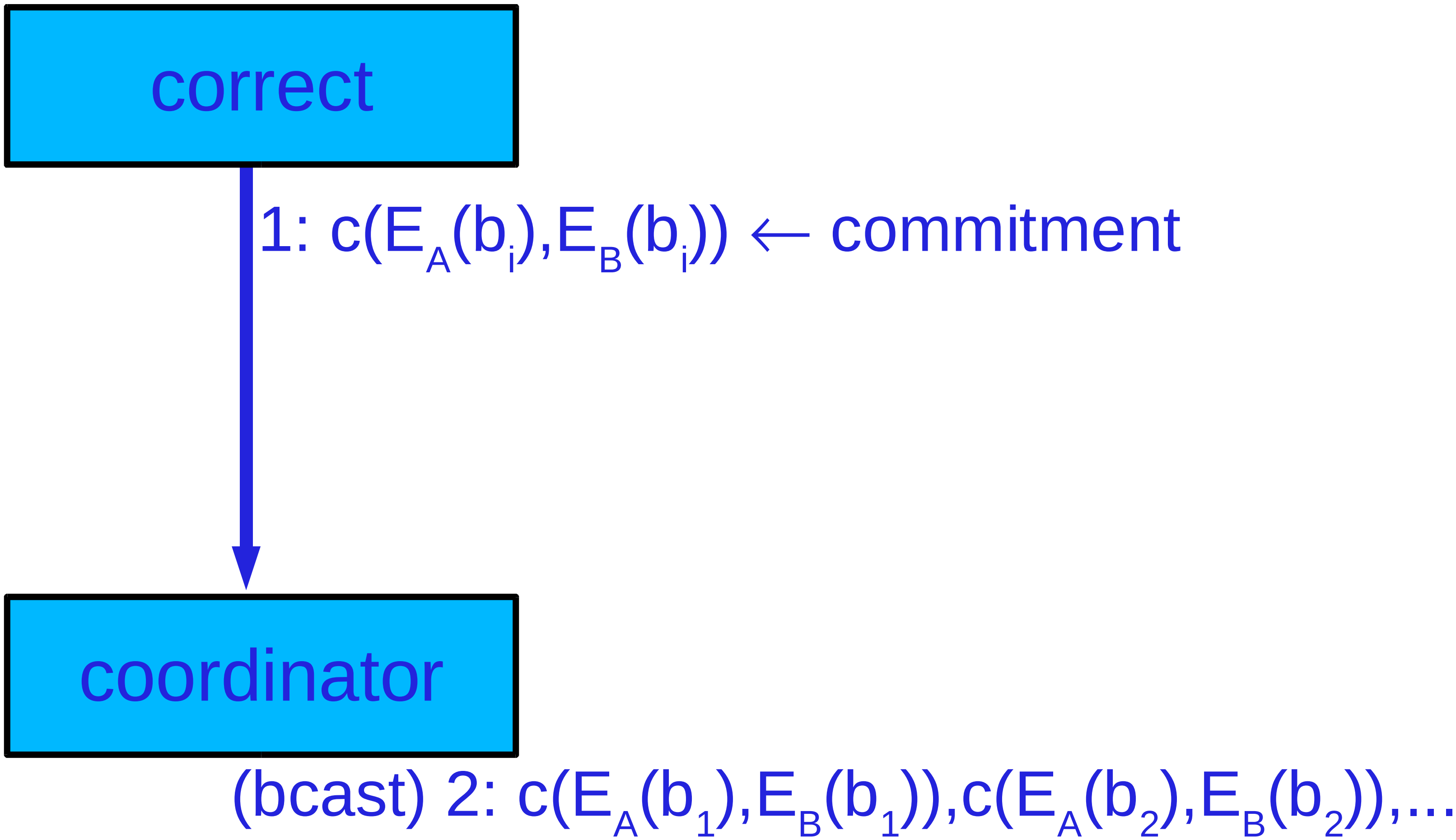



$b_i$ confidential, b neither confidential nor random

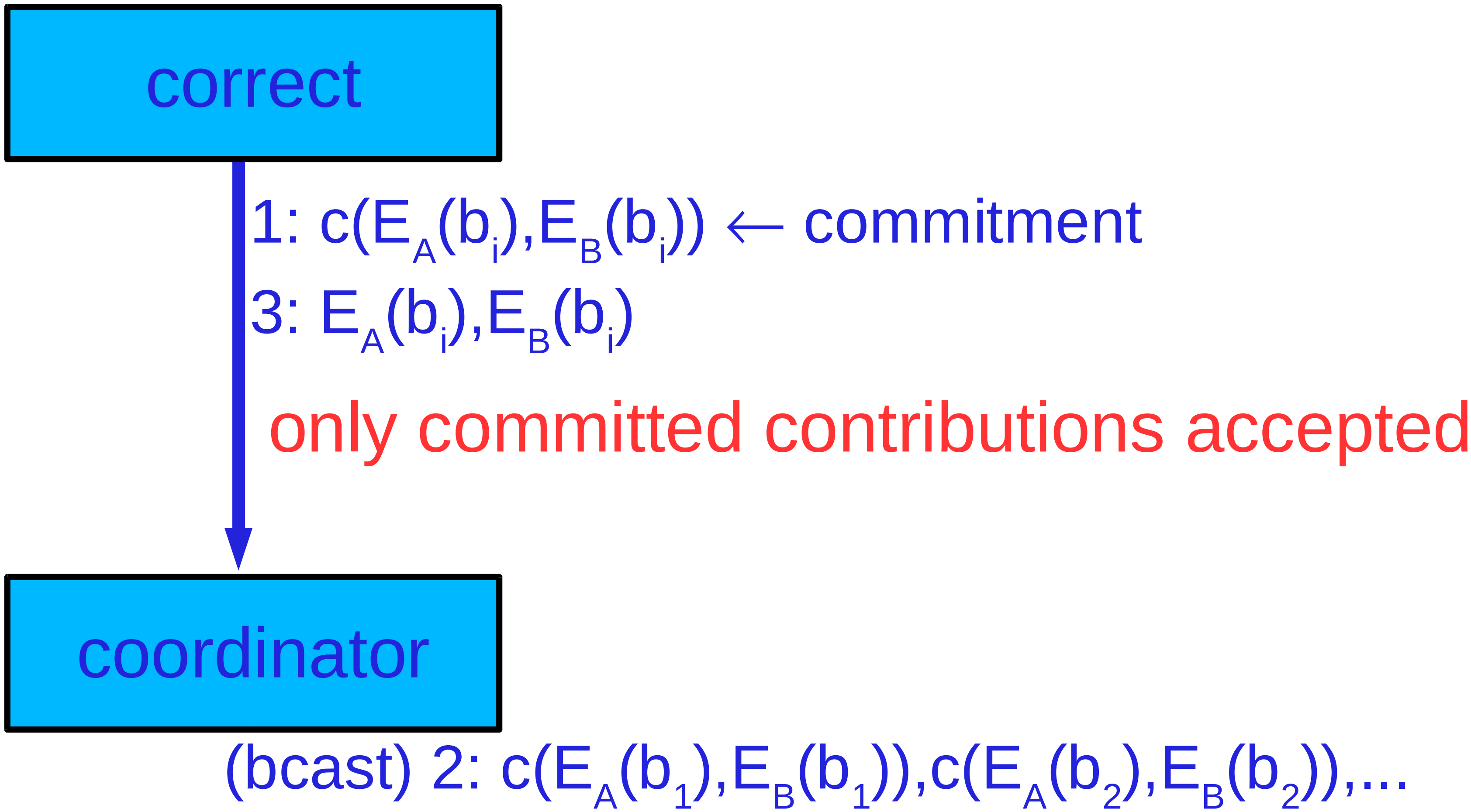
# Maintaining Confidentiality and Randomness of b



# Maintaining Confidentiality and Randomness of b



# Maintaining Confidentiality and Randomness of b



# Summary

trustworthiness is important
network, individual processors not trustworthy

contributions:

- CODEX key distribution service
  (implementation in progress)
- distributed blinding for composing systems
  (protocol developed, working on proof
  of security properties)

# References


**[Chaum83]** D. Chaum, “Blind Signatures for untraceable payments”, *Crypto '82*, 199-203, 1983

**[CP02]** C. Cachin and J. Poritz, "Secure intrusion-tolerant replication on the Internet”, *DSN-2002*, 167-176, 2002

**[DJ97]** Y. Desmedt and S. Jajodia, “Redistributing secret shares to new access structures and its applications”, tech report ISSE TR-97-01, George Mason University, 1997

**[HJKY95]** A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung, “Proactive secret sharing or: How to cope with perpetual leakage”, *Crypto '95*, LNCS 963:457-469, 1995

**[Jakobsson99]** M. Jakobsson, “On quorum controlled asymmetric proxy re-encryption”, *PKC '99*, LNCS 1560:112-121, 1999

**[KBC+00]** J. Kubiatowicz *et. al.*, “OceanStore: An Architecture for Global-scale Persistent Storage”, *Proceedings of ACM ASPLOS 2000*, 190-201, 2000

**[OY91]** R. Ostrovsky and M. Yung, “How to withstand mobile virus attacks”, *Proceedings of the 10th ACM Symposium on Principles of Distributed Computing*, 51-59, 1991

**[WWW02]** T.M. Wong, C. Wang, and J.M. Wing, “Verifiable secret redistribution for archive systems”, *Proceedings of the First International Security in Storage Workshop*, 2002

**[Zhou01]** L. Zhou, *Towards fault-tolerant and secure on-line services*, Ph.D. thesis, Cornell University, 2001

**[ZSvR02]** L. Zhou, F.B. Schneider, R. van Rennesse, “COCA: A secure distributed on-line certification authority”, *ACM Transactions on Computer Systems*, 20(4):329-368, 2002