

Fall 2008: CS442
Intro. to Computer Security

Vinod Ganapathy

Lecture 23

Virtual machines and anonymity

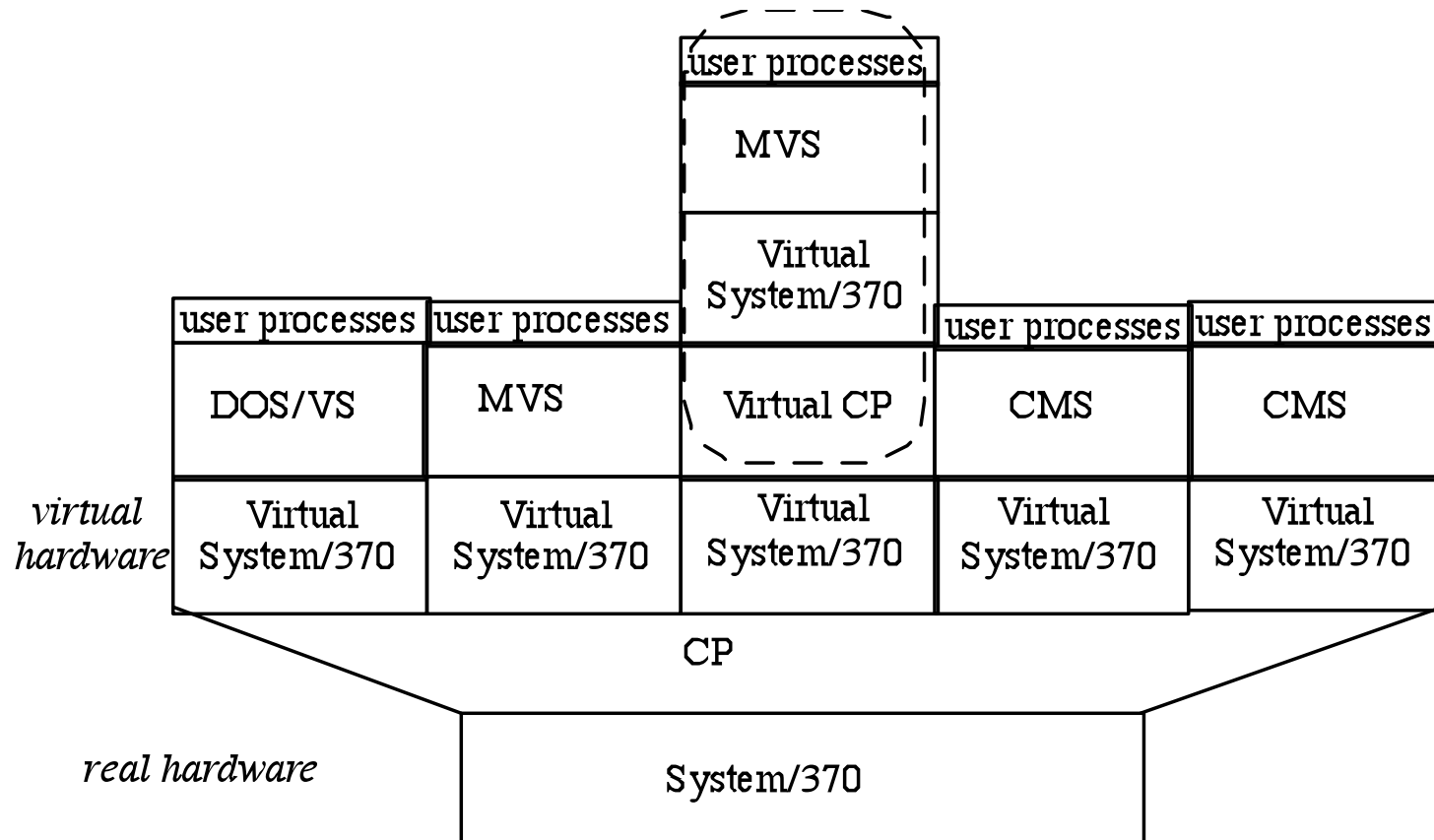
Overview

- Virtual Machine Structure
- Virtual Machine Monitor
 - Privilege
 - Physical Resources
 - Paging

What Is It?

- *Virtual machine monitor* (VMM) virtualizes system resources
 - Runs directly on hardware
 - Provides interface to give each program running on it the illusion that it is the only process on the system and is running directly on hardware
 - Provides illusion of contiguous memory beginning at address 0, a CPU, and secondary storage to *each* program

Example: IBM VM/370



Adapted from Dietel, pp. 606–607

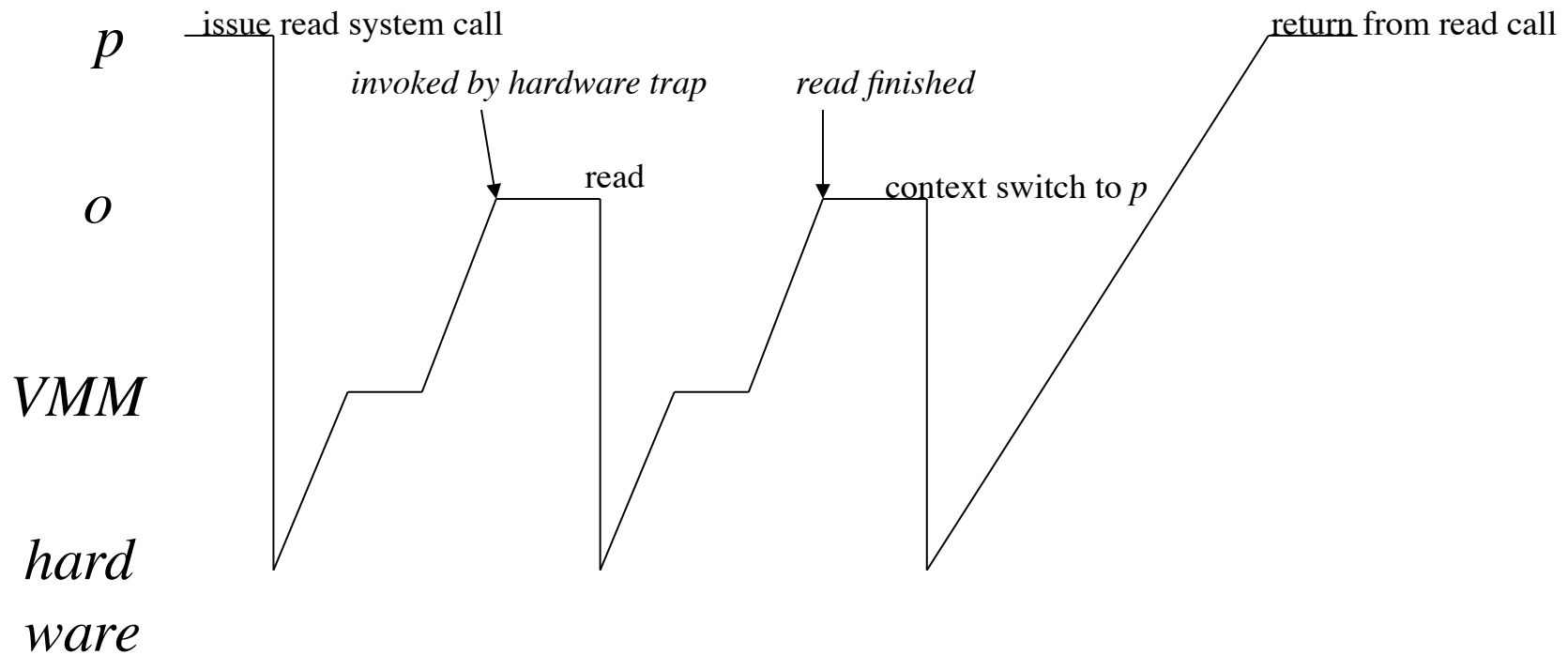
Privileged Instructions

1. VMM running operating system o , which is running process p
 - p tries to read—privileged operation traps to hardware
2. VMM invoked, determines trap occurred in o
 - VMM updates state of o to make it look like hardware invoked o directly, so o tries to read, causing trap
3. VMM does read
 - Updates o to make it seem like o did read
 - Transfers control to o

Privileged Instructions

4. o tries to switch context to p , causing trap
5. VMM updates virtual machine of o to make it appear o did context switch successfully
 - Transfers control to o , which (as o apparently did a context switch to p) has the effect of returning control to p

Privileged Instructions



Privilege and VMs

- *Sensitive instruction* discloses or alters state of processor privilege
- *Sensitive data structure* contains information about state of processor privilege

When Is VM Possible?

- Can virtualize an architecture when:
 1. All sensitive instructions cause traps when executed by processes at lower levels of privilege
 2. All references to sensitive data structures cause traps when executed by processes at lower levels of privilege

Example: VAX System

- 4 levels of privilege (user, supervisor, executive, kernel)
 - CHMK changes privilege to kernel level; sensitive instruction
 - Causes trap *except* when executed in kernel mode; meets rule 1
 - Page tables have copy of PSL, containing privilege level; sensitive data structure
 - If user level processes prevented from altering page tables, trying to do so will cause a trap; this meets rule 2

Multiple Levels of Privilege

- Hardware supports n levels of privilege
 - VM must also support n levels
 - VM monitor runs at highest level, so $n-1$ levels of privilege left!
- Solution: virtualize levels of privilege
 - Called *ring compression*

Example: VAX VMM System

- VMM at kernel level
- VMM maps virtual kernel and executive level to (real) executive mode
 - Called *VM kernel level*, *VM executive level*
 - Virtual machine bit added to PSL
 - If set, current process running on VM
 - Special register, VMPSL, records PSL of currently running VM
 - All sensitive instructions that *could* reveal level of privilege get this information from VMPSL or trap to the VMM, which then emulates the instruction

Alternate Approach

- Divide users into different classes
- Control access to system by limiting access of each class

Example: IBM VM/370

- Each control program command associated with user privilege classes
 - “G” (general user) class can start a VM
 - “A” (primary system operator) class can control accounting, VM availability, other system resources
 - “Any” class can gain or surrender access to VM

Physical Resources and VMs

- Distributes resources among VMs as appropriate
 - Each VM appears to have reduced amount of resources from real system
 - Example: VMM to create 10 VMs means real disk split into 10 minidisks
 - Minidisks may have different sizes
 - VMM does mapping between minidisk addresses, real disk addresses

Example: Disk I/O

- VM's OS tries to write to disk
 - I/O instruction privileged, traps to VMM
- VMM checks request, services it
 - Translates addresses involved
 - Verifies I/O references disk space allocated to that VM
 - Services request
- VMM returns control to VM when appropriate
 - If I/O synchronous, when service complete
 - If I/O asynchronous, when service begun

Paging and VMs

- Like ordinary disk I/O, but 2 problems
 - Some pages may be available only at highest level of privilege
 - VM must remap level of privilege of these pages
 - Performance issues
 - VMM paging its own pages is transparent to VMs
 - VM paging is handled by VMM; if VM's OS does lots of paging, this may introduce significant delays

Example: VAX/VMS

- On VAX/VMS, only kernel level processes can read some pages
 - What happens if process at VM kernel level needs to read such a page?
 - Fails, as VM kernel level is at real executive level
 - VMM reduces level of page to executive, then it works
 - Note: security risk!
 - In practice, OK, as VMS allows executive level processes to change to kernel level

Example: IBM VM/370

- Supports several different operating systems
 - OS/MFT, OS/MVT designed to access disk storage
 - If jobs being run under those systems depend on timings, delay caused by VM may affect success of job
 - If system supports virtual paging (like MVS), either MVS or VMM may cause paging
 - The VMM paging may introduce overhead (delays) that cause programs to fail that would not were the programs run directly on the hardware

Anonymity on the Internet

(slides from Prof. Vitaly Shmatikov, UTAustin)

Privacy on Public Networks

- Internet is designed as a public network
 - Machines on your LAN may see your traffic, network routers see all traffic that passes through them
- Routing information is public
 - IP packet headers identify source and destination
 - Even a passive observer can easily figure out **who is talking to whom**
- Encryption does not hide identities
 - Encryption hides payload, but not routing information
 - Even IP-level encryption (tunnel-mode IPsec/ESP) reveals IP addresses of IPsec gateways

Applications of Anonymity (I)

- Privacy
 - Hide online transactions, Web browsing, etc. from intrusive governments, marketers and archivists
- Untraceable electronic mail
 - Corporate whistle-blowers
 - Political dissidents
 - Socially sensitive communications (online AA meeting)
 - Confidential business negotiations
- Law enforcement and intelligence
 - Sting operations and honeypots
 - Secret communications on a public network

Applications of Anonymity (II)

- Digital cash
 - Electronic currency with properties of paper money (online purchases unlinkable to buyer's identity)
- Anonymous electronic voting
- Censorship-resistant publishing
- Crypto-anarchy
 - “Some people say `anarchy won't work'. That's not an argument against anarchy; that's an argument against work.” – Bob Black

What is Anonymity?

- Anonymity is the state of being not identifiable within a **set of subjects**
 - You cannot be anonymous by yourself!
 - Big difference between anonymity and confidentiality
 - Hide your activities among others' similar activities
- Unlinkability of action and identity
 - For example, sender and his email are no more related after observing communication than they were before
- Unobservability (hard to achieve)
 - Any item of interest (message, event, action) is indistinguishable from any other item of interest

Attacks on Anonymity

- Passive traffic analysis
 - Infer from network traffic who is talking to whom
 - To hide your traffic, must carry other people's traffic!
- Active traffic analysis
 - Inject packets or put a timing signature on packet flow
- Compromise of network nodes
 - Attacker may compromise some routers
 - It is not obvious which nodes have been compromised
 - Attacker may be passively logging traffic
 - Better not to trust any individual router
 - Assume that some fraction of routers is good, don't know which

Chaum's Mix

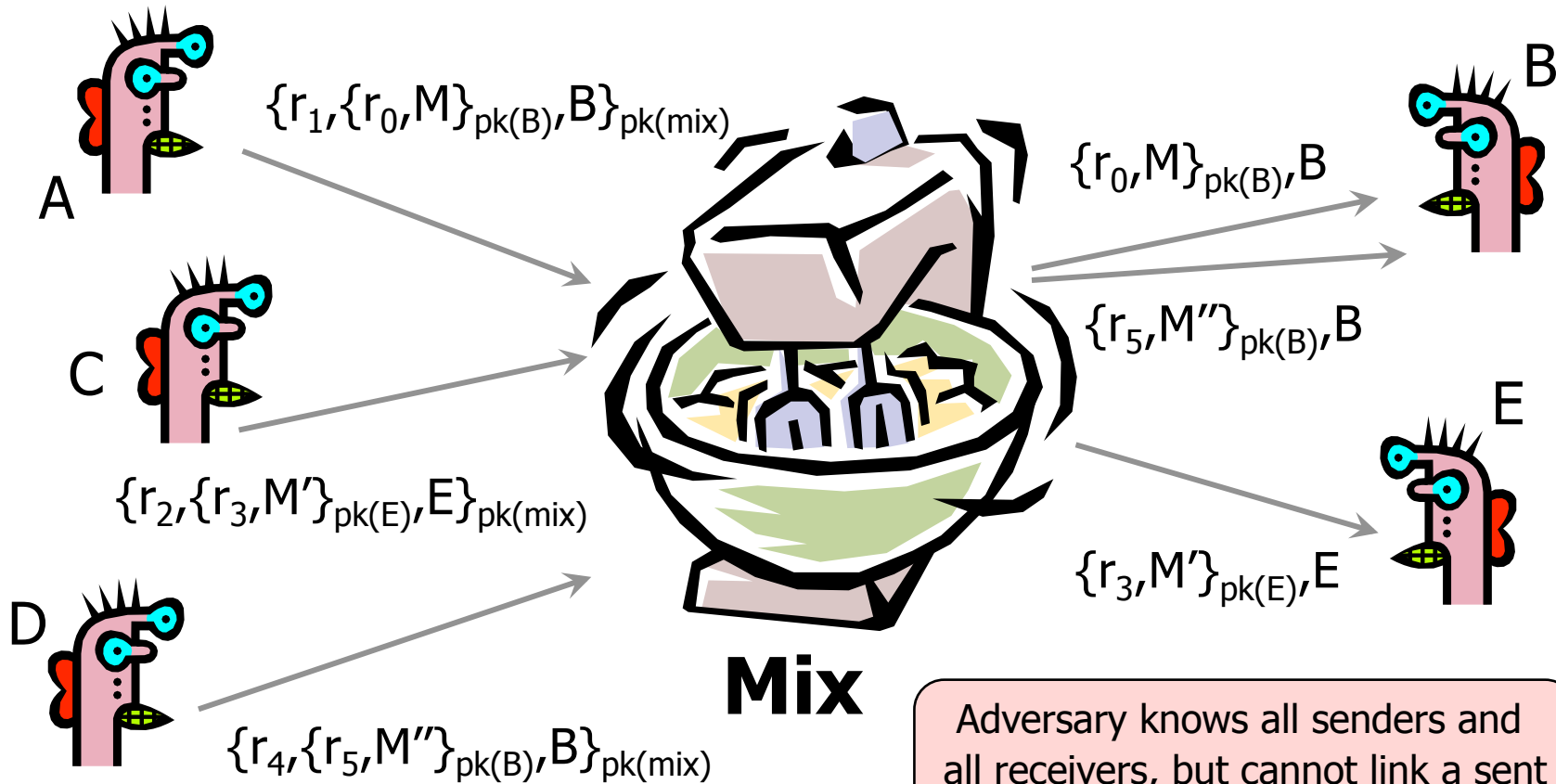
- Early proposal for anonymous email
 - David Chaum. “Untraceable electronic mail, return addresses, and digital pseudonyms”.

Communications of the ACM

Before spam, people thought anonymous email was a good idea 😊

- Public key crypto + trusted re-mailer (Mix)
 - Untrusted communication medium
 - Public keys used as persistent pseudonyms
- Modern anonymity systems use Mix as the basic building block

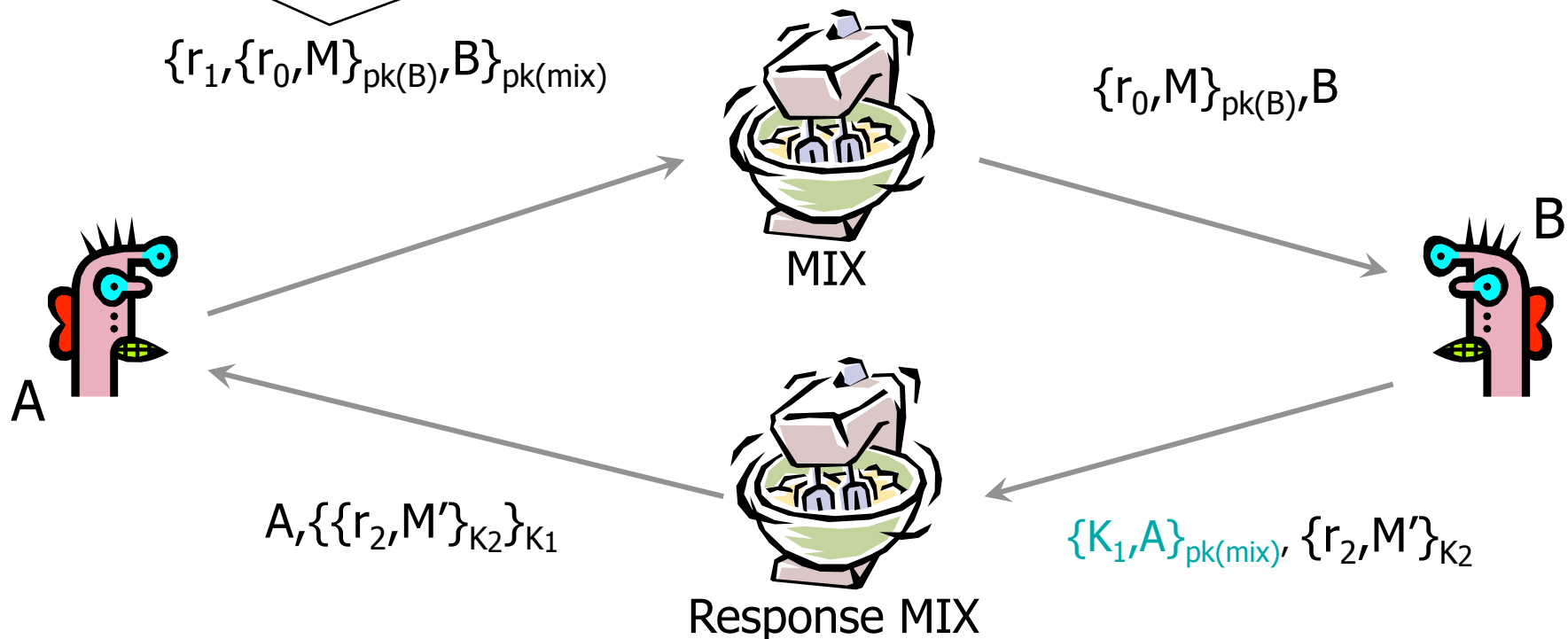
Basic Mix Design



Adversary knows all senders and all receivers, but cannot link a sent message with a received message

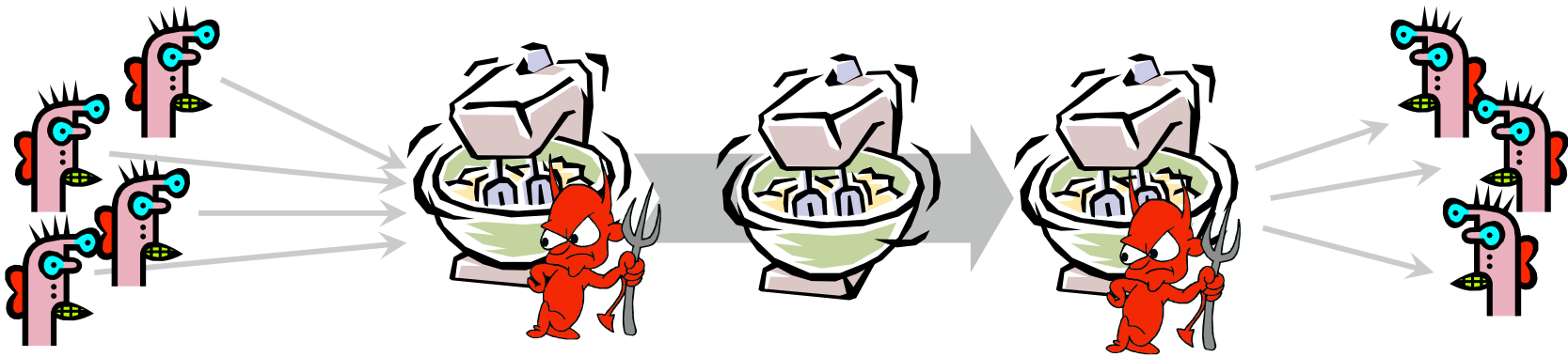
Anonymous Return Addresses

M includes $\{K_1, A\}_{pk(mix)}$, K_2 where K_2 is a fresh public key



Secrecy without authentication
(good for an online confession service 😊)

Mix Cascade

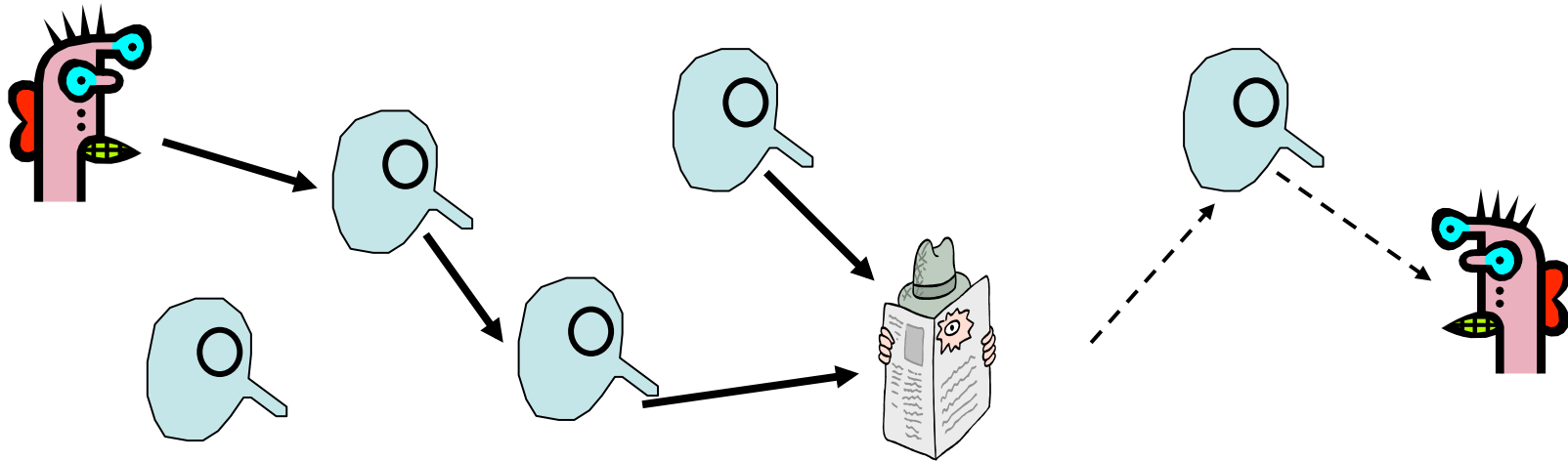


- Messages are sent through a **sequence of mixes**
 - Can also form an arbitrary network of mixes (“mixnet”)
- Some of the mixes may be controlled by attacker, but even a single good mix guarantees anonymity
- Pad and buffer traffic to foil correlation attacks

Disadvantages of Basic Mixnets

- Public-key encryption and decryption at each mix are computationally expensive
- Basic mixnets have high latency
 - Ok for email, not Ok for anonymous Web browsing
- Challenge: low-latency anonymity network
 - Use public-key cryptography to establish a “circuit” with pairwise symmetric keys between hops on the circuit
 - Then use symmetric decryption and re-encryption to move data messages along the established circuits
 - Each node behaves like a mix; anonymity is preserved even if some nodes are compromised

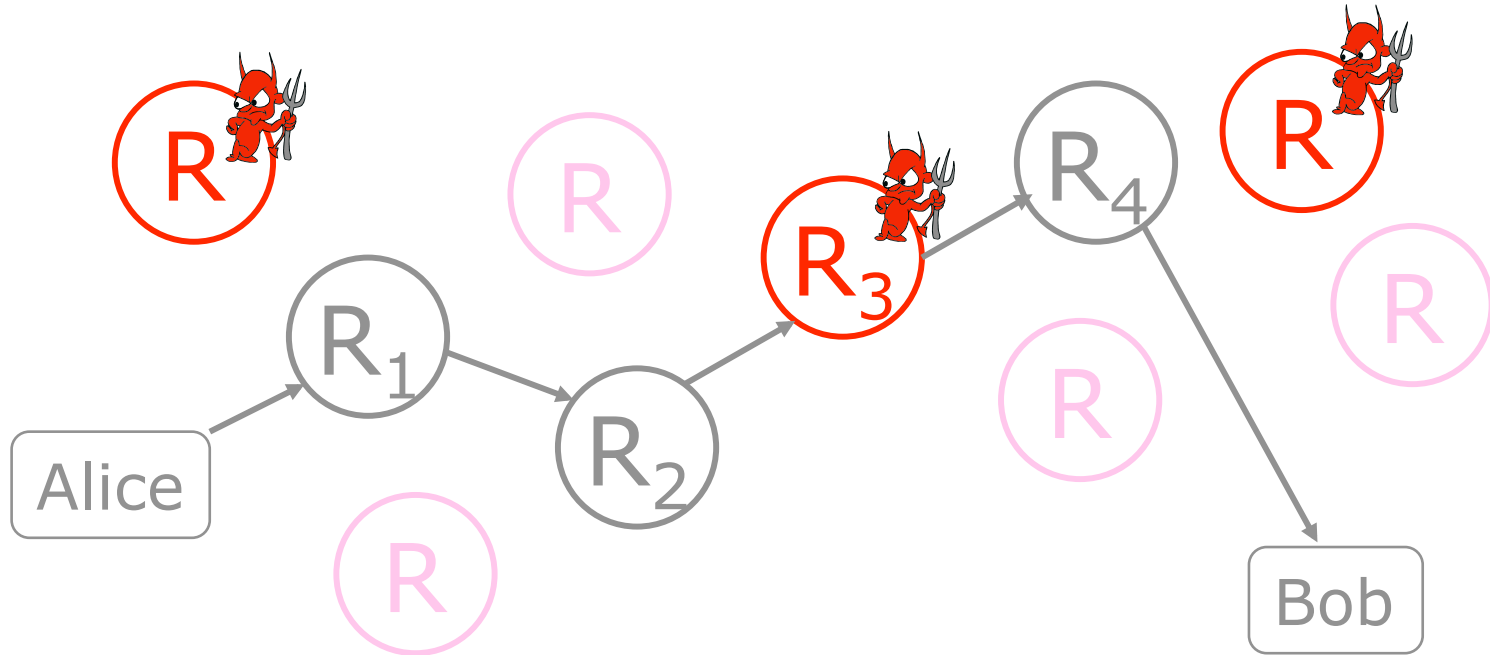
Another Idea: Randomized Routing



- Hide message source by routing it randomly
 - Popular technique: Crowds, Freenet, Onion routing
- Routers don't know for sure if the apparent source of a message is the true sender or another router

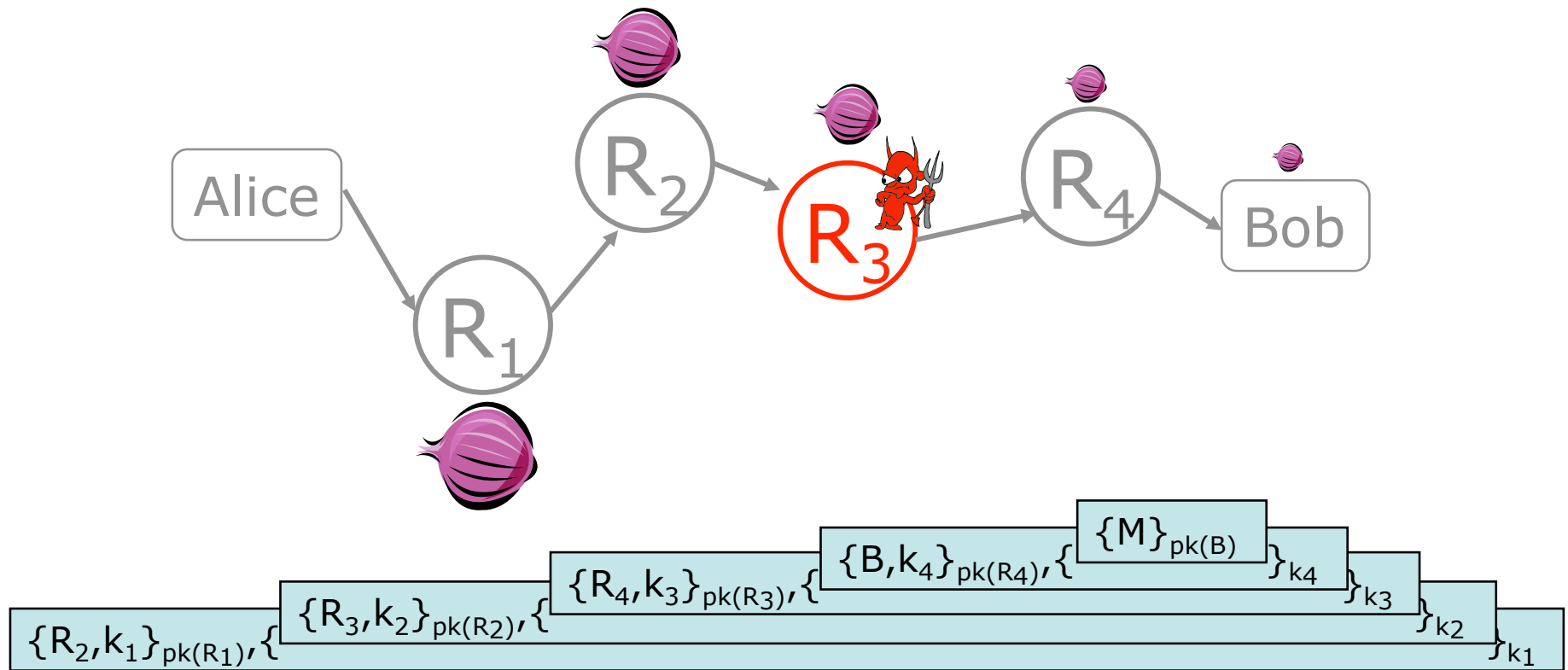
Onion Routing

[Reed, Syverson, Goldschlag '97]



- Sender chooses a random sequence of routers
 - Some routers are honest, some controlled by attacker
 - Sender controls the length of the path

Route Establishment



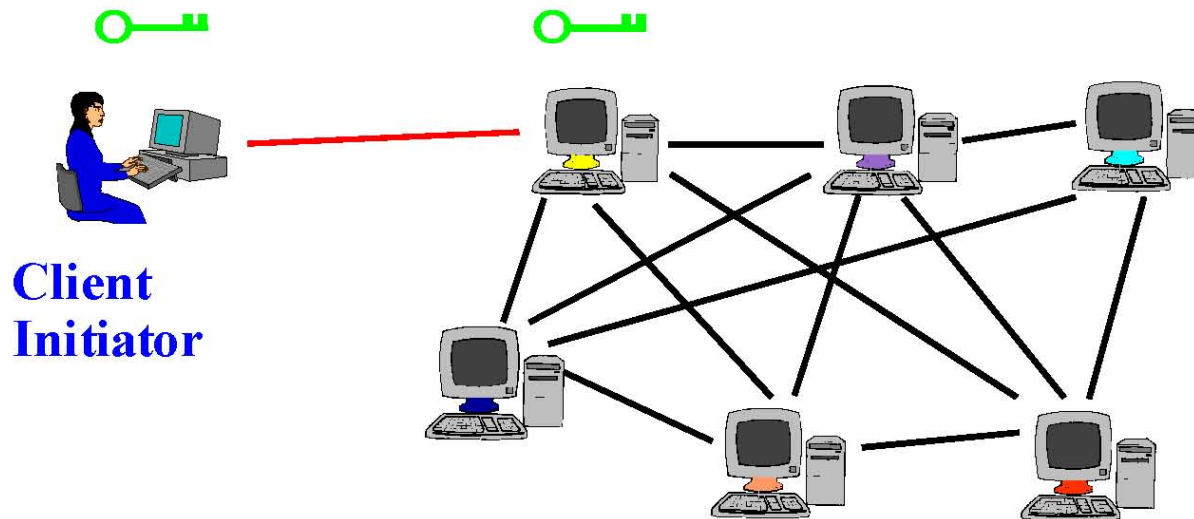
- Routing info for each link encrypted with router's public key
- Each router learns only the identity of the next router

Tor




- Second-generation onion routing network
 - <http://tor.eff.org>
 - Developed by Roger Dingledine, Nick Mathewson and Paul Syverson
 - Specifically designed for **low-latency** anonymous Internet communications
- Running since October 2003
- 100 nodes on four continents, thousands of users
- “Easy-to-use” client proxy
 - Freely available, can use it for anonymous browsing

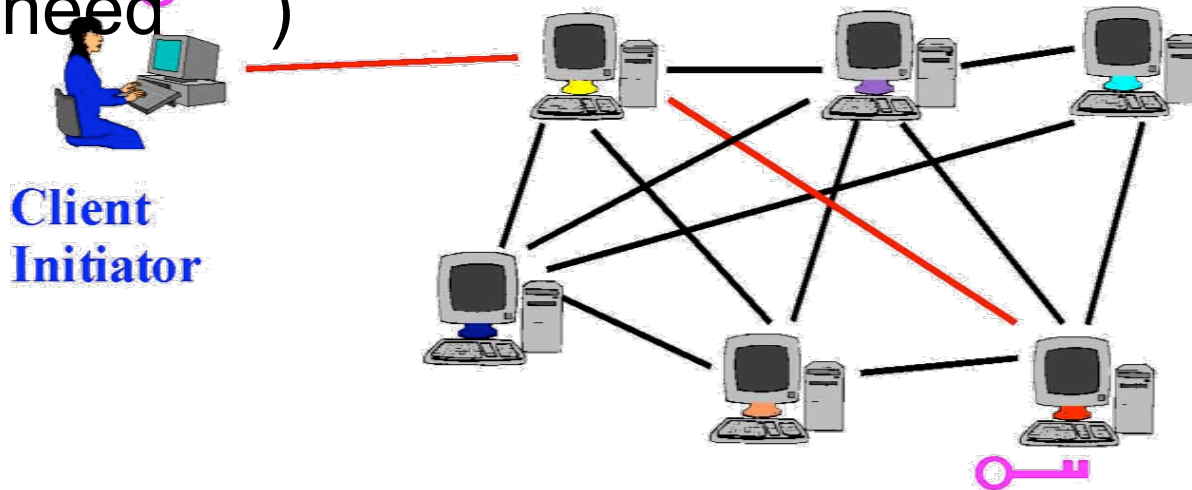
Tor Circuit Setup (1)

- Client proxy establish a symmetric session key and circuit with Onion Router #1



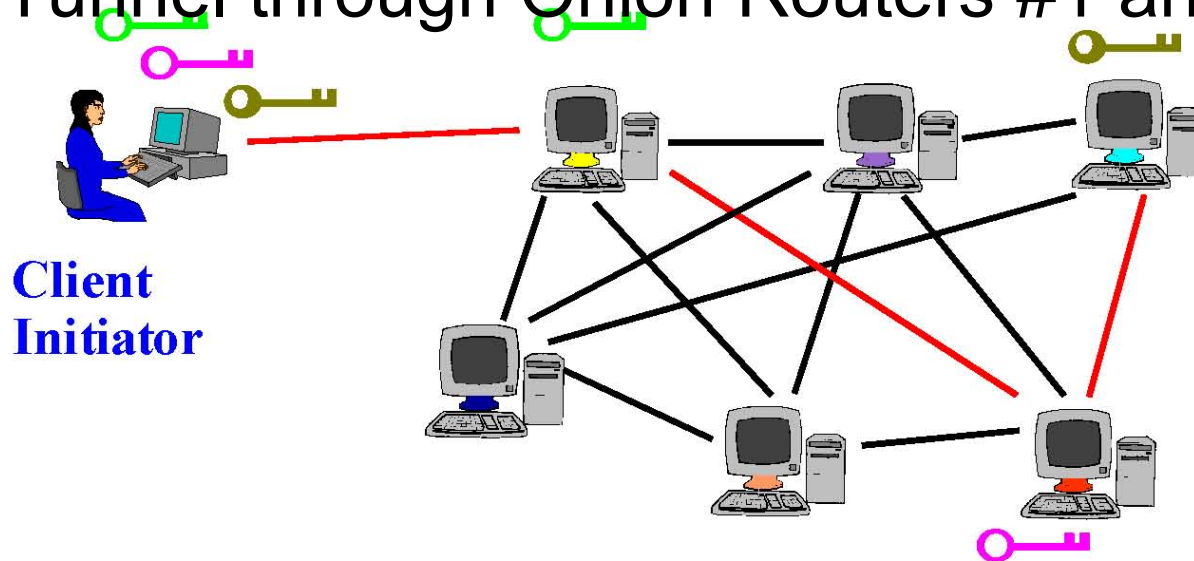
Tor Circuit Setup (2)

- Client proxy extends the circuit by establishing a symmetric session key with Onion Router #2 
- Tunnel through Onion Router #1 (don't need  )



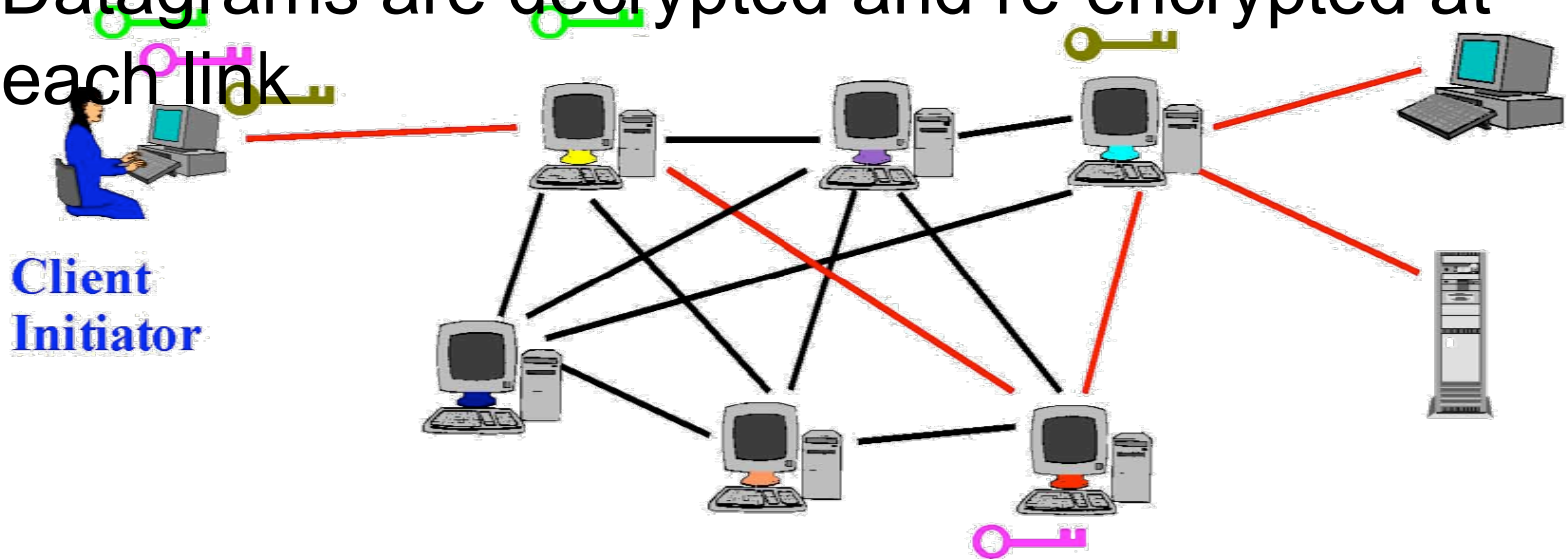
Tor Circuit Setup (3)

- Client proxy extends the circuit by establishing a symmetric session key with Onion Router #3
 - Tunnel through Onion Routers #1 and #2



Using a Tor Circuit

- Client applications connect and communicate over the established Tor circuit
 - Datagrams are decrypted and re-encrypted at each link



Tor Management Issues

- Many applications can share one circuit
 - Multiple TCP streams over one anonymous connection
- Tor router doesn't need root privileges
 - Encourages people to set up their own routers
 - More participants = better anonymity for everyone
- Directory servers
 - Maintain lists of active onion routers, their locations, current public keys, etc.
 - Control how new routers join the network