Reflections on security by design

Aslan Askarov, 2025-12-04

Gallery of Horrors: Denmark relies fundamentally on flawed software. (public & private sector, critical infrastructure, defense)

Recent High-Profile Cyberattacks in Denmark

Reuters World Business Market

emails
Russian state hackers

Russia hacked Danish defense for two years, minister tells newspaper

By Reuters

April 24, 2017 5:22 PM GMT+2 · Updated April 24, 2017

WIRED

ransomware
Russian state hackers
Mærsk lost $300M, others $10B

ANDY GREENBERG EXCERPT SECURITY AUG 22, 2018 5:00 AM

The Untold Story of NotPetya, the Most Devastating Cyberattack in History

Crippled ports. Paralyzed corporations. Frozen government agencies. How a single piece of code crashed the world.

Hackerangreb koster danske virksomheder over 1,4 millioner kr. om ugen

13.3.2025 09:02:00 CET | TDC Erhverv | Pressemeddelelse

Ingeniøren

backdoor for 7 months
Russian state hackers

Version2 Nyheder Blo… Debat Nyhedsbreve Tip os

Central bank of Denmark hacked as part of 'the world's most sophisticated hacker attack'

It-sikkerhed 29. juni 2021 kl. 12:24 3 kommentarer

FINTECH FUTURES

leaked 100k credit cards

PAYTECH NEWS

Nets warns of massive credit card hack in Denmark

Danish payment processor Nets has warned local b… and advised them to block up to 100,000 credit card…

Tanya Andreasyan
October 26, 2016

Zyxel firewalls hijacked
22 energy companies breached (industrial control systems)

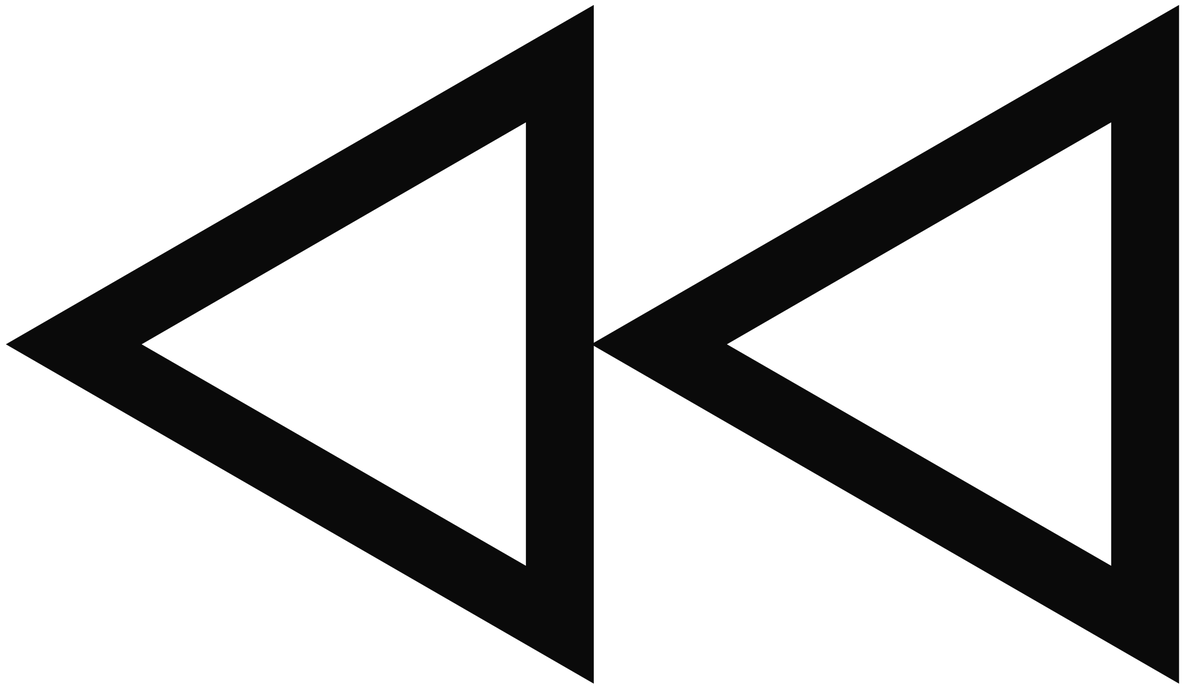
The attack against Danish, critical infrastructure

SEKTOR CERT

Published: November 2023

- What will it mean for the next-generation OS to be secure and how will that change software development? ■ Digital Tech Summit

Slide ack: Willard Rafnsson



2012

The Research Value of Publishing Attacks

David Basin and Srdjan Capkun, *CACM, Nov 2012*

viewpoints

DOI:10.1145/2366316.2366324

David Basin and Srdjan Capkun

Privacy and Security

The Research Value of Publishing Attacks

Security research can be improved by more effectively sharing what is learned from attacks on information systems.

INFORMATION SECURITY IS booming. Companies are making money selling fear and countermeasures. The research community is also extremely active, churning out papers featuring attacks on systems and their components. This includes attacks on traditional IT systems as well as IT-enhanced systems, such as cars, implantable medical devices, voting systems, and smart meters, which are not primarily IT systems but have increasing amounts of IT inside. Moreover, any new paper on analysis methods for critical systems is now considered incomplete without a collection of security-relevant scalps on its belt. Pretty much every system imaginable, critical or not, is now a target of attacks.

There are good reasons for this trend. Fear sells! Headlines are good for conference attendance, readership, and tenure cases. Moreover, negative messages about successful attacks are simple and understandable by the general public, much more so than other research results. And security and insecurity are, after all, two sides of the same coin.

Seek and Ye Shall Find

Systems have bugs and large, complex systems have many bugs. In their recent analysis of open source projects, Coverty[2] used a static analysis tool to find 16,884 defects in approximately 37.5 million lines of source code from well-managed open source projects, which is approximately 0.45 bugs per 1,000 lines of code. These were medium to high-risk defects, including typical security-critical vulnerabilities such as memory corruption problems and API usage errors. For large-scale projects, developers cope with the seemingly infinite number of bugs in their products by employing triage processes to classify which bugs they work on and which they ignore. There are simply too many to address them all.

This should not come as a surprise. Complexity is at odds with security. Moreover, economic factors are often at play, where timeliness and functionality are more important than security. But there are other reasons why insecurity is omnipresent.

To begin with, systems undergo constant evolution. There has been a recent surge in attacks where once-closed systems, like medical devices and cars, open up and are enhanced with new communication interfaces (for example, see Francillon et al.,[3] Halperin et al.,[4] and Rouf et al.[6]). The problem here is that the extended capabilities were usually not anticipated in the original design, often resulting in vulnerabilities that are easy to exploit. Not surprisingly, adding wireless communication without measures to ensure the confidentiality and authenticity of transmitted data results in a system vulnerable to eavesdropping and spoofing. This problem is particularly acute for products manufactured by traditional industries that did not previously require expertise in information security.

Systems not only interface with the outside world, they also interface with each other. For their composition to be secure, the assumptions of one subsystem must match the guarantees of the other. However, economics and market availability often dictate the choices made, especially for hardware components where manufactur-

"As our physical and digital worlds become more tightly coupled, the incidence of attacks will increase as well as their consequences. Many of these attacks will be newsworthy, but most will not be research-worthy."

"What makes security special is the role of the adversary. A system's security can only be evaluated with respect to a model of the adversary, that is, a description of his capabilities. Thus, in our view, the most important reason for studying attacks is that they can help refine this model for the domain at hand"

Threat model

(aka attacker model)

Who is the attacker and what can they do?

for example, where are they vertically in the stack

Structured Programming Language
Compiler
User-level Instruction Set Architecture
Operating system
Privileged Instruction Set Architecture
Processor
Hardware description language

Sometimes, it is meaningful to compare attackers *strength*

Observing only I/O vs intermediate memory writes vs timing

Attacker knowing the source of the program vs providing the program

The relevance of threat models evolves over time

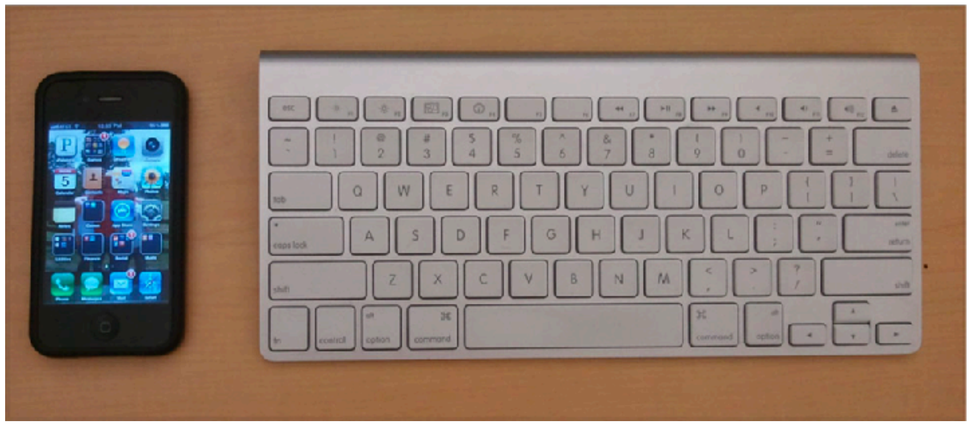
Adversarial thinking: exploiting gaps between abstraction layers

Structured Programming Language
Compiler
User-level Instruction Set Architecture
Operating system
Privileged Instruction Set Architecture
Processor
Hardware description language

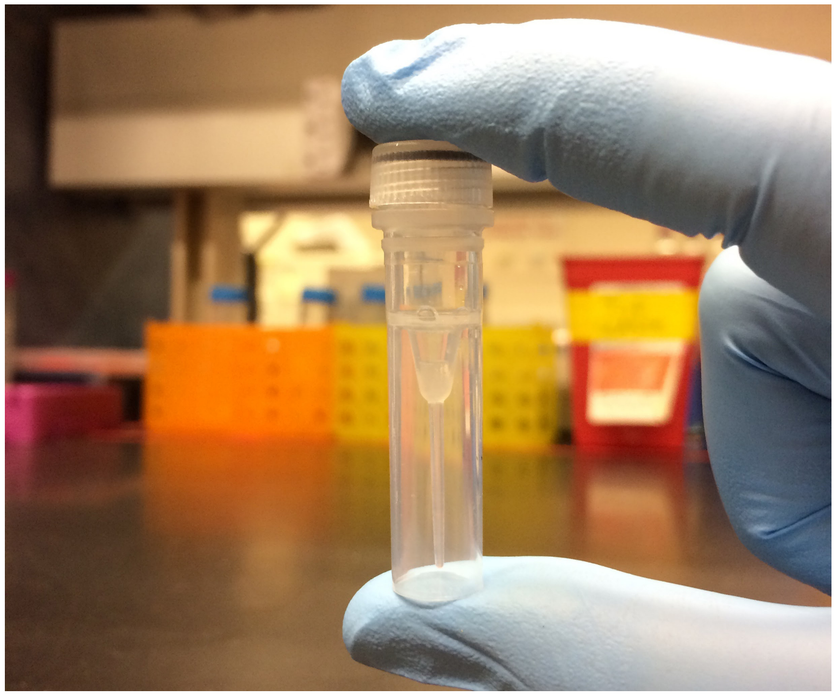
[Security across abstraction layers: old and new examples, Frank Piessens, 2020]

Exploiting gaps between abstraction layers

Also, in the physical world



Using smartphone accelerometer to infer keystrokes from desk vibrations [Marquardt et al., 2011]



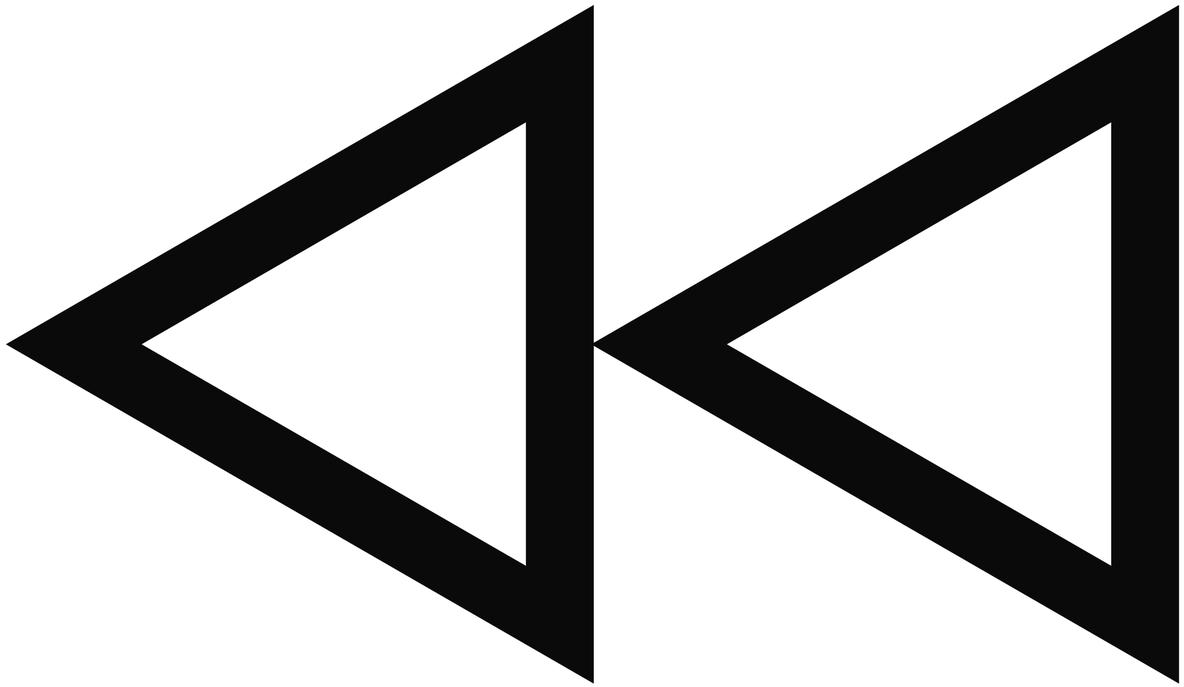
Synthesized biomaterial that triggers buffer overflows in a DNA sequencing machine [Ney et al, 2017]

Threat model subtleties

Security against a strong adversary typically implies security against a weaker one

Not always the case [Askarov, Chong, CSF'12]

```
// A is the manager
// B is an employee
send (A, B.salary)  // OK, because A is the manager
A.retire ()
send (A, B.salary)  // Is this OK?
                    // - Depends on how much A remembers
```



2004

Computer Security in the Real World

Most computers are insecure because security is expensive. Security depends on authentication, authorization, and auditing: the gold standard. The key to uniform security in the Internet is the idea of one security principal speaking for another.

Butler W. Lampson
Microsoft

Computer system security is more than 30 years old. It has had many intellectual successes, among them the subject/object access matrix model,[1] access control lists,[2] multilevel security using information flow[3,4] and the star property,[5] public-key cryptography,[6] and cryptographic protocols.[7] Despite these successes, in an absolute sense the security of the hundreds of millions of deployed computer systems remains terrible. A determined and competent attacker could steal or destroy most of the information on most of these systems. Even worse, the attacker could do this to millions of systems at once.

The Internet has made computer security much more difficult. Twenty years ago, a computer system had a few dozen users at most, all members of the same organization. Today, half a billion people all over the world connect to the Internet. Anyone can attack your system. Your system, if compromised, can infect others automatically. You face possibly hostile code that comes from many different sources, often without your knowledge. Your laptop faces a hostile physical environment. If you own content and want to sell it, you face hostile hosts. You can't just isolate yourself, because you may want to share information with anyone or run code from anywhere.

These vulnerabilities invite vandalism: worms and viruses. They also make it much easier to attack a specific target, either to steal information or to corrupt data. On the other hand, the actual harm these attacks cause is limited, though increasing. Unfortunately, there is no accurate data about the cost of computer security failures: Most are never made public for fear of embarrassment, but when a public incident does occur, security experts and vendors have every incentive to exaggerate its costs.

Money talks, though. Many companies have learned that although people may complain about inadequate security, they won't spend much money, sacrifice many features, or put up with much inconvenience to improve it. This strongly suggests that bad security is not really costing them much. Firewalls and antivirus programs are the only really successful security products, and they are carefully designed to require no end user setup and to interfere very little with daily life.

The experience of the past few years confirms this analysis. Virus attacks have increased, and people are now more likely to buy a firewall and antivirus software and to install patches that fix security flaws. Vendors are making their systems more secure, at some cost in backward compatibility and user convenience. But the changes have not been dramatic.

Many people have suggested that the PC monoculture makes security problems worse and that more diversity would improve security. It's true that vandals can get more impressive results when most systems have the same flaws. On the other hand, if an organization installs several different systems that all have access to the same critical data, as they probably will, then a targeted attack only needs to find a flaw in one of them to succeed.

WHAT IS SECURITY?

What do we want from secure computer systems? Here is a reasonable goal: *Computers are as secure as real-world systems, and people believe it.*

Most real-world systems are not very secure by any absolute standard. It's easy to break into someone's house; in fact, in many places people don't

0018-9162/04/$20.00 © 2004 IEEE Published by the IEEE Computer Society June 2004

Security is about risks

- We don't have "real" security and the main reason is that people don't buy it. The danger is small and security is pain. People prefer to buy features
- Systems are complicated. There are bugs that an attackers can exploit, such as buffer overruns or other flaws that break the basic programming abstractions. This gets all the attention, but it is not the heart of the problem.
- Will things get better? Certainly, when security flaws cause serious damage, buyers change their priorities and systems become more secure, but unless there's a catastrophe, these changes are slow.
- Short of that, the best we can do is to drastically simplify the parts of systems that have to do with security.

Elements of security

[Lampson'06]

Policy

Specifying security
What is it supposed to do?

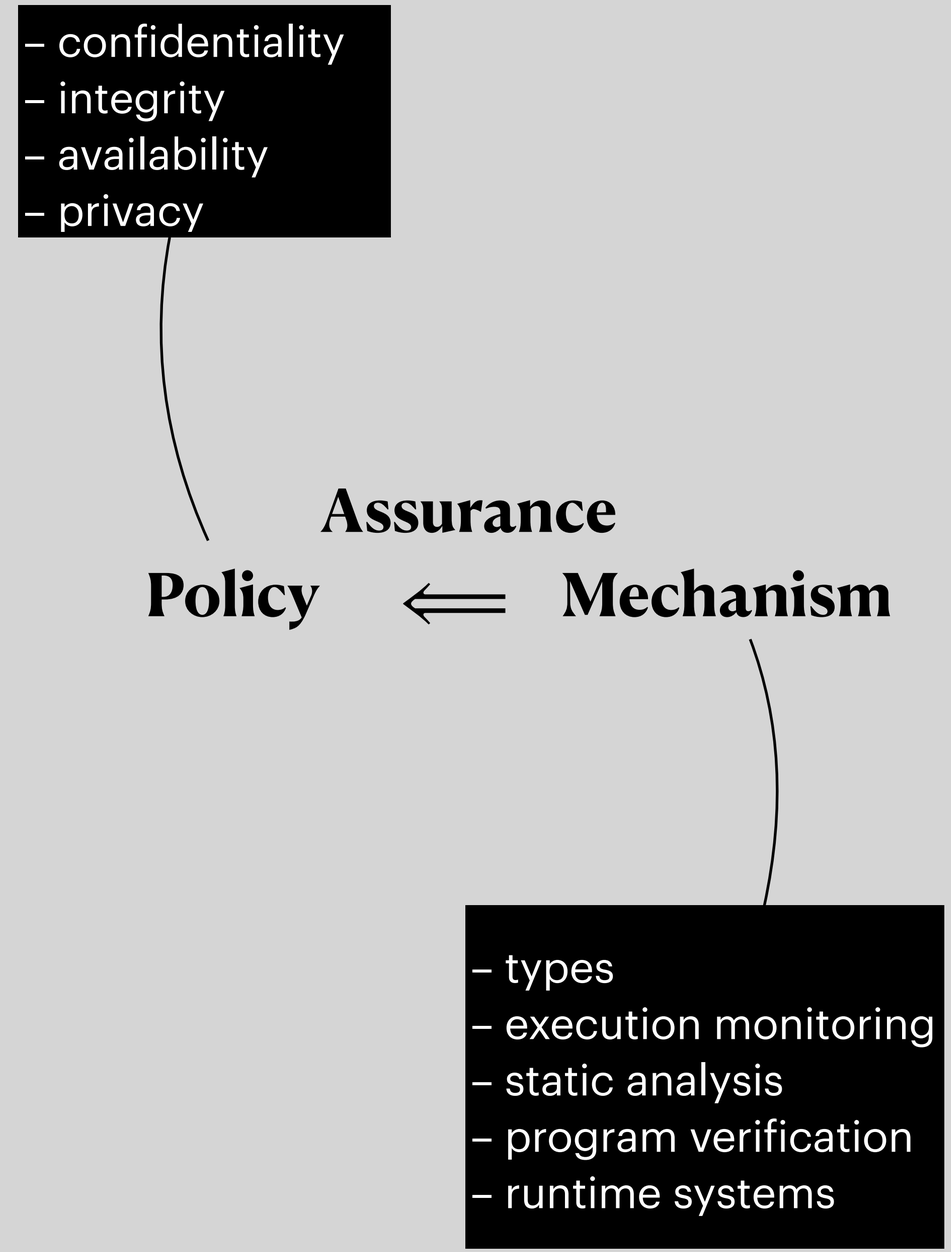
Mechanism

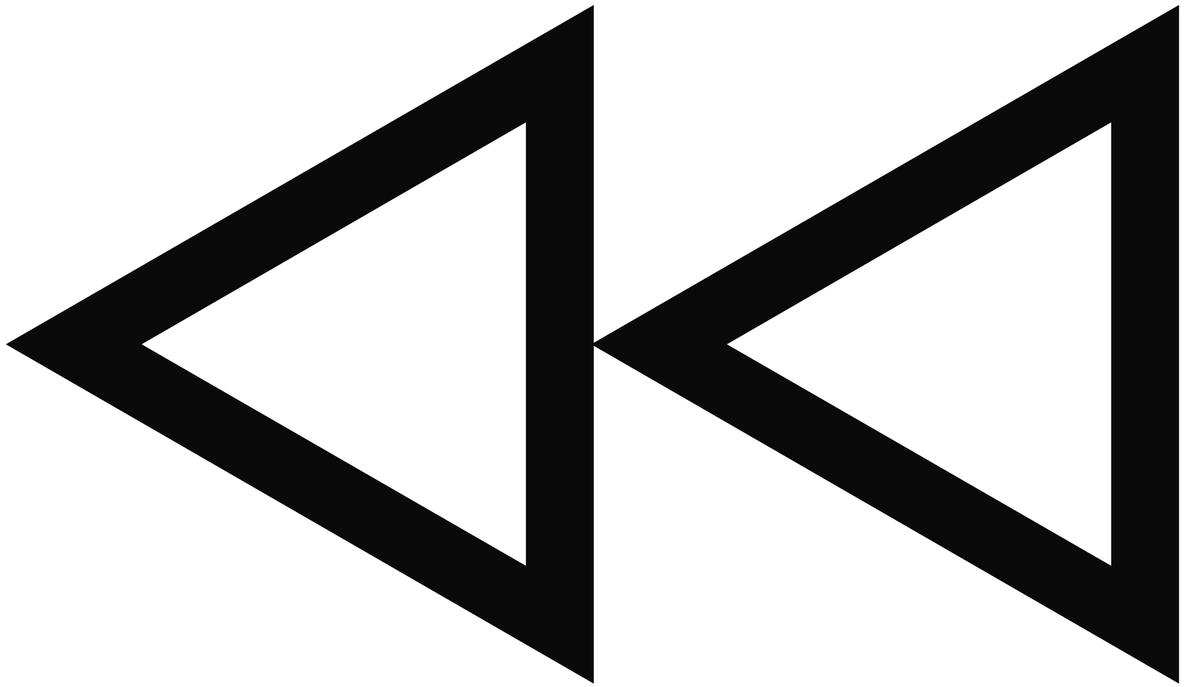
Implementing security
How does it do it?

Assurance

Correctness of security
Does it really work?

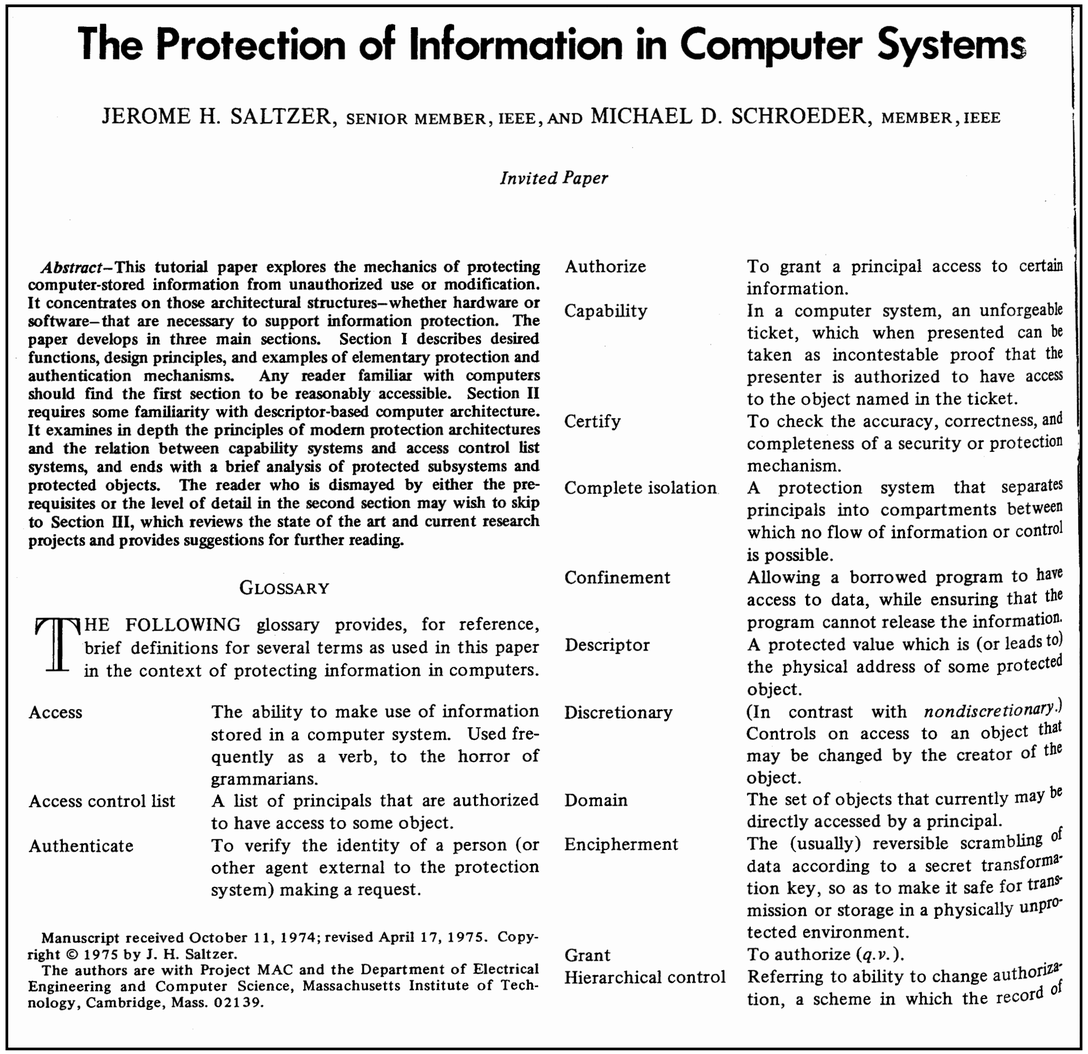
[https://www.microsoft.com/en-us/research/uploads/prod/2007/01/Lampson-Security-for-Marktoberdorf-slides-8-5.pdf]





1975

Saltzer & Schroeder's design principles, 1975



The Protection of Information in Computer Systems

JEROME H. SALTZER, SENIOR MEMBER, IEEE, AND MICHAEL D. SCHROEDER, MEMBER, IEEE

Invited Paper

Abstract—This tutorial paper explores the mechanics of protecting computer-stored information from unauthorized use or modification. It concentrates on those architectural structures—whether hardware or software—that are necessary to support information protection. The paper develops in three main sections. Section I describes desired functions, design principles, and examples of elementary protection and authentication mechanisms. Any reader familiar with computers should find the first section to be reasonably accessible. Section II requires some familiarity with descriptor-based computer architecture. It examines in depth the principles of modern protection architectures and the relation between capability systems and access control list systems, and ends with a brief analysis of protected subsystems and protected objects. The reader who is dismayed by either the prerequisites or the level of detail in the second section may wish to skip to Section III, which reviews the state of the art and current research projects and provides suggestions for further reading.

GLOSSARY

THE FOLLOWING glossary provides, for reference, brief definitions for several terms as used in this paper in the context of protecting information in computers.

Access	The ability to make use of information stored in a computer system. Used frequently as a verb, to the horror of grammarians.
Access control list	A list of principals that are authorized to have access to some object.
Authenticate	To verify the identity of a person (or other agent external to the protection system) making a request.
Authorize	To grant a principal access to certain information.
Capability	In a computer system, an unforgeable ticket, which when presented can be taken as incontestable proof that the presenter is authorized to have access to the object named in the ticket.
Certify	To check the accuracy, correctness, and completeness of a security or protection mechanism.
Complete isolation	A protection system that separates principals into compartments between which no flow of information or control is possible.
Confinement	Allowing a borrowed program to have access to data, while ensuring that the program cannot release the information.
Descriptor	A protected value which is (or leads to) the physical address of some protected object.
Discretionary	(In contrast with *nondiscretionary*.) Controls on access to an object that may be changed by the creator of the object.
Domain	The set of objects that currently may be directly accessed by a principal.
Encipherment	The (usually) reversible scrambling of data according to a secret transformation key, so as to make it safe for transmission or storage in a physically unprotected environment.
Grant	To authorize (*q.v.*).
Hierarchical control	Referring to ability to change authorization, a scheme in which the record of

Manuscript received October 11, 1974; revised April 17, 1975.
The authors are with Project MAC and the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, Mass. 02139.

Economy of mechanism

Fail-safe defaults

Complete mediation

Open design

Separation of privilege

Least privilege

Least common mechanism

Psychological acceptability

Saltzer & Schroeder's design principles, 1975

Economy of mechanism	*simpler mechanisms are easier to verify*
Fail-safe defaults	*do not give access by default*
Complete mediation	*do not trade performance for security*
Open design	*no security through obscurity*
Separation of privilege	*minimize damage of a single accident, e.g, multi-factor auth*
Least privilege	*need to know, e.g., capability machines*
Least common mechanism	*sharing leads to security problems*
Psychological acceptability	*there are humans in the loop, e.g,. passwords not ideal*

Negative requriments

A characteristic aspect of security is that it is about what is *not* supposed to happen. It is a *negative requirement* that is difficult to test. Unlike functionality, where one can test for expected behavior, one cannot test what they don’t know.

It is also why formal methods are necessary. The use of formal methods in security is driven by need – we do it because otherwise we have no assurance.

Least Common Mechanism

Minimize the amount of mechanism common to more than one user and depended on by all users. Every shared mechanism (especially one involving shared variables) represents a potential information path between users and must be designed with great care to be sure it does not unintentionally compromise security.

Further, any mechanism serving all users must be certified to the satisfaction of every user, a job presumably harder than satisfying only one or a few users.

Structured Programming Language
Compiler
User-level Instruction Set Architecture
Operating system
Privileged Instruction Set Architecture
Processor
Hardware description language

Insecurities in common mechanisms

Minimalist cache side-channel

https://github.com/aslanix/cache-attack-example/blob/master/example.c

```
void attack()
{   // read the secret
    int secret = readNumber ();
    // we allocate a couple of blocks
    // that are likely to be adjacently allocated
    int* secret_buffer = (int*) malloc (512) ;
    int* public_buffer = (int*) malloc (512) ;

    if (secret > 42) {
        // accessing secret buffer brings the public_buffer into
        // the cache
        secret_buffer [0] = secret;
    }

    uint64_t t1 = get_cycles () ;
    int garbage = public_buffer [0]; // read something from the public block
    uint64_t t2 = get_cycles () ;

    printf ("Time delta = %llu; read garbage = %d\n", t2 - t1, garbage) ;
    return;
}
```

```
» make
clang example.c -o example
» make run
./example
Enter an integer:100
Time delta = 52; read garbage = 0
» make
clang example.c -o example
» make run
./example
Enter the secret:0
Time delta = 7335; read garbage = 0
```

Are cache channels new?

- No, Paul Kocher demonstrated attacks on RSA in 1996
- The general concept of covert channels is described by Lampson's 1973 "A note on the confinement problem" paper
- Intel Secure Guard Extensions (2013) intended to provide secure enclaves had little protection against side channels?
  - Cost/Risk?
  - Subsequently shown to have many issues, often due to side channels

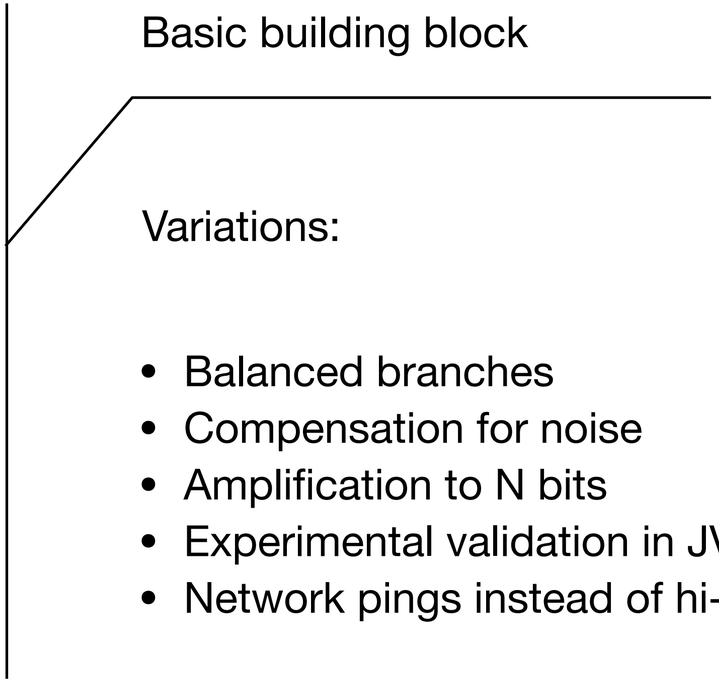
Leaking via GC-timing [Pedersen, Askarov, 2017]

Special case: only one bit

```
if (secretBit) {
    x = new [LARGENUMBER]
}
x = null

t1 = getTime()
y = new [LARGENUMBER]
t2 = getTime()

aliceServer.send (t2 - t1)
```

Leaking via GC-timing

```
if (secretBit) {
    x = new [LARGENUMBER]
}
x = null

t1 = getTime()
y = new [LARGENUMBER]
t2 = getTime()

aliceServer.send (t2 - t1)
```



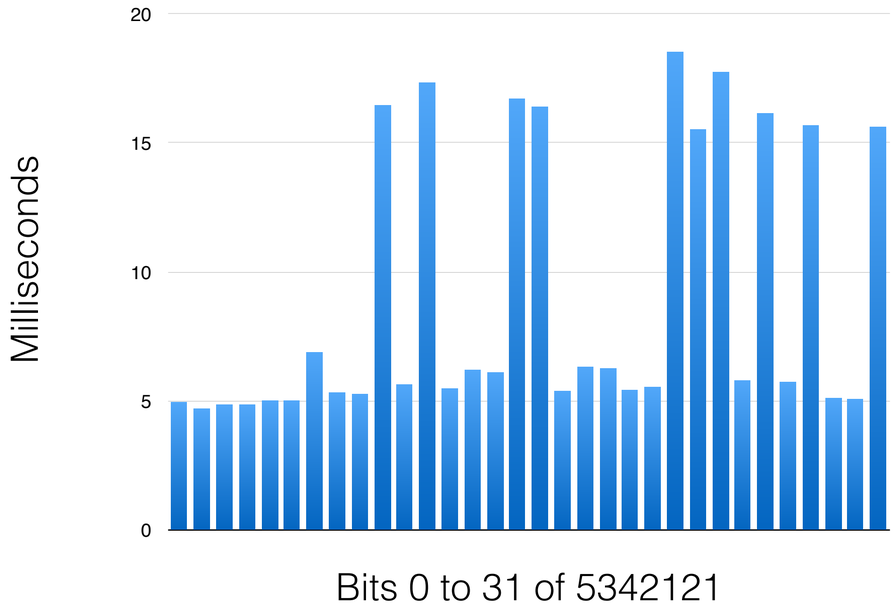
Basic building block

Variations:

- Balanced branches
- Compensation for noise
- Amplification to N bits
- Experimental validation in JVM and V8
- Network pings instead of hi-res timers

Amplification to N bits

```
long[] times = new long[N];

for(int bit = 31; bit >= 0; --bit) {
  for(int i = 0; i < N; ++i) {
    int[][] a = new int[K][size];
    int[][] b;
    int[][] c;
    int[][] d;
    if(((secret >> bit) & 1) > 0) {
      b = new int[K][size];
      d = a;
    }
    else {
      c = new int[K][size];
      b = a;
    }
    long before = System.nanoTime();
    int[] c = new int[size2];
    long after = System.nanoTime();
    if(after - before > threshold) {
      times[i] = after - before;
    }
    else {
      times[i] = 0;
    }
  }

  long sum = 0;
  long gcs = 0;
  for(int i = 0; i < N; ++i) {
    long t = times[i];
    t += times[i];
    if(t != 0) ++gcs;
  }
  if(gcs == 0) {
    ++bit; continue;
  }
  System.out.println(sum / gcs);
}
```



Also works in V8

Rate: 0.98 byte/s

Least Common Mechanism

Minimize the amount of mechanism common to more than one user and depended on by all users. Every shared mechanism (especially one involving shared variables) represents a potential information path between users and must be designed with great care to be sure it does not unintentionally compromise security.

Further, any mechanism serving all users must be certified to the satisfaction of every user, a job presumably harder than satisfying only one or a few users.

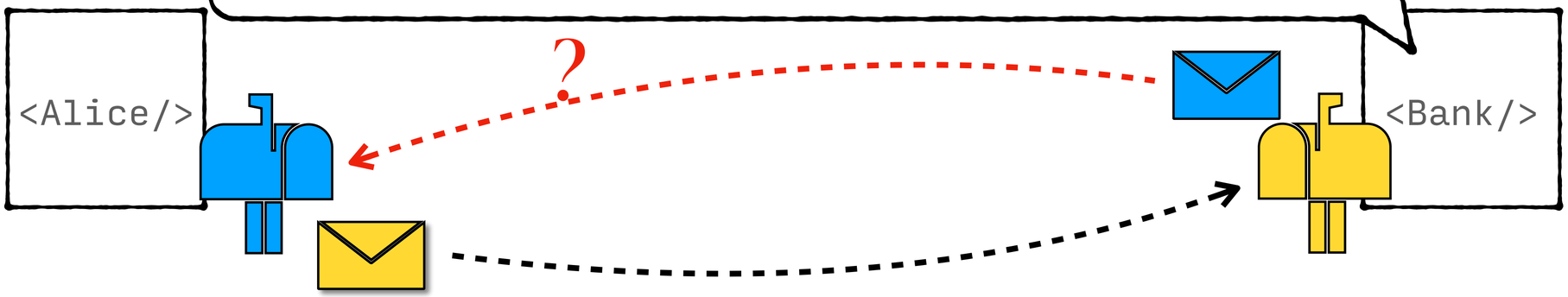
Structured Programming Language
Compiler
User-level Instruction Set Architecture
Operating system
Privileged Instruction Set Architecture
Processor
Hardware description language

Traffic analysis

Traffic analysis

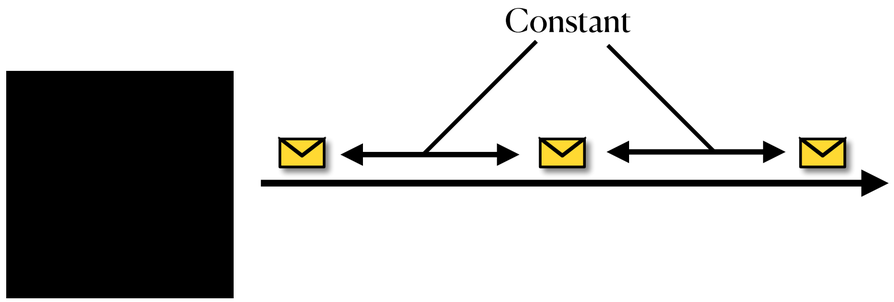
Example

```
TRANSFER(from: int, amount: int, to: int) {
  if amount <= balance[from]
  then {
    balance[from] = balance[from] - amount;
    balance[to]   = balance[to]   + amount;
  }
  else send(ALICE, “ERROR!”);
}
```



Mitigating traffic analysis

System-level mitigation



Independent link padding

Dependent link padding

- Treat program as black-box
- Two main approaches
  - Independent-link padding: Commonly, constant rate of fixed-size packets
  - Dependent-link padding: Shape of outgoing traffic computed from the shape of incoming traffic
- Prohibitive overheads in practice: traffic or latency[1]

[1] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, "Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail," in 2012 IEEE symposium on security and privacy. IEEE, 2012, pp. 332–346

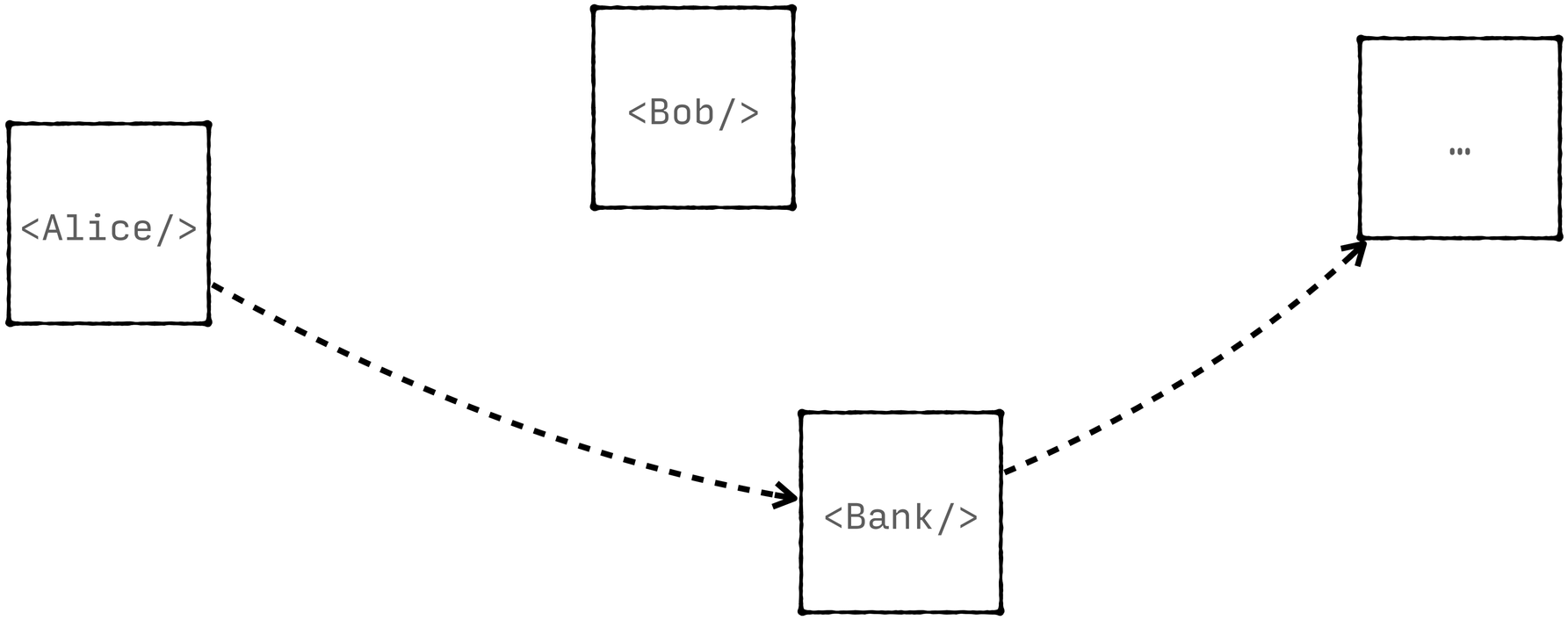
D. Das, S. Meiser, E. Mohammadi, and A. Kate, "Anonymity trilemma: Strong anonymity, low bandwidth overhead, low latency - choose two," IACR Cryptology ePrint Archive, vol. 2017, p. 954, 2017.

Idea: use language-level techniques

```
RELAY(x: int) {
  if cnd
  then send(FORWARD, x);
  else skip;
}
```

- Traffic padding only needed if `cnd` is secret
  - Not known at the system level
  - But doable if we know the source!

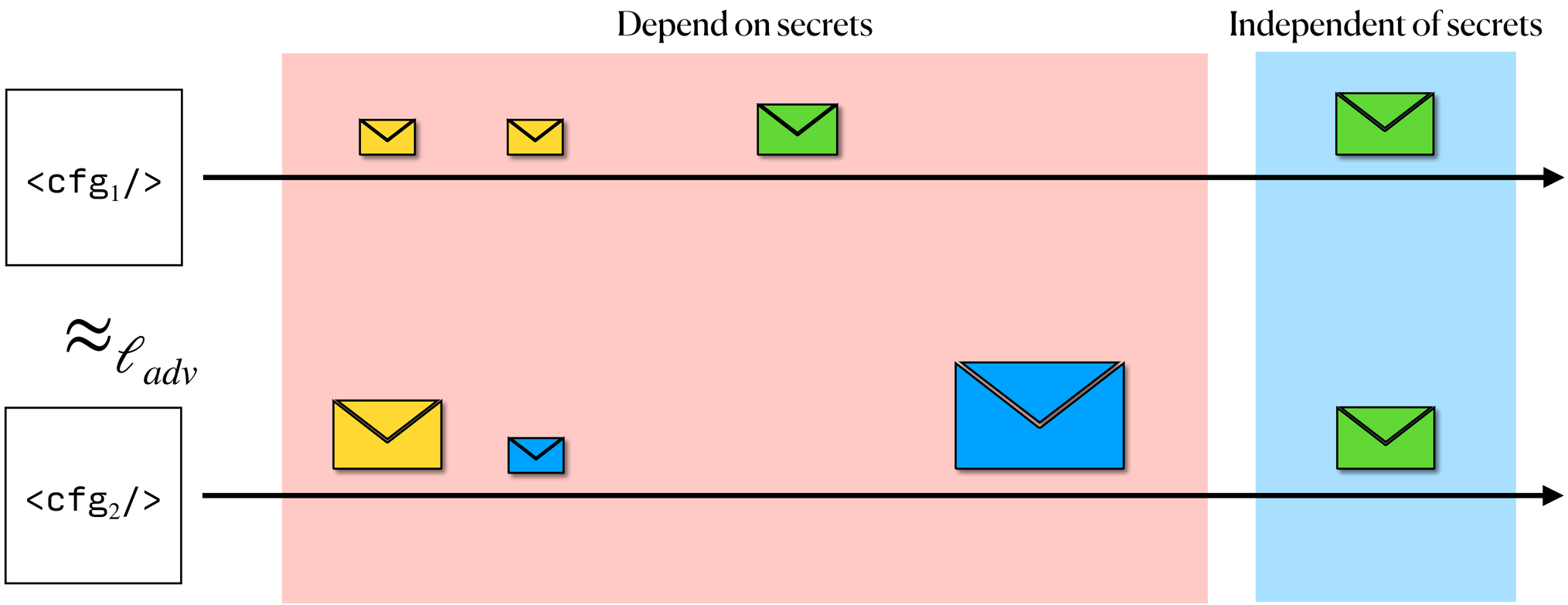
OblivIO



- All network nodes run OblivIO
- Each node has an associated security level
- Attacker is a network and/or nodes with low security

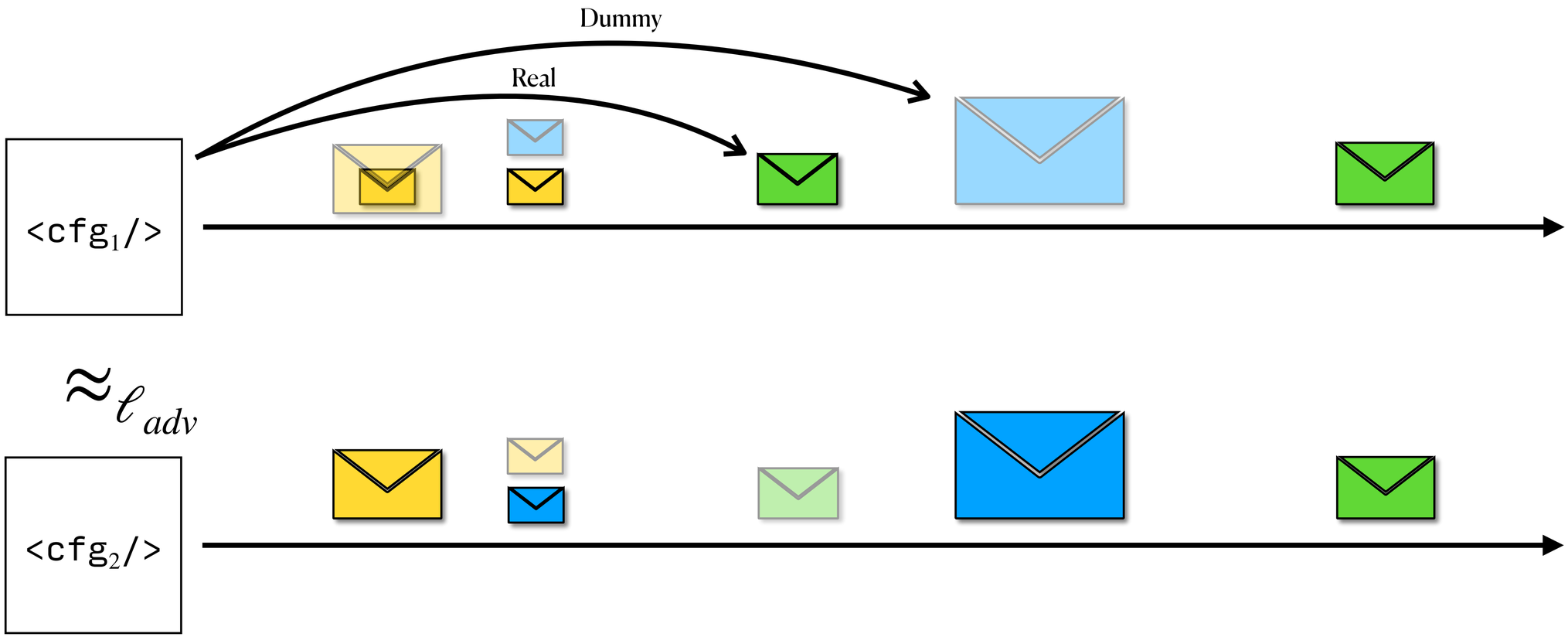
Mitigating traffic analysis

What must be protected?



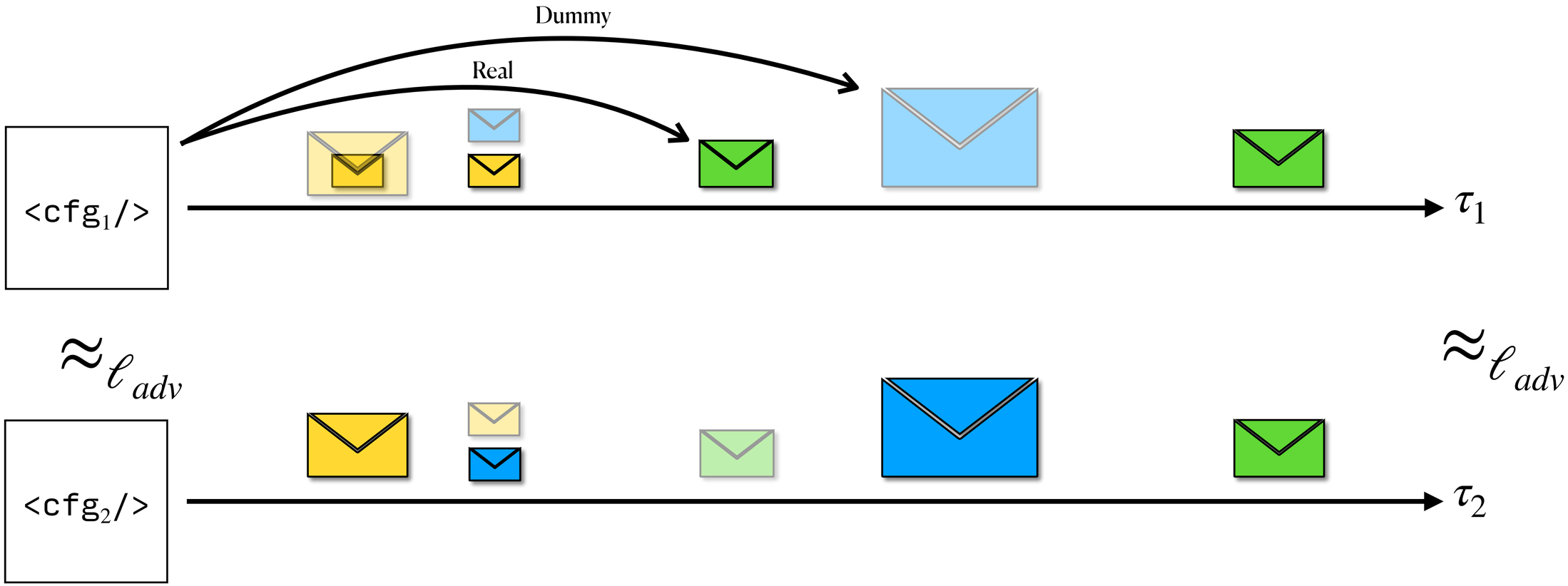
Mitigating traffic analysis

OblivIO: Traffic padding guided by program source



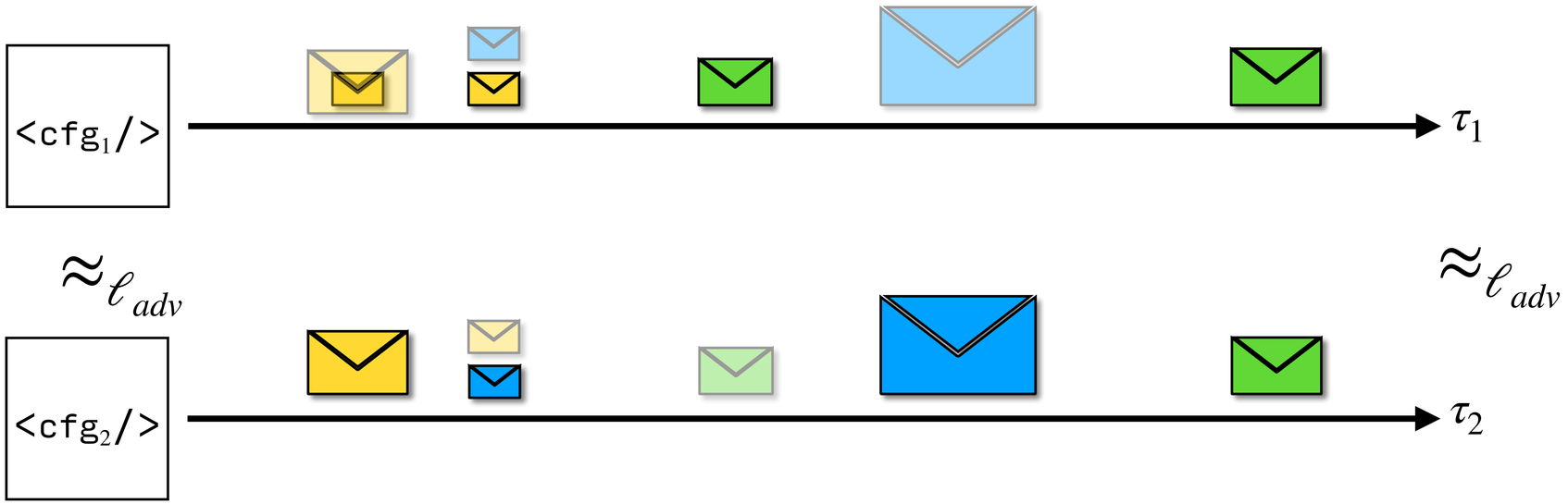
Mitigating traffic analysis

OblivIO: Traffic padding guided by program source



Mitigating traffic analysis

OblivIO: Traffic padding guided by program source



$$k(cfg_1, \tau_1, \ell_{adv}) \triangleq \{cfg_2 \mid cfg_1 \approx_{\ell_{adv}} cfg_2 \wedge cfg_2 \longrightarrow^*_{\tau_2} cfg_2' \wedge \tau_1 \approx_{\ell_{adv}} \tau_2\}$$

Attacker knowledge*

$$k(cfg_1, \tau_1 \cdot \alpha_1, \ell_{adv}) \supseteq k(cfg_1, \tau_1, \ell_{adv})$$

Progress-sensitive noninterference (PSNI)

* Askarov and A. Sabelfeld, “Gradual release: Unifying declassification, encryption and key release policies,” 2007 IEEE Symposium on Security and Privacy.

OblivIO

Language and syntax

- Simple imperative language for reactive programs
- Two execution modes: *real* and *phantom*
  - Data-obliviousness* — control-flow is never secret
- Formal model includes computational history for computing timestamp**

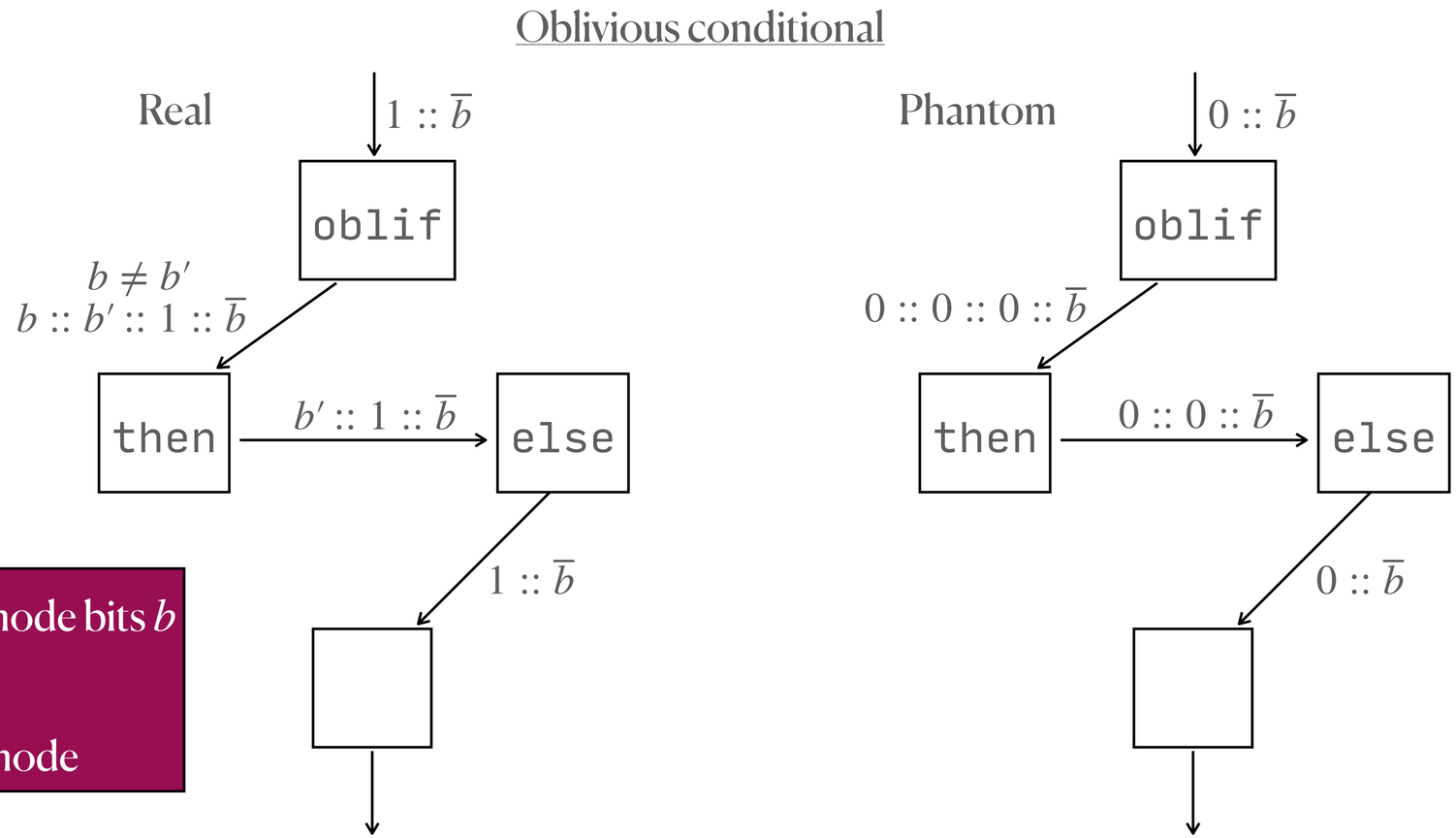
$$
\begin{aligned}
p &::= \cdot \mid ch(x)\{c\}; p \\
c &::= \texttt{skip} \mid c_1; c_2 \mid x = e \mid \texttt{if } e \texttt{ then } c \texttt{ else } c \mid \texttt{while } e \texttt{ do } c \mid \texttt{send}(ch, e) \\
&\quad \mid \texttt{oblif } e \texttt{ then } c \texttt{ else } c \qquad \text{(* Oblivious conditional — executes both branches *)} \\
&\quad \mid x \mathbin{?}= e \qquad \text{(* Oblivious, padding assignment *)} \\
&\quad \mid x \mathbin{?}= \texttt{input}(ch, e) \qquad \text{(* Local input *)}
\end{aligned}
$$

* S. Zahur and D. Evans, "Obliv-c: A language for extensible data-oblivious computation," IACR Cryptol. ePrint Arch., p. 1153, 2015. [Online]. Available: http://eprint.iacr.org/2015/1153

** Daniel Hedin and David Sands. Timing aware information flow security for a javacard-like bytecode. *Electronic Notes in Theoretical Computer Science*, 141 (1):163–182, 2005.

Oblivious semantics

Control flow



$\overline{b}$ is a stack of execution mode bits b

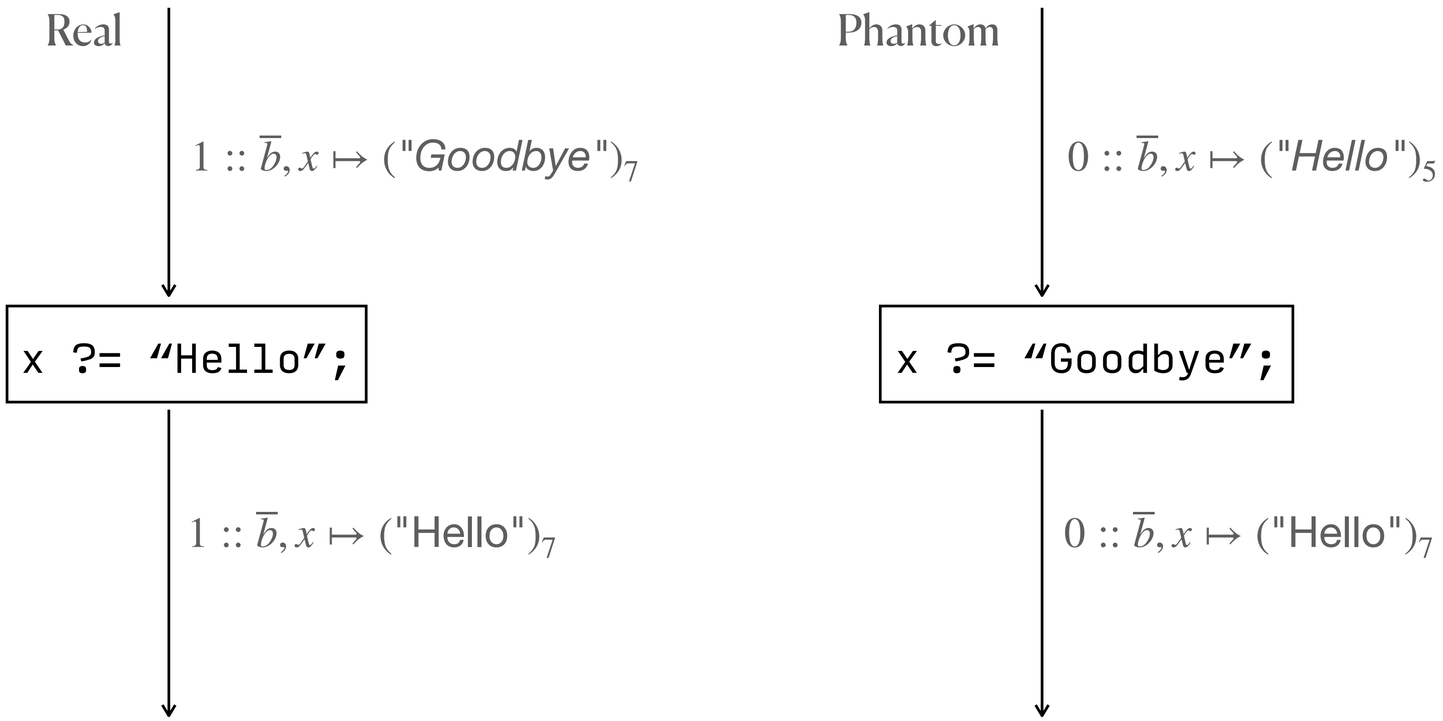
$b = 1$ denotes *real* mode

$b = 0$ denotes *phantom* mode

Oblivious semantics

Assignment

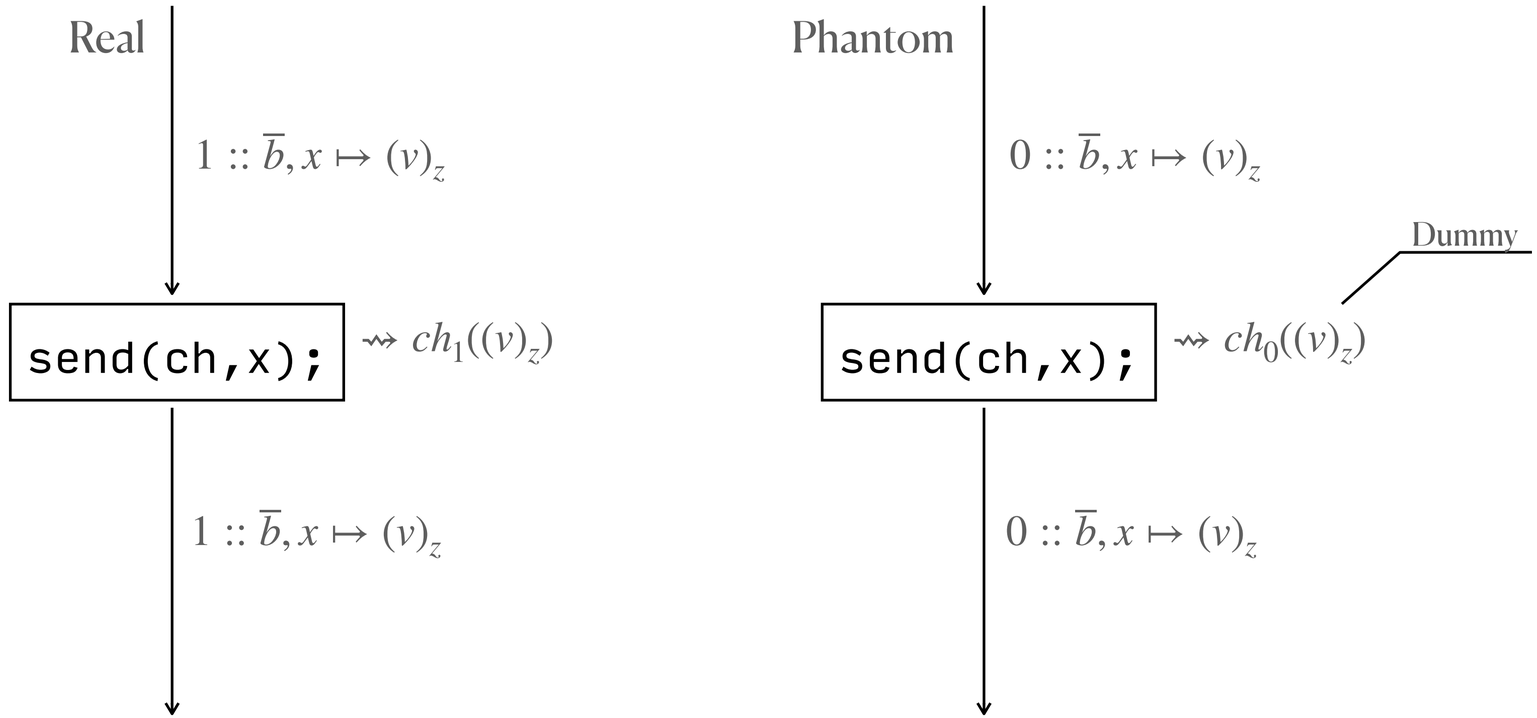
Oblivious assignment



Oblivious semantics

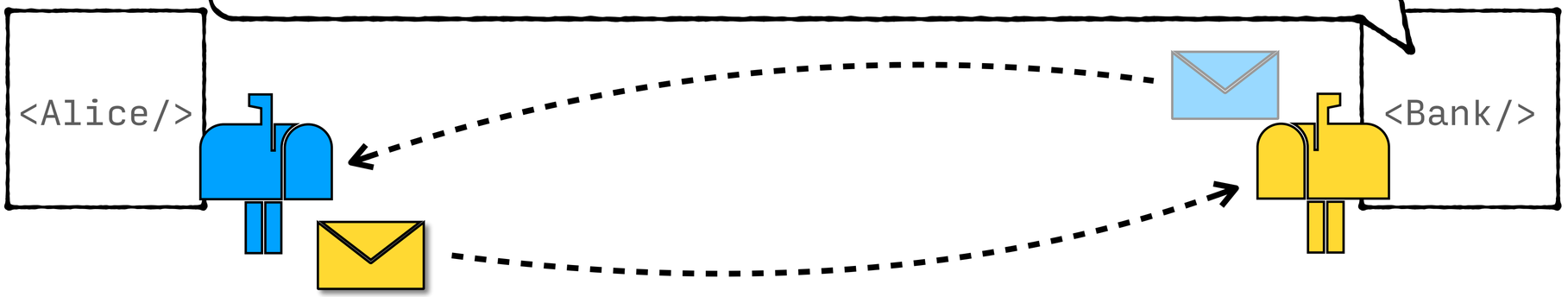
Sending

Send



Transfer example in OblivIO

```
TRANSFER(from: int, amount: int, to: int) {
  oblif amount <= balance[from]
  then {
    balance[from] ?= balance[from] - amount;
    balance[to]   ?= balance[to]   + amount;
  }
  else send(ALICE, “ERROR!”);
}
```



Type system

Part a

Public guard

T-If

$$\frac{\Gamma;\Delta \vdash e : int@\bot \qquad \Gamma,\Pi,\Lambda;\Delta;pc \vdash c_1 \qquad \Gamma,\Pi,\Lambda;\Delta;pc \vdash c_2}{\Gamma,\Pi,\Lambda;\Delta;pc \vdash \texttt{if } e \texttt{ then } c_1 \texttt{ else } c_2}$$

Non-public guard

T-OblivIf

$$\frac{\Gamma;\Delta \vdash e : int@\ell \qquad \ell \neq \bot \qquad \Gamma,\Pi,\Lambda;\Delta;pc \sqcup \ell \vdash c_1 \qquad \Gamma,\Pi,\Lambda;\Delta;pc \sqcup \ell \vdash c_2}{\Gamma,\Pi,\Lambda;\Delta;pc \vdash \texttt{oblif } e \texttt{ then } c_1 \texttt{ else } c_2}$$

T-Assign

$$\frac{x \notin dom(\Delta) \qquad \Gamma(x) = \sigma@\ell_x \qquad \Gamma;\Delta \vdash e : \sigma@\ell_e \qquad \ell_e \sqsubseteq \ell_x}{\Gamma,\Pi,\Lambda;\Delta;\bot \vdash x = e}$$

Public pc

T-OblivAssign

$$\frac{x \notin dom(\Delta) \qquad \Gamma(x) : \sigma@\ell_x \qquad \Gamma;\Delta \vdash e : \sigma@\ell_e \qquad \ell_e \sqcup pc \sqsubseteq \ell_x}{\Gamma,\Pi,\Lambda;\Delta;pc \vdash x \mathrel{?}= e}$$

Any pc

T-Send

$$\frac{\Gamma;\Delta \vdash e : \sigma@\ell_e \qquad \Lambda(ch) = \sigma@\ell_{mode};\ell_{val} \qquad pc \sqsubseteq \ell_{mode} \qquad \ell_e \sqsubseteq \ell_{val}}{\Gamma,\Pi,\Lambda;\Delta;pc \vdash \texttt{send}(ch, e)}$$

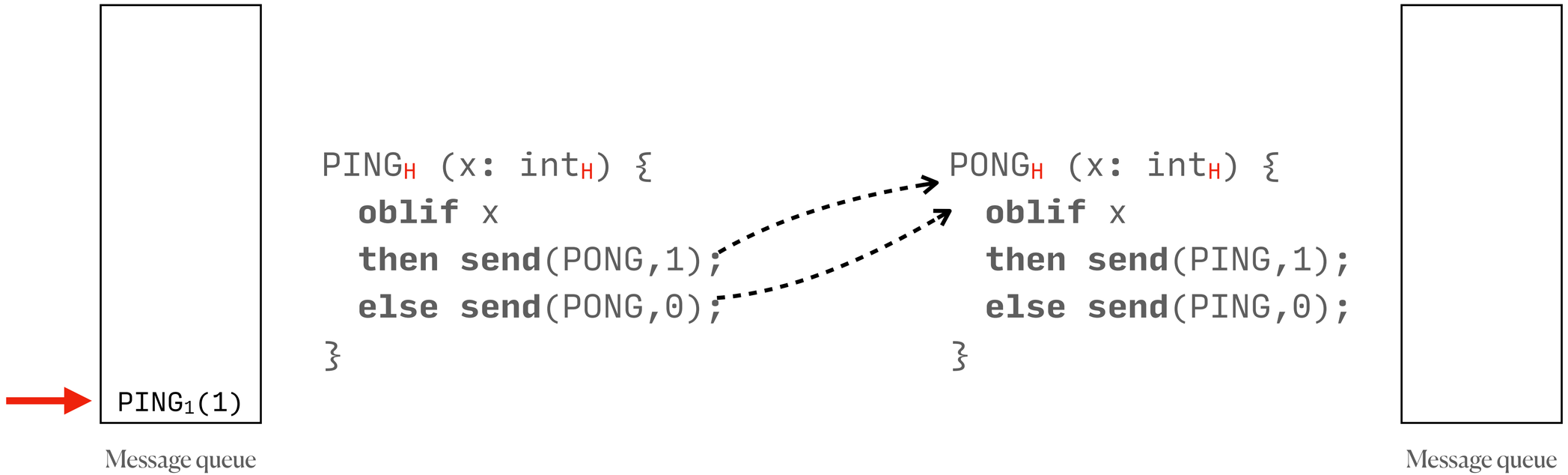
Soundness Theorem:

Well-typed OblivIO programs do not leak by their traffic patterns

$$k(cfg, \tau \cdot \alpha, \ell_{adv}) \supseteq k(cfg, \tau, \ell_{adv})$$

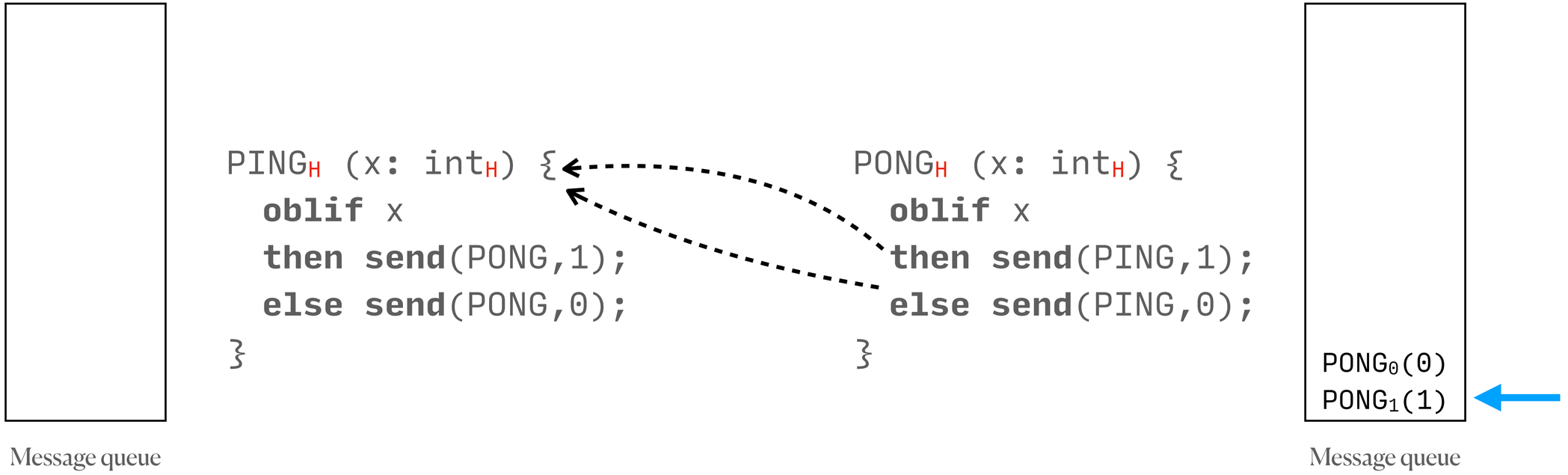
Secure, but at what cost...

A pitfall of oblivious execution



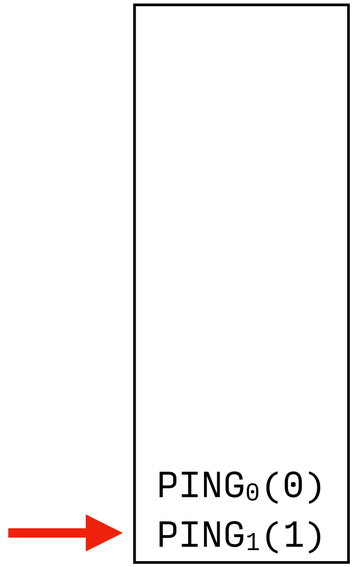
Secure, but at what cost...

A pitfall of oblivious execution



Secure, but at what cost...

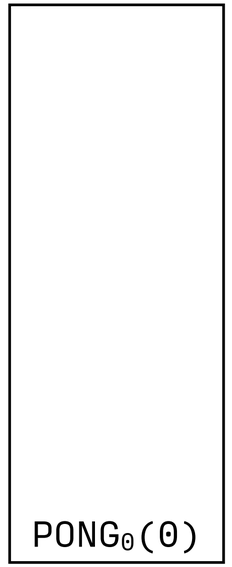
A pitfall of oblivious execution



Message queue

```
PINGH (x: intH) {
  oblif x
  then send(PONG,1);
  else send(PONG,0);
}
```

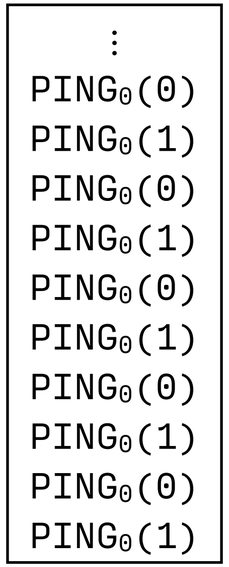
```
PONGH (x: intH) {
  oblif x
  then send(PING,1);
  else send(PING,0);
}
```



Message queue

Secure, but at what cost...

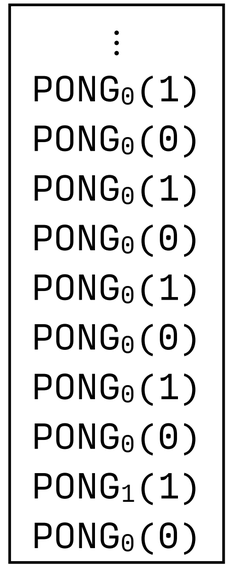
A pitfall of oblivious execution



Message queue

```
PINGH (x: intH) {
  oblif x
  then send(PONG,1);
  else send(PONG,0);
}
```

```
PONGH (x: intH) {
  oblif x
  then send(PING,1);
  else send(PING,0);
}
```



Message queue

Idea:

Statically restrict the amount of dummy traffic produced by a program

Restricting the amount of dummy traffic

Resource awareness*

- Declare integer *potential* q of a handler
  - Spend potential when sending obliviously
  - Oblivious send on channel with potential q costs $1 + q$
    - 1 to pay for the message itself
    - q to pay for the potential of the handler
- Instrument typing judgements with potentials

* J. Hoffmann and M. Hofmann, “Amortized resource analysis with polynomial potential,” in European Symposium on Programming. Springer, 2010, pp. 287–306.
J. Hoffmann, K. Aehlig, and M. Hofmann, “Resource aware ml,” in International Conference on Computer Aided Verification. Springer, 2012, pp. 781–786.

Adding potentials

T-If

$$\frac{\Gamma; \Delta \vdash e : int@\bot \qquad \Gamma, \Pi, \Lambda; \Delta; pc \vdash\ c_1 \qquad \Gamma, \Pi, \Lambda; \Delta; pc \vdash\ c_2}{\Gamma, \Pi, \Lambda; \Delta; pc \vdash\ \texttt{if}\ e\ \texttt{then}\ c_1\ \texttt{else}\ c_2}$$

T-OblivIf

$$\frac{\Gamma; \Delta \vdash e : int@\ell \qquad \ell \neq \bot \qquad \Gamma, \Pi, \Lambda; \Delta; pc \sqcup \ell \vdash\ c_1 \qquad \Gamma, \Pi, \Lambda; \Delta; pc \sqcup \ell \vdash\ c_2}{\Gamma, \Pi, \Lambda; \Delta; pc \vdash\ \texttt{oblif}\ e\ \texttt{then}\ c_1\ \texttt{else}\ c_2}$$

T-Send

$$\frac{\Gamma; \Delta \vdash e : \sigma@\ell_e \qquad \Lambda(ch) = \sigma@\ell_{mode}; \ell_{val} \qquad pc \sqsubseteq \ell_{mode} \qquad \ell_e \sqsubseteq \ell_{val}}{\Gamma, \Pi, \Lambda; \Delta; pc \vdash\ \texttt{send}(ch, e)}$$

Adding potentials

T-If

$$\frac{\Gamma;\Delta \vdash e : int@\bot \qquad \Gamma,\Pi,\Lambda;\Delta;pc \vdash^{q} c_1 \qquad \Gamma,\Pi,\Lambda;\Delta;pc \vdash^{q} c_2}{\Gamma,\Pi,\Lambda;\Delta;pc \vdash^{q} \texttt{if } e \texttt{ then } c_1 \texttt{ else } c_2}$$

T-OblivIf

$$\frac{\Gamma;\Delta \vdash e : int@\ell \qquad \ell \neq \bot \qquad \Gamma,\Pi,\Lambda;\Delta;pc \sqcup \ell \vdash^{q_1} c_1 \qquad \Gamma,\Pi,\Lambda;\Delta;pc \sqcup \ell \vdash^{q_2} c_2}{\Gamma,\Pi,\Lambda;\Delta;pc \vdash^{q_1+q_2} \texttt{oblif } e \texttt{ then } c_1 \texttt{ else } c_2}$$

T-Send

$$\frac{\Gamma;\Delta \vdash e : \sigma@\ell_e \qquad \Lambda(ch) = \sigma@\ell_{mode};\ell_{val};r \qquad pc \sqsubseteq \ell_{mode} \qquad \ell_e \sqsubseteq \ell_{val} \qquad q' = \begin{cases} 0 & \text{if } pc = \bot \\ 1+r & \text{otherwise} \end{cases}}{\Gamma,\Pi,\Lambda;\Delta;pc \vdash^{q+q'} \texttt{send}(ch,e)}$$

Overhead Theorem:

- Given
  - (System-wide) OblivIO trace τ_1
  - (System-wide) Unpadded trace τ_2
    - Without *dummy* messages
- Then
  - $|\tau_1| \leq |\tau_2| * c$

Example revisited

```
PINGH $N (x: intH) {
  oblif x
  then send(PONG,1);
  else send(PONG,0);
}
```

$N ≥ 2+2*$M

```
PONGH $M (x: intH) {
  oblif x
  then send(PING,1);
  else send(PING,0);
}
```

$M ≥ 2+2*$N

Example: Round auction

```
var round_counter: int_L = 500;
var leader: string_H = "";
var leading_bid: int_H = 0;

BID_H $0 (name: string_H, bid: int_H) {
  oblif leading_bid < bid
  then {
    leader ?= name;
    leading_bid ?= bid;
  }
  else skip;
}

TICK_L $0 (dmy: int_L) {
  if round_counter > 0
  then {
    round_counter = round_counter - 1;
    send(AUCTIONTIMER/BEGIN, 2000);
    ... // send AUCTION_STATUS to all users
   } else {
     ... // send AUCTION_OVER to all users
   }
}
```

AUCTIONHOUSE

```
var max_bid: int_H = 432;

AUCTION_STATUS_L $1 (name: string_H, bid: int_H) {
    oblif bid < max_bid && name != "Alice"
    then send(AUCTIONHOUSE/BID, ("Alice", bid + 1));
    else skip;
}

AUCTION_OVER_L $0 (winner: string_H, winning_bid: int_H) {
    ...
}
```

ALICE

```
var c: int_L = 0;

BEGIN_L $0 (i: int_L) {
    c = i;
    while (c > 0) do {
        c = c - 1;
    }
    send(AUCTIONHOUSE/TICK, 0);
}
```

AUCTIONTIMER

OblivIO Limitations

- Events are network messages only
  - Cannot react to events with secret presence
- Constant-time implementation of all operations
- Programs are static
  - No dynamically registered handlers
  - Functions not first-class
- Channels not first-class

```
oblif secret
then ch ?= ALICE/GREET;
else ch ?= BOB/GREET;
send(ch,”Hello");
```

Privacy

- Security and privacy are bundled together in the scope of many conferences but their relationship is complex
- Many definitions of privacy across all of the CS
  - from contextual integrity in information systems
  - to differential privacy in theory
- *I don’t think I fully understand privacy*

Security or Privacy: Can You Have Both?

James Bret Michael, Naval Postgraduate School

Richard Kuhn and Jeffrey Voas, IEEE Fellow

Computer hosts a virtual roundtable with six experts to discuss cybersecurity versus electronic privacy.

Since 1971, there have been more than 300 articles published in *Computer* on the topic of electronic privacy and about three times that number on cybersecurity, with many of these articles covering both topics. Roughly a third of all these articles appeared within the past five years, indicating that there is a lot of interest in security and privacy. The
tics are also telling in that *Computer* competes fo
with other IEEE Computer Society publications
larly *IEEE Security and Privacy*, which was first p
in 2003.

In their seminal article from 2004, Avižien
treated security as one of the attributes of de
systems, defining dependability as "the a
deliver service that can justifiably be truste
made no reference to privacy. Likewise, that sa
Voas[2] mentioned security but not privacy as o

many quality attributes of software. Looking back at these articles, one might ask, for instance, "Why wouldn't we want both security and privacy, given our dependence on the navigation apps on our smartphones and that we need to not only trust in the integrity of the data these apps use to compute travel routes but also in the protection of the privacy of our location data? What are the tradeoffs among security, privacy, and other system attributes, such as testability?" In a systems context, it seems natural to think about the interplay between security and privacy requirements, poli-

Digital Object Identifier 10.1109/MC.2020.3004606
Date of current version: 4 September 2020

ROUNDTABLE PANELISTS

Matt Bishop is a professor of computer science and the codirector of the Computer Security Laboratory at the University of California, Davis. He is the author of *Computer Security: Art and Science* (Addison-Wesley Professional, second edition, 2018). Bishop received his Ph.D. in computer science from Purdue University. Contact him at mabishop@ucdavis.edu.

Dorothy Denning is an emeritus distinguished professor of Defense Analysis with the Naval Postgraduate School. She has authored numerous books and articles on computer security and privacy. Denning received her Ph.D. in computer science from Purdue University. In 2019, she received the Test of Time Award from the IEEE Technical Committee on Security and Privacy. She is a fellow of the Association for Computing Machinery. Contact her at dedennin@nps.edu.

Simson Garfinkel is the senior computer scientist for confidentiality and data access at the U.S. Census Bureau. Garfinkel received his Ph.D. in computer science from the Massachusetts Institute of Technology. He holds seven U.S. patents and has published extensively on cybersecurity and digital forensics. He is a Fellow of the IEEE and the Association for Computing Machinery. Contact him at simson.l.garfinkel@census.gov.

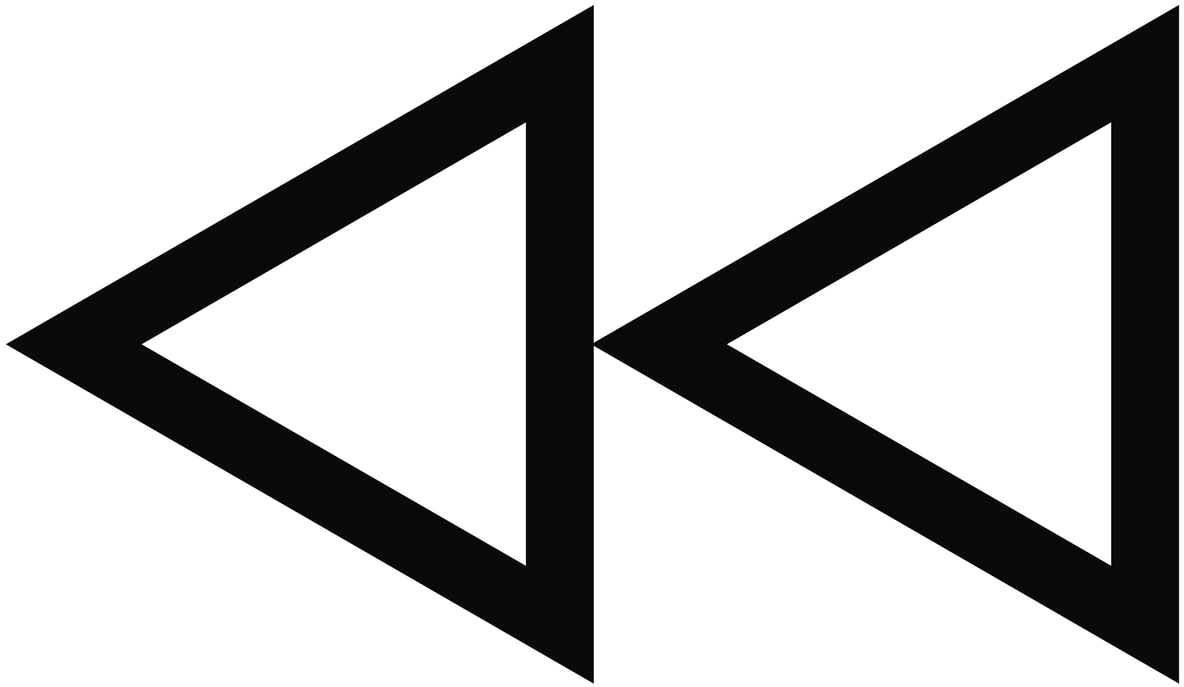
William Stallings is an independent consultant and the author of numerous textbooks on cybersecurity, cryptography, operating systems, and computer networking. His latest book is *Information Privacy and Privacy by Design* (Pearson, 2020). Stallings received his Ph.D. in computer science from the Massachusetts Institute of Technology. He is a member of the editorial board of *Cryptologia*. Contact him at wllmst@me.com.

David Thaw is an associate research professor of law and an assistant research professor of computing and information with the University of Pittsburgh, with expertise in cybersecurity, cybercrime, cyber warfare, and privacy. Thaw received his Ph.D. from the University of California, Berkeley. He is an affiliated fellow of the Information Society Project at Yale Law School. Contact him at dbthaw@pitt.edu.

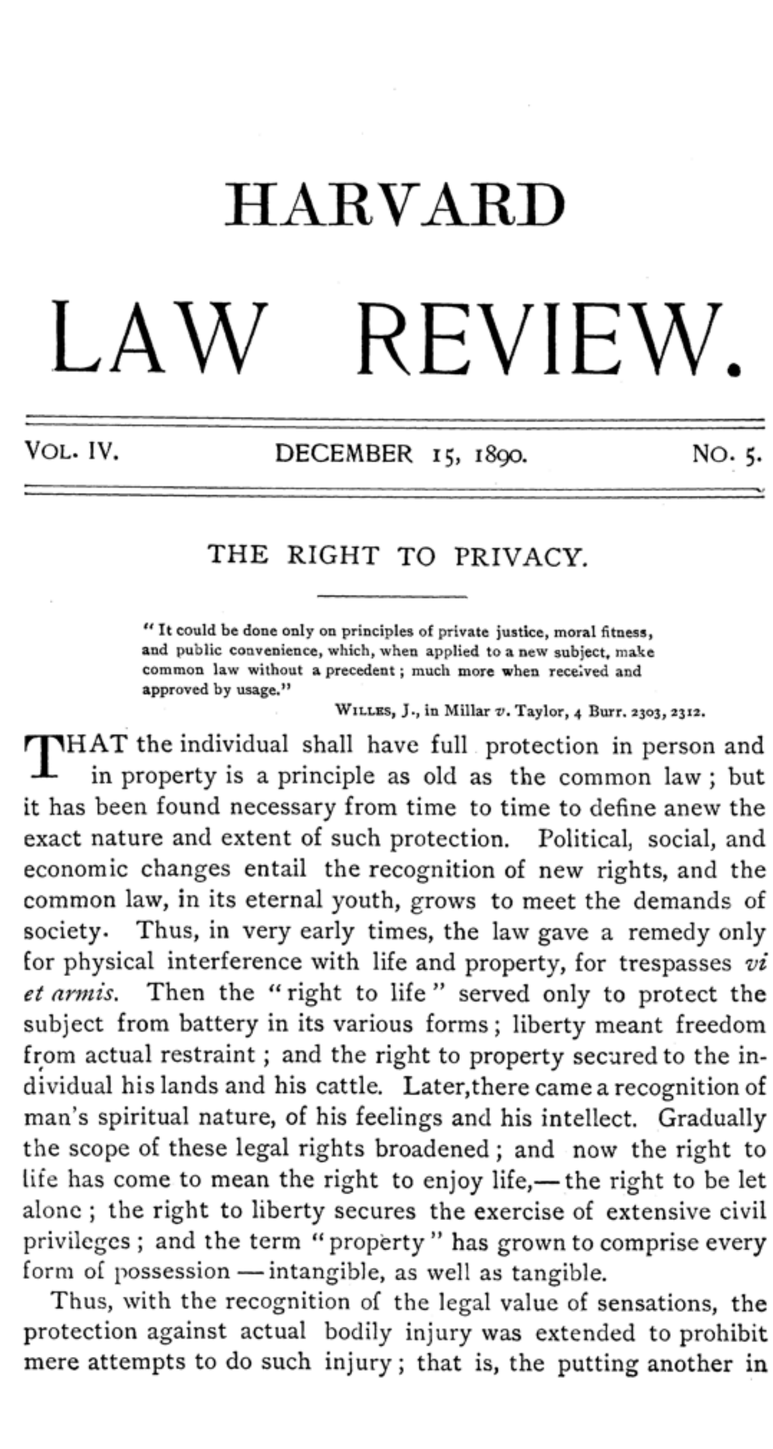
Duminda Wijesekera is a professor of computer science with George Mason University and a visiting research scientist at the National Institute of Standards and Technology. He has authored numerous articles on cybersecurity, privacy, and digital forensics. Wijesekera received a Ph.D. in mathematical logic from Cornell University and in computer science from the University of Minnesota. Contact him at dwijesek@gmu.edu.

D. Thaw: Privacy is a normative choice—values-based decisions we, as a society, make regarding policies, procedures, and goals respecting the degree to which citizens can enjoy limitations on intrusion or observation of certain aspects of their lives and property.

Security, by contrast, is an objective exercise; security does not have an inherent stake in how small or large the sphere of privacy is created. Once the appropriate social or political process determines the nature of privacy protections, it becomes the job of security to implement those protections within those guidelines.



1890



HARVARD

LAW REVIEW.

VOL. IV. DECEMBER 15, 1890. NO. 5.

THE RIGHT TO PRIVACY.

"It could be done only on principles of private justice, moral fitness, and public convenience, which, when applied to a new subject, make common law without a precedent; much more when received and approved by usage."

WILLES, J., in Millar *v.* Taylor, 4 Burr. 2303, 2312.

THAT the individual shall have full protection in person and in property is a principle as old as the common law; but it has been found necessary from time to time to define anew the exact nature and extent of such protection. Political, social, and economic changes entail the recognition of new rights, and the common law, in its eternal youth, grows to meet the demands of society. Thus, in very early times, the law gave a remedy only for physical interference with life and property, for trespasses *vi et armis*. Then the "right to life" served only to protect the subject from battery in its various forms; liberty meant freedom from actual restraint; and the right to property secured to the individual his lands and his cattle. Later, there came a recognition of man's spiritual nature, of his feelings and his intellect. Gradually the scope of these legal rights broadened; and now the right to life has come to mean the right to enjoy life,—the right to be let alone; the right to liberty secures the exercise of extensive civil privileges; and the term "property" has grown to comprise every form of possession—intangible, as well as tangible.

Thus, with the recognition of the legal value of sensations, the protection against actual bodily injury was extended to prohibit mere attempts to do such injury; that is, the putting another in

The Right To Privacy

[Warren and Brendeis, 1890]

the right to be let alone

Instantaneous photographs and newspaper enterprise have invaded the sacred precincts of private and domestic life; and numerous mechanical devices threaten to make good the prediction that "what is whispered in the closet shall be proclaimed from the house-tops"

Even gossip apparently harmless, when widely and persistently circulated, is potent for evil. It both belittles and perverts.

HARVARD

LAW REVIEW.

VOL. IV. DECEMBER 15, 1890. NO. 5.

THE RIGHT TO PRIVACY.

"It could be done only on principles of private justice, moral fitness, and public convenience, which, when applied to a new subject, make common law without a precedent; much more when received and approved by usage."

WILLES, J., in Millar *v.* Taylor, 4 Burr. 2303, 2312.

THAT the individual shall have full protection in person and in property is a principle as old as the common law; but it has been found necessary from time to time to define anew the exact nature and extent of such protection. Political, social, and economic changes entail the recognition of new rights, and the common law, in its eternal youth, grows to meet the demands of society. Thus, in very early times, the law gave a remedy only for physical interference with life and property, for trespasses *vi et armis*. Then the "right to life" served only to protect the subject from battery in its various forms; liberty meant freedom from actual restraint; and the right to property secured to the individual his lands and his cattle. Later, there came a recognition of man's spiritual nature, of his feelings and his intellect. Gradually the scope of these legal rights broadened; and now the right to life has come to mean the right to enjoy life,—the right to be let alone; the right to liberty secures the exercise of extensive civil privileges; and the term "property" has grown to comprise every form of possession—intangible, as well as tangible.

Thus, with the recognition of the legal value of sensations, the protection against actual bodily injury was extended to prohibit mere attempts to do such injury; that is, the putting another in

The Right To Privacy

[Warren and Brendeis, 1890]

Even gossip apparently harmless, when widely and persistently circulated, is potent for evil. It both belittles and perverts.

It belittles by inverting the relative importance of things, thus dwarfing the thoughts and aspirations of a people. When personal gossip attains the dignity of print, and crowds the space available for matters of real interest to the community, what wonder that the ignorant and thoughtless mistake its relative importance.

Cute Cat Theory of Censorship

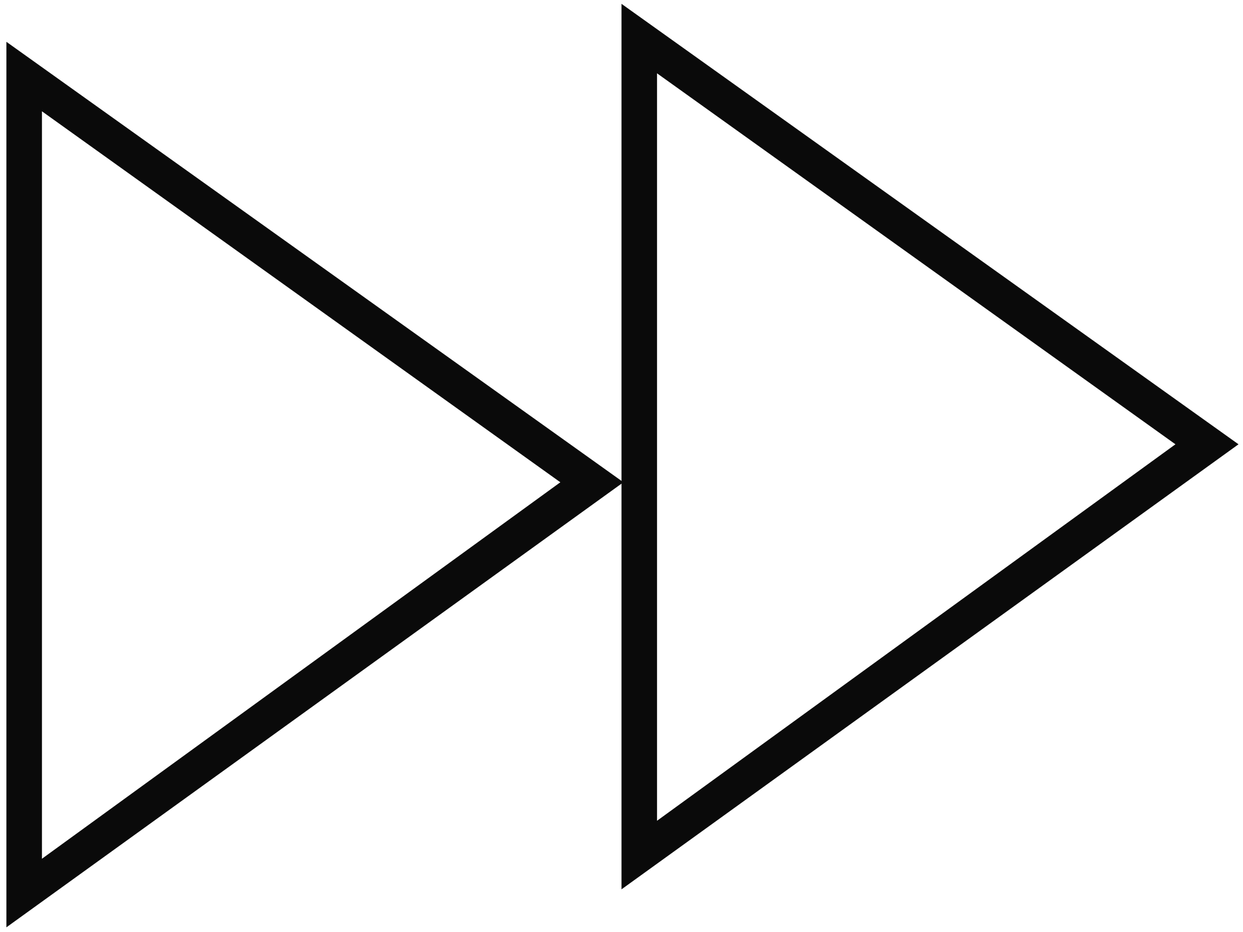
[Zuckerman' 13]

A single platform used for multiple purposes is harder to shut down and is therefore better for privacy

is at odds with

the Least Common Mechanism of S&S that argues for less sharing

Image credit: https://www.flickr.com/photos/goney/

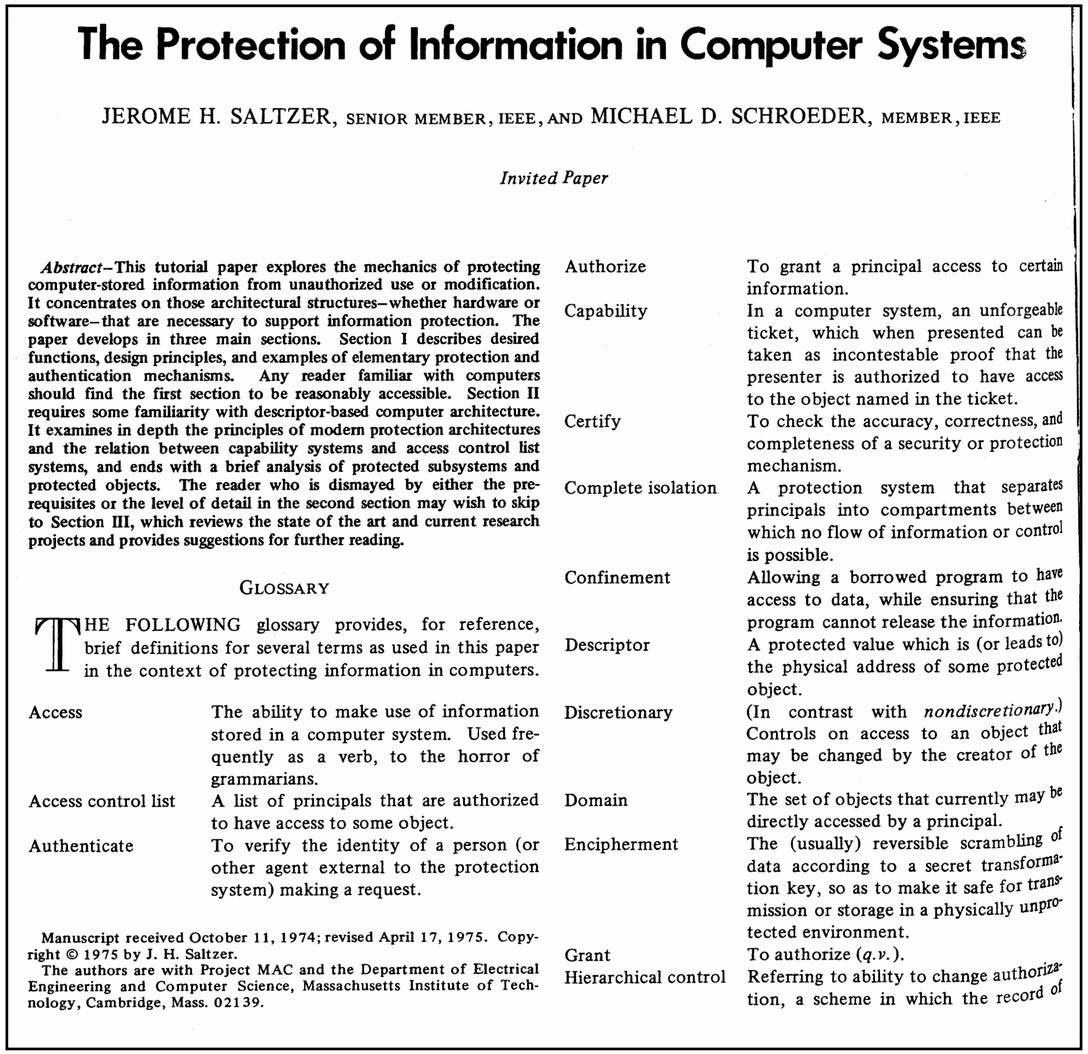


2025

Where are we now?

- Security > Memory safety
- LLMs
  - Untrusted code everywhere
    - Injection attacks?
  - But also
    - Can LLMs help us break the gravity of untrusted code
- Security principles remain ever-green

Saltzer & Schroeder's design principles, 1975



The Protection of Information in Computer Systems

JEROME H. SALTZER, SENIOR MEMBER, IEEE, AND MICHAEL D. SCHROEDER, MEMBER, IEEE

Invited Paper

Abstract—This tutorial paper explores the mechanics of protecting computer-stored information from unauthorized use or modification. It concentrates on those architectural structures—whether hardware or software—that are necessary to support information protection. The paper develops in three main sections. Section I describes desired functions, design principles, and examples of elementary protection and authentication mechanisms. Any reader familiar with computers should find the first section to be reasonably accessible. Section II requires some familiarity with descriptor-based computer architecture. It examines in depth the principles of modern protection architectures and the relation between capability systems and access control list systems, and ends with a brief analysis of protected subsystems and protected objects. The reader who is dismayed by either the prerequisites or the level of detail in the second section may wish to skip to Section III, which reviews the state of the art and current research projects and provides suggestions for further reading.

GLOSSARY

THE FOLLOWING glossary provides, for reference, brief definitions for several terms as used in this paper in the context of protecting information in computers.

Access	The ability to make use of information stored in a computer system. Used frequently as a verb, to the horror of grammarians.
Access control list	A list of principals that are authorized to have access to some object.
Authenticate	To verify the identity of a person (or other agent external to the protection system) making a request.
Authorize	To grant a principal access to certain information.
Capability	In a computer system, an unforgeable ticket, which when presented can be taken as incontestable proof that the presenter is authorized to have access to the object named in the ticket.
Certify	To check the accuracy, correctness, and completeness of a security or protection mechanism.
Complete isolation	A protection system that separates principals into compartments between which no flow of information or control is possible.
Confinement	Allowing a borrowed program to have access to data, while ensuring that the program cannot release the information.
Descriptor	A protected value which is (or leads to) the physical address of some protected object.
Discretionary	(In contrast with *nondiscretionary*.) Controls on access to an object that may be changed by the creator of the object.
Domain	The set of objects that currently may be directly accessed by a principal.
Encipherment	The (usually) reversible scrambling of data according to a secret transformation key, so as to make it safe for transmission or storage in a physically unprotected environment.
Grant	To authorize (*q.v.*).
Hierarchical control	Referring to ability to change authorization, a scheme in which the record of

Manuscript received October 11, 1974; revised April 17, 1975.
The authors are with Project MAC and the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, Mass. 02139.

Economy of mechanism

Fail-safe defaults

Complete mediation

Open design

Separation of privilege

Least privilege

Least common mechanism

Psychological acceptability