SoK: Secure Data Deletion

Joel Reardon, David Basin, Srdjan Capkun
ETH Zurich

2013-05-21

Secure deletion: the task of deleting data from a physical medium so that the data is irrecoverable.

Why is this needed?

- Secure deletion protects the security and privacy of sensitive data
- Secure deletion is lacking in most deployed systems
  - e.g., file systems are not designed to offer secure deletion
- Old solutions to this problem make many assumptions invalid for modern systems
  - e.g., simply overwriting a file with zeros fails for many systems
- Secure deletion is seeing active research
  - 7 of the 20 papers we survey in detail were published since 2010

Contributions of our systematization

- Define adversaries by their capabilities
  - Define a *strength* partial ordering among adversaries
- Determine solution properties and environmental assumptions
  - Developed by analyzing the existing literature
  - Define a *substitutability* partial ordering among solutions
- Organize solutions by the interface to which they access the storage medium
- Extensively survey the literature
  - determine each solution's properties and its defeated adversary
- Experiment with numerous user-level secure deletion tools
  - Show which file systems and use cases effect secure deletion.

Outline of this talk

- Define adversaries by their capabilities
  - Define a *strength* partial ordering among adversaries
- Determine solution properties and environmental assumptions
  - Developed by analyzing the existing literature
  - Define a *substitutability* partial ordering among solutions
- Organize solutions by the interface to which they access the storage medium
- Extensively survey the literature
  - determine each solution's properties and its defeated adversary
- Experiment with numerous user-level secure deletion tools
  - Show which file systems and use cases effect secure deletion.

Terminology

- **Data Objects**: Addressable units of data
  - e.g., blocks, DB records, SMS messages, files
- **Physical Medium**: Any device capable of storing and retrieving data objects
  - e.g., magnetic hard drive, flash memory, piece of paper
- **Interface**: How the user interacts with the physical medium; it transforms data objects into a form suitable for the physical medium
  - e.g., a file system

A **data object** is securely deleted from a **physical medium** if an adversary that is given some manner of access to the medium is unable to recover the deleted data object.

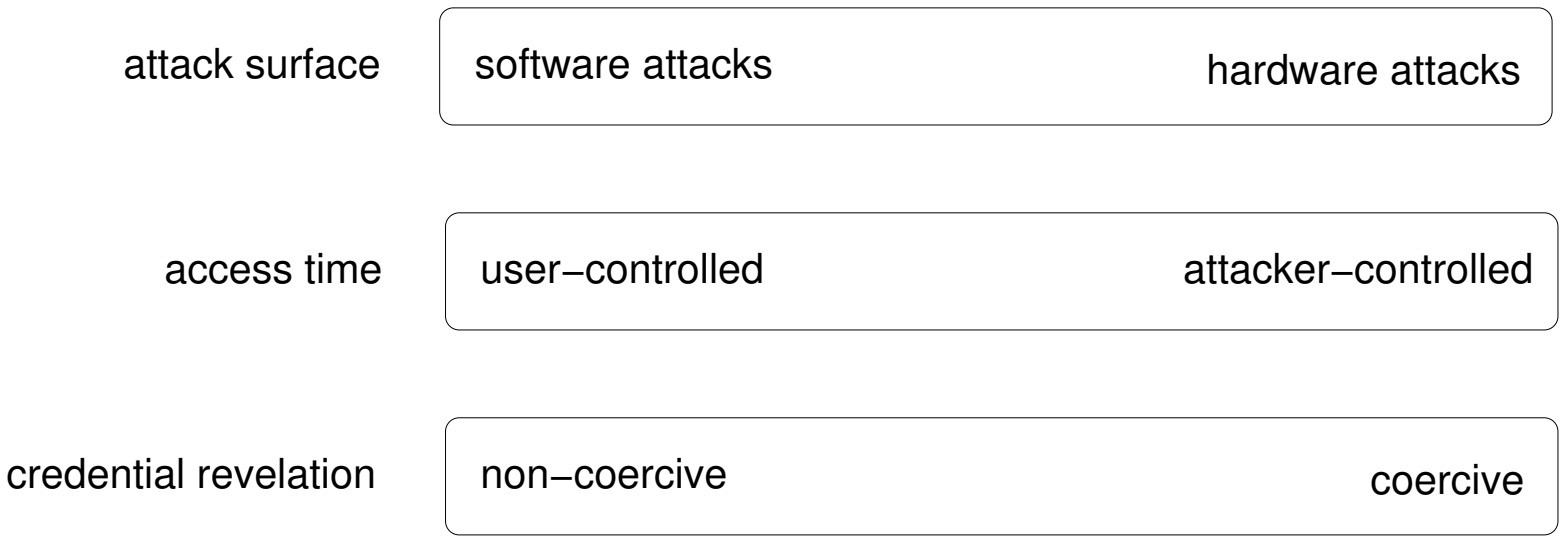
Classes of Adversarial Capabilities

- We divide capabilities into discrete **classes**
- Each class is fully ordered by adversarial strength
- An adversary is defined as a set of capabilities
  - one capability per class

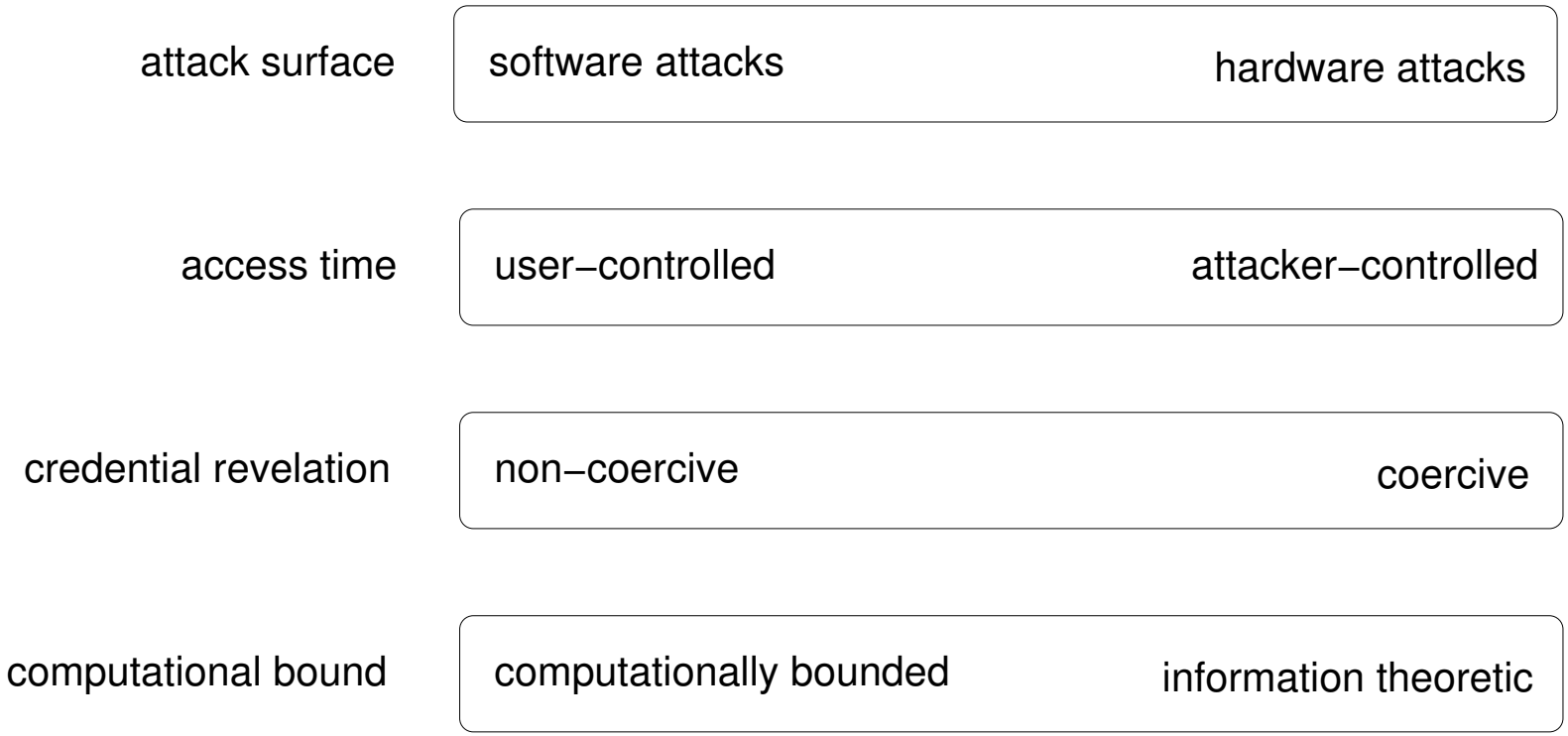
Classes of Adversarial Capabilities

Classes of Adversarial Capabilities

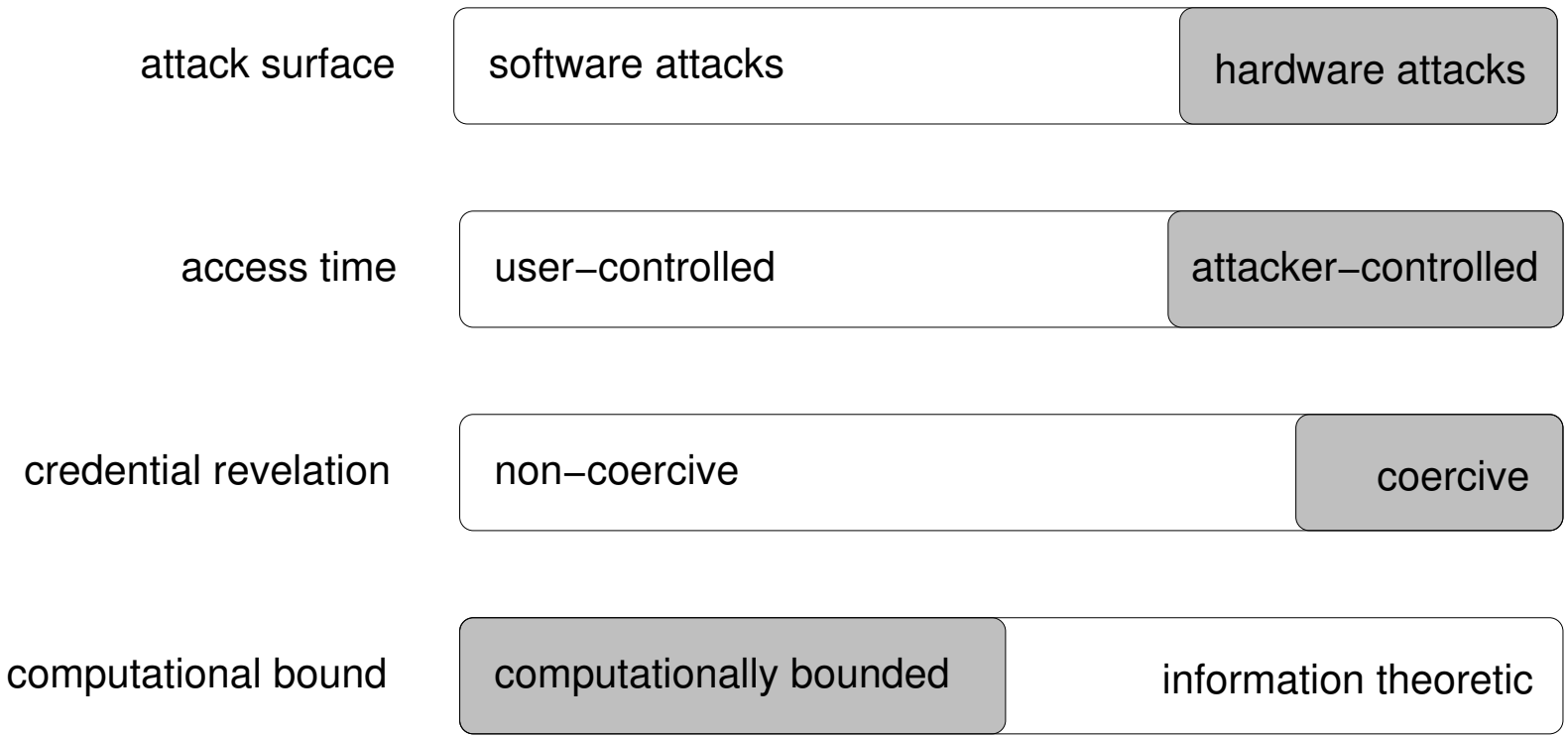
Classes of Adversarial Capabilities



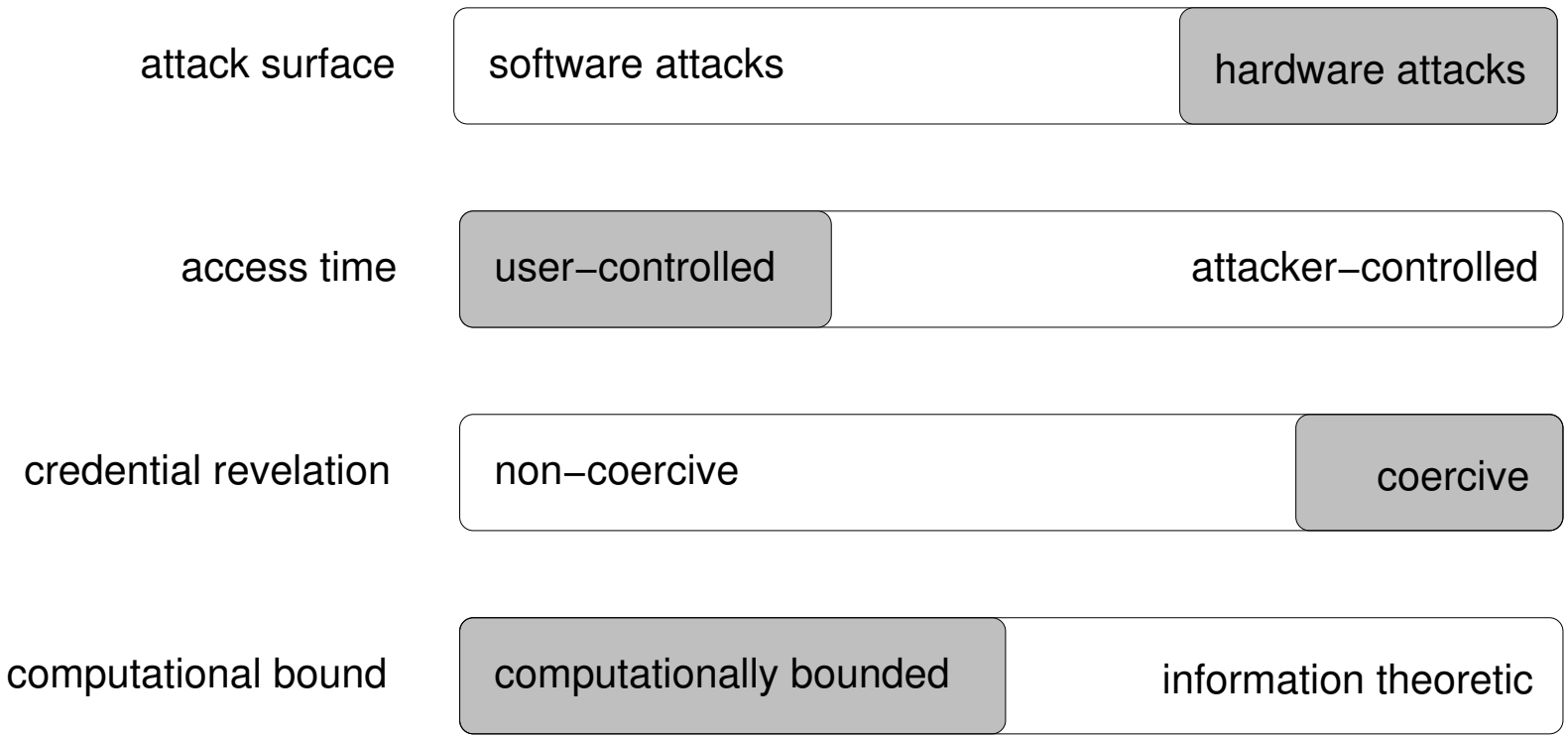
Classes of Adversarial Capabilities



Example: Subpoena Adversary



Example: Border Crossing



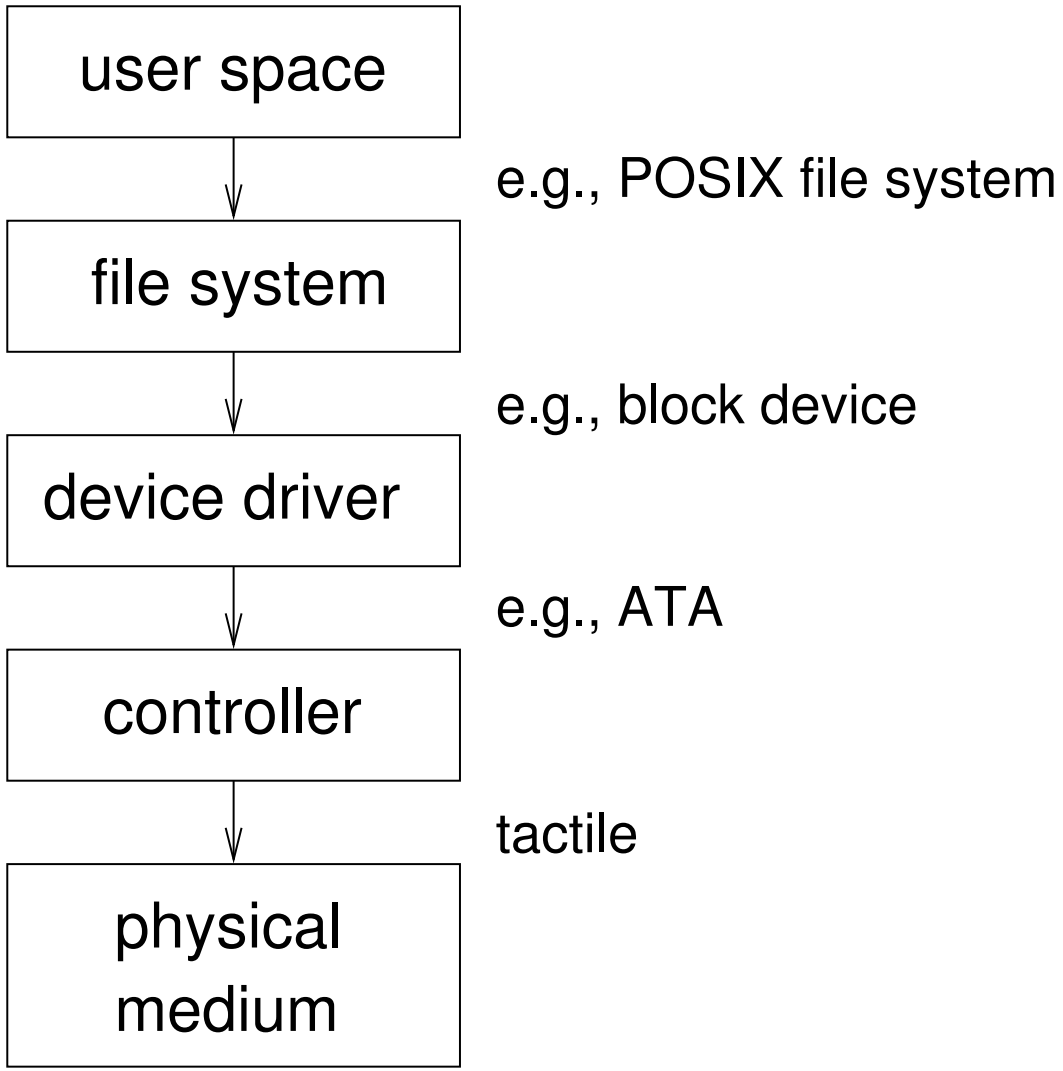
As adversaries differ in their strength, secure deletion solutions may differ by which adversaries they defeat. Solutions may also differ in other ways.

Some Properties of Secure Deletion Solutions

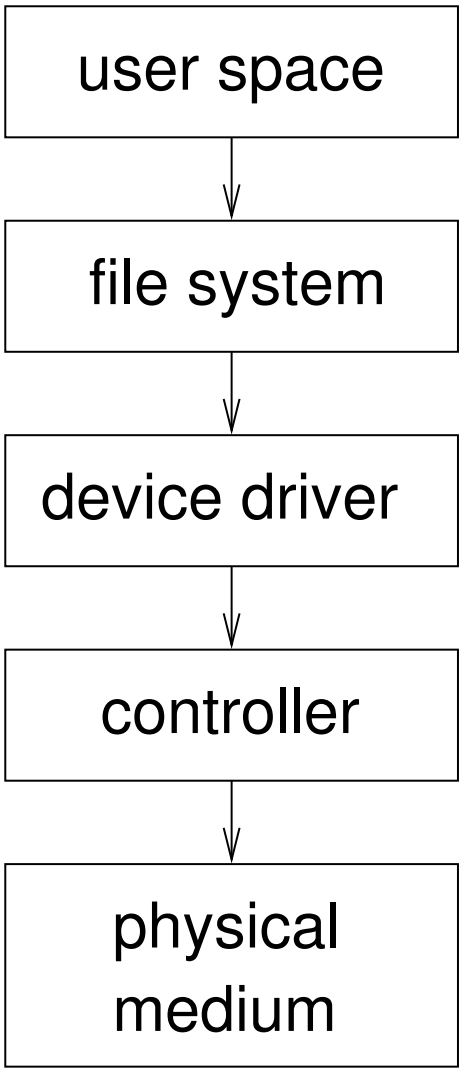
- **integration**: How does it integrate into a system? What assumptions does it make on the behaviour of the interface?
- **granularity**: Does it delete data at the smallest granularity possible, or only entire files, or only entire storage media?
- **scope**: Does it securely delete all discarded data, or just specially marked sensitive data?
- **lifetime**: Does it cause wear on the storage medium? does it destroy it?

We'll now take a closer look at the integration property.

Interface and Layers

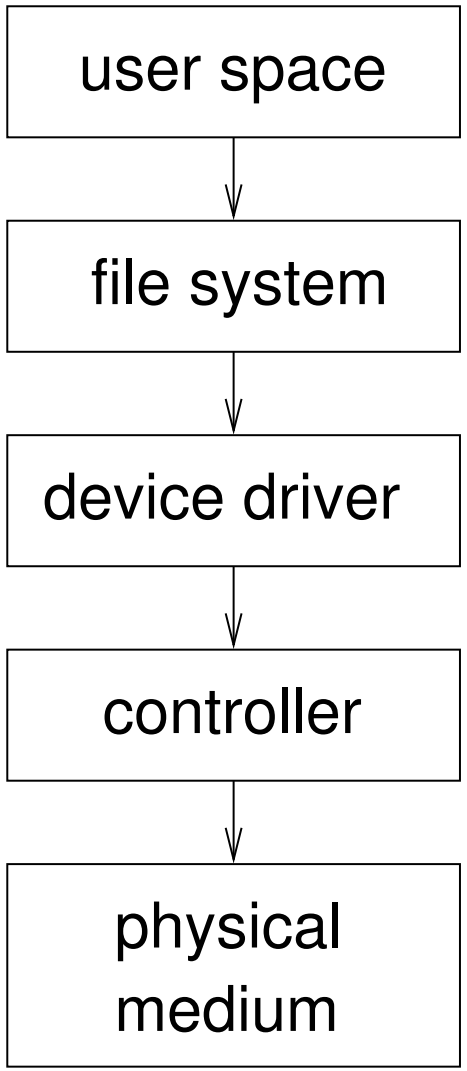


Interface and Layers



physical access
e.g., degaussing,
destruction, etc.

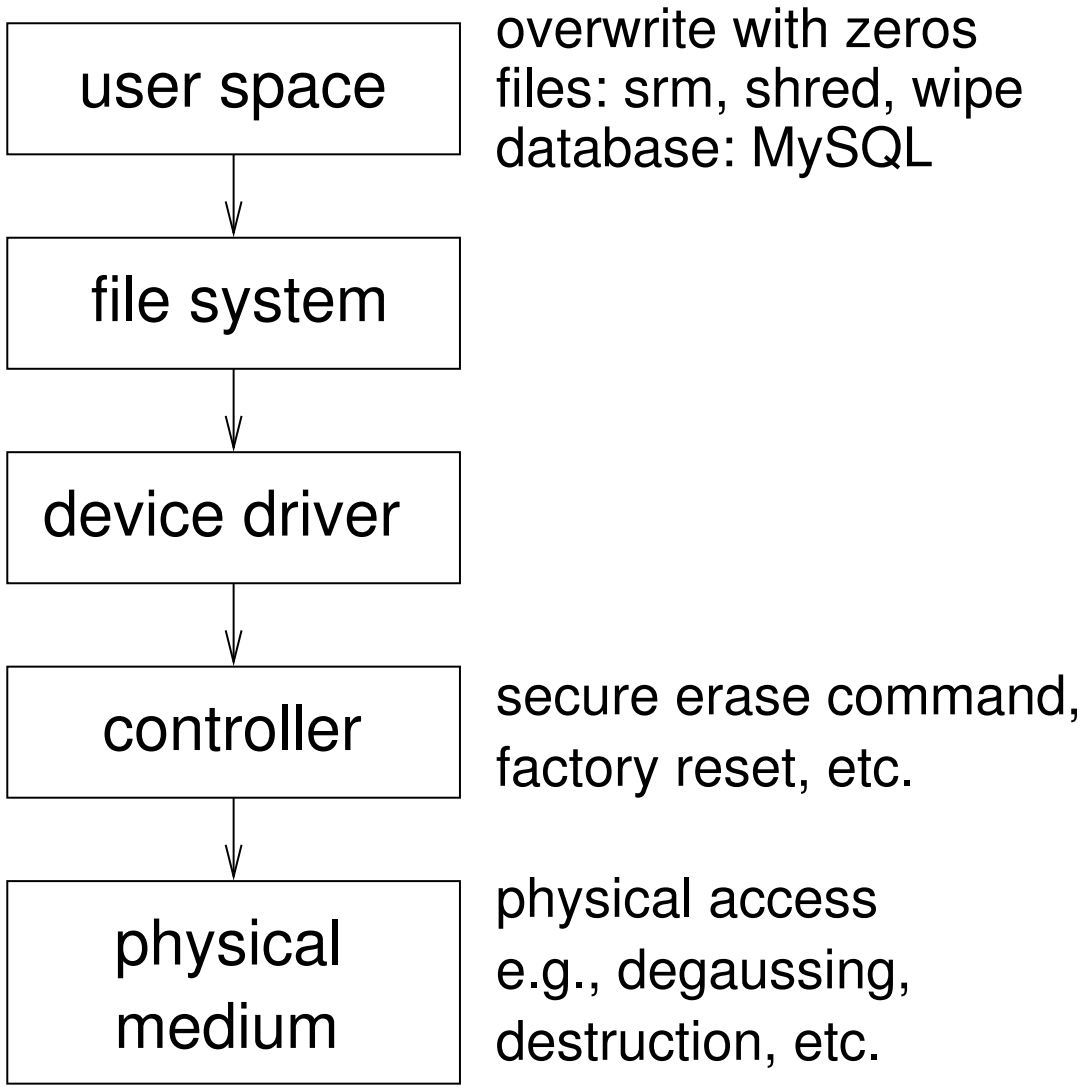
Interface and Layers



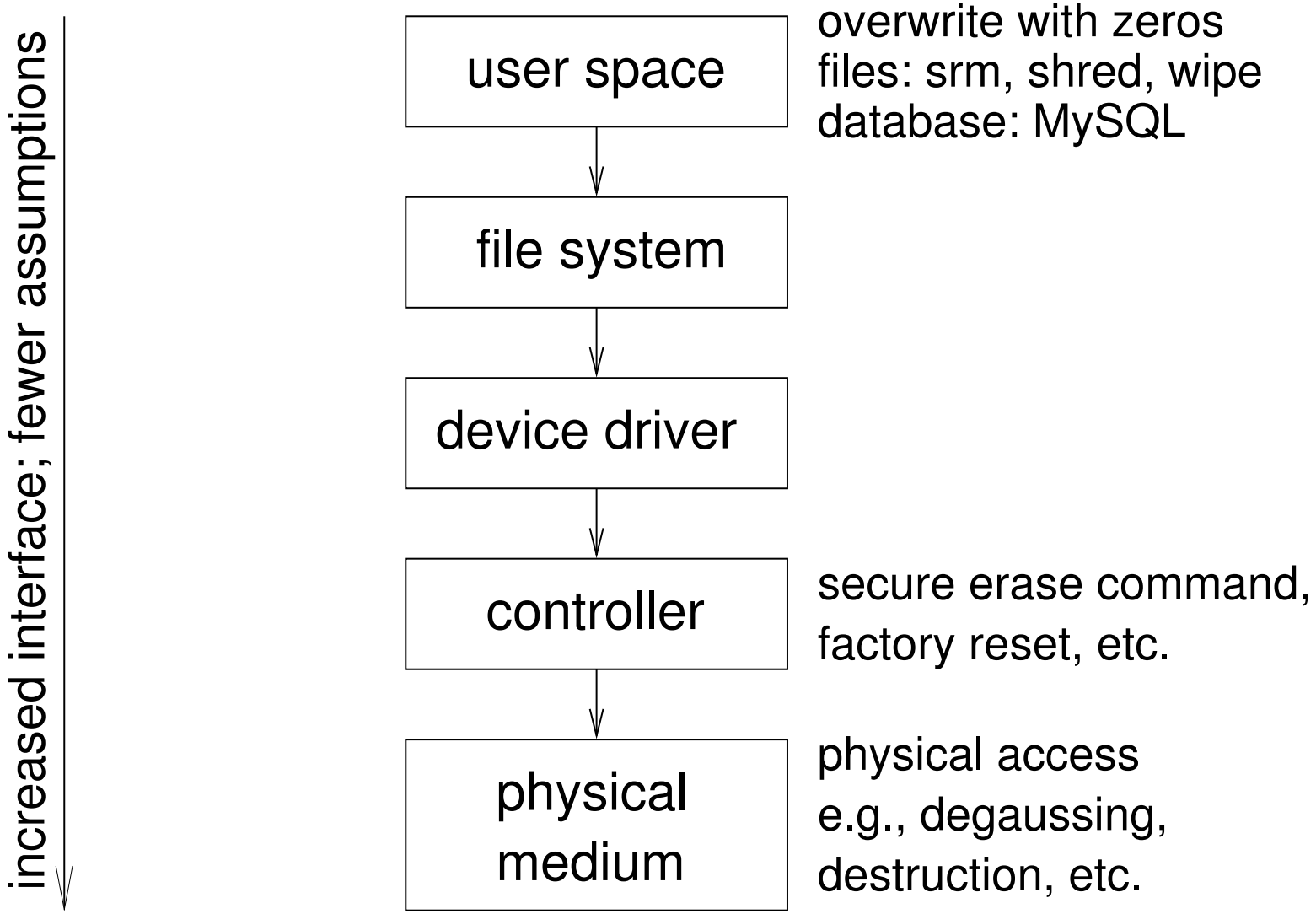
secure erase command,
factory reset, etc.

physical access
e.g., degaussing,
destruction, etc.

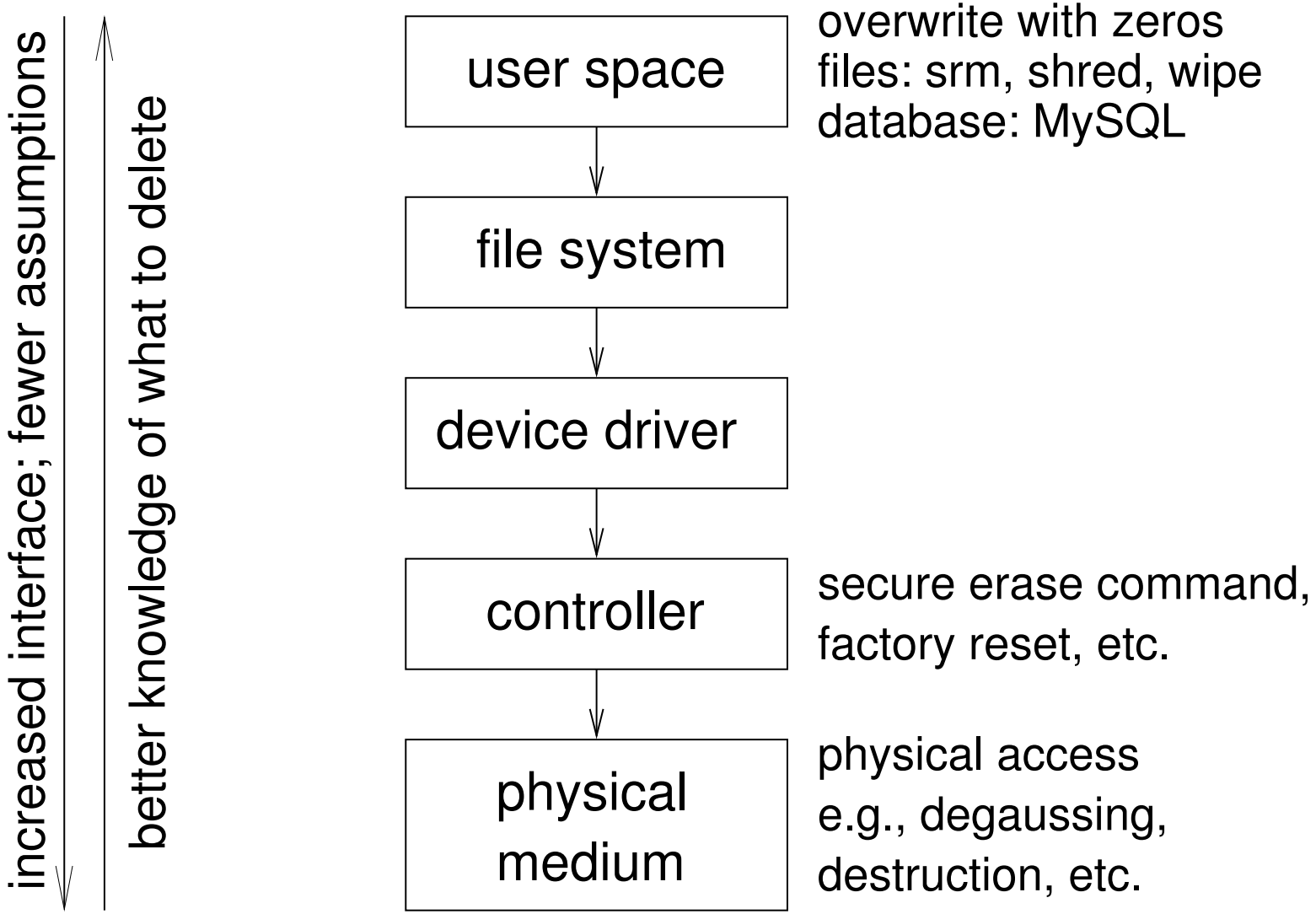
Interface and Layers



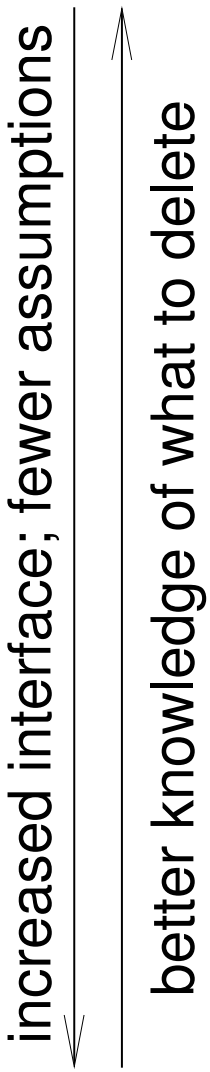
Interface and Layers

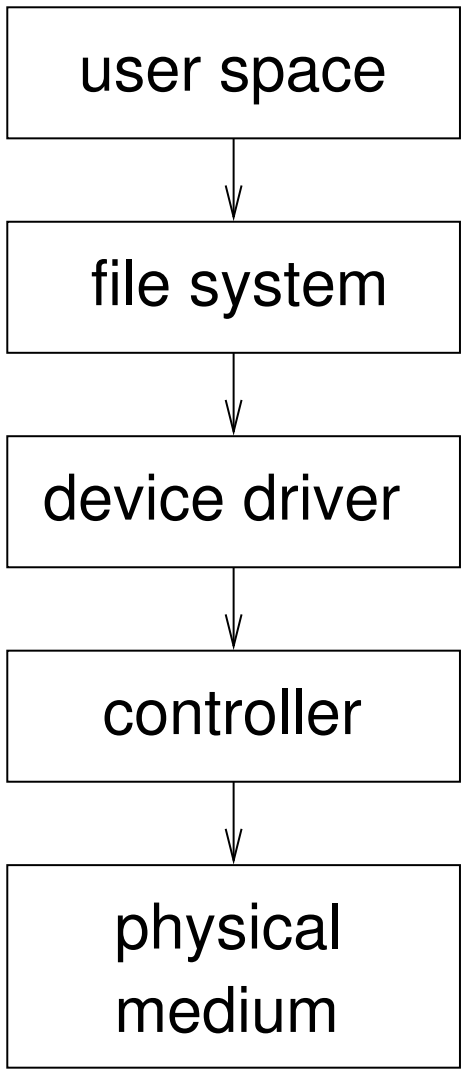


Interface and Layers



Interface and Layers





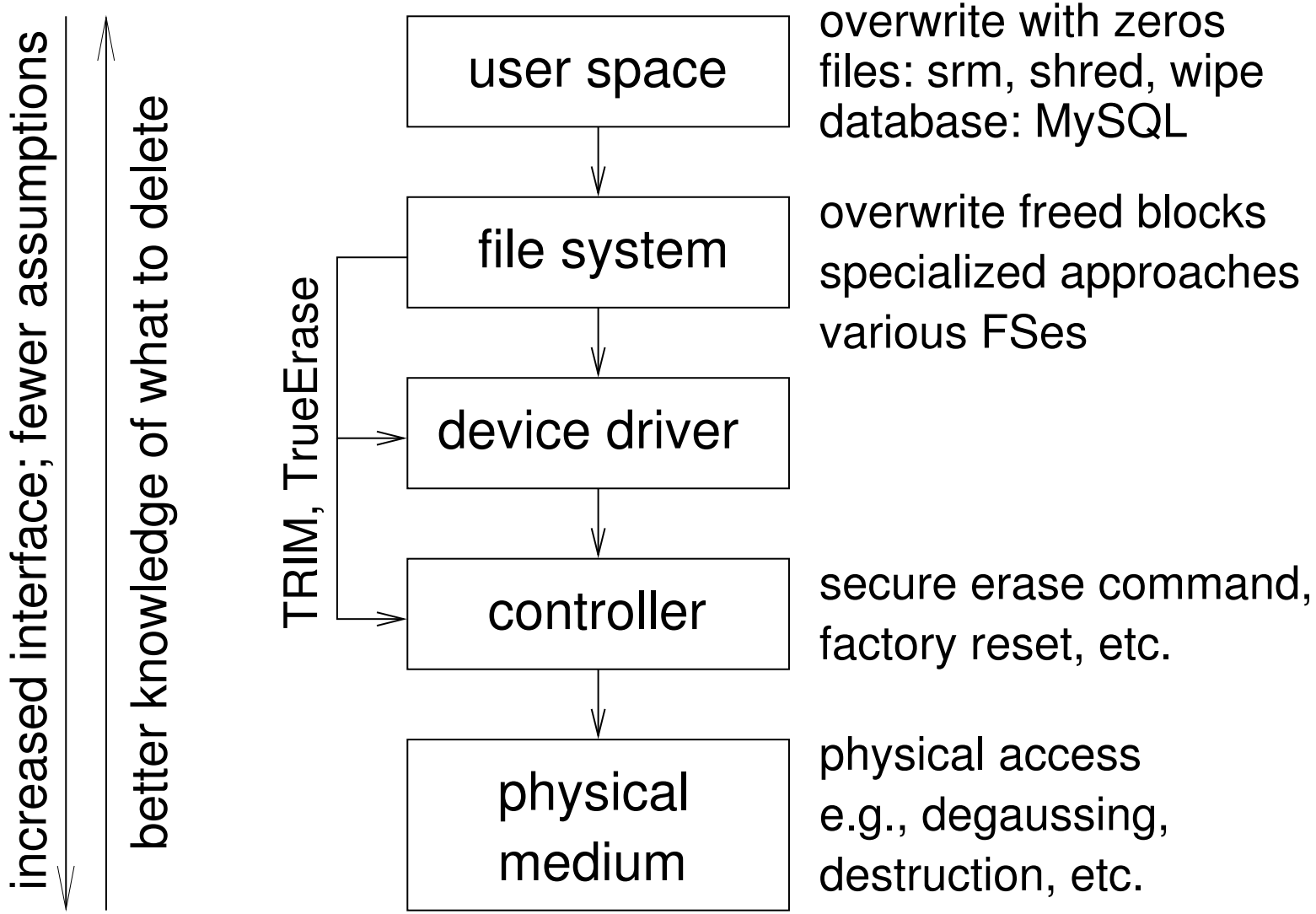
overwrite with zeros
files: srm, shred, wipe
database: MySQL

overwrite freed blocks
specialized approaches
various FSes

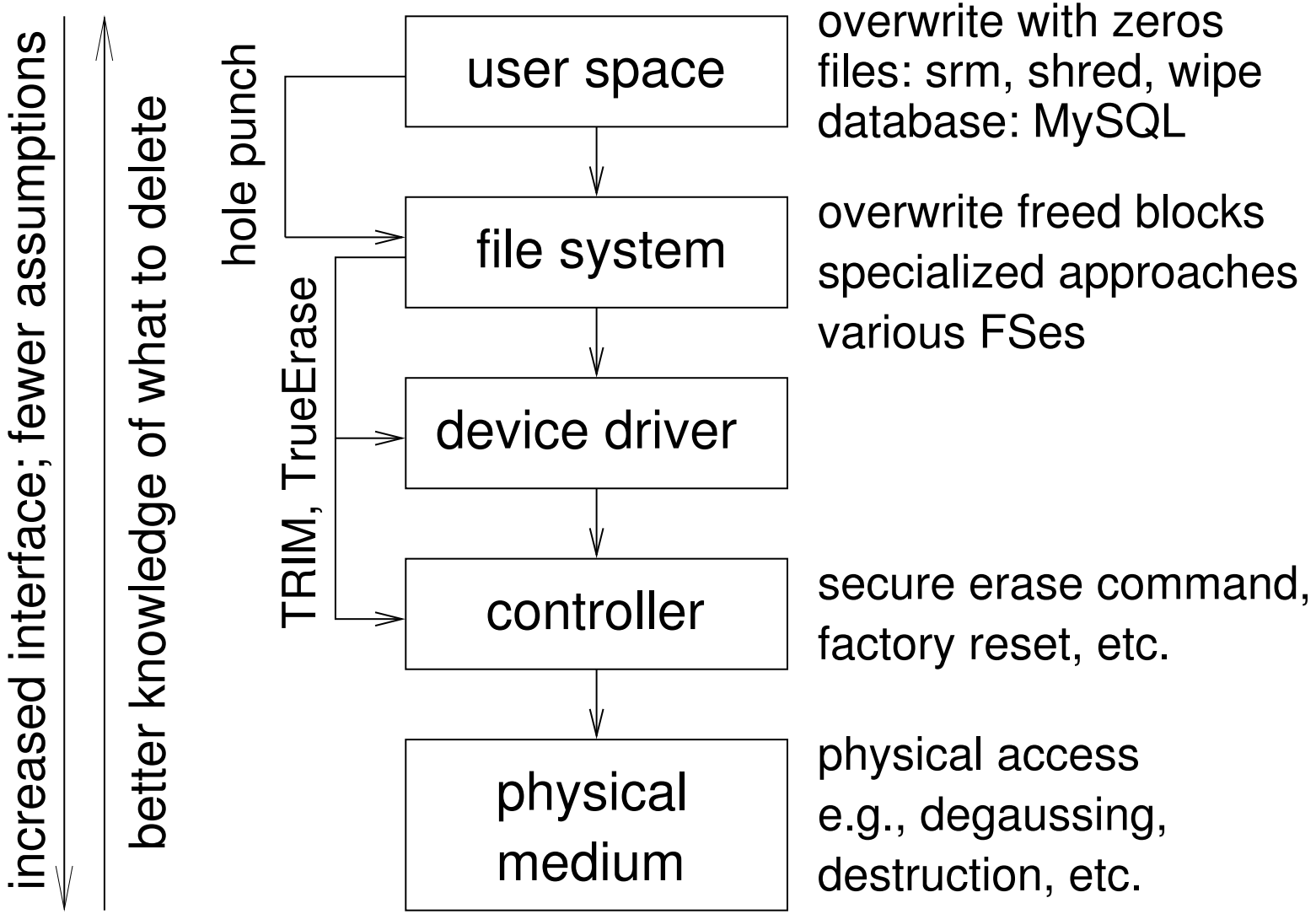
secure erase command,
factory reset, etc.

physical access
e.g., degaussing,
destruction, etc.

Interface and Layers



Interface and Layers



Solution Space

Solution Name	Target Adversary	Integration	Granularity	Scope	Lifetime
overwrite	unbounded coercive	user-level (in)	per-file	targeted	unchanged
fill	unbounded coercive	user-level	per-block	untargeted	unchanged
NIST clear	internal repurposing	varies	per-medium	untargeted	varies
NIST purge	external repurposing	varies	per-medium	untargeted	varies
NIST destroy	advanced forensic	physical	per-medium	untargeted	destoyed
ATA secure erase	external repurposing	controller	per-medium	untargeted	unchanged
flash SAFE	external repurposing	controller	per-medium	untargeted	some wear
renaming	unbounded coercive	kernel (in)	per-block	targeted	unchanged
ext2 sec del	unbounded coercive	kernel (in)	per-block	targeted	unchanged
ext3 basic	unbounded coercive	kernel (in)	per-block	targeted	unchanged
ext3 comprehensive	unbounded coercive	kernel (in)	per-block	targeted	unchanged
purgefs	unbounded coercive	kernel (in)	per-block	targeted	unchanged
ext3cow sec del	bounded coersive	kernel (in)	per-block	untargeted	unchanged
compaction	unbounded coercive	kernel	per-block	untargeted	some wear
batched compaction	unbounded coercive	kernel	per-block	untargeted	some wear
per-file encryption	bounded coersive	kernel	per-file	targeted	some wear
DNEFS	bounded peek-a-boo	kernel	per-block	untargeted	some wear
scrubbing	unbounded coercive	kernel (in)	per-block	untargeted	unchanged
ShredDroid	unbounded coercive	user-level	per-block	untargeted	some wear

Conclusions and Future Work

- Secure deletion solutions may differ in many ways
  - e.g., adversaries, behaviour properties, interface assumptions
  - by systematizing these spaces, we can better evaluate when one solution properly achieves secure deletion in a concrete setting, and better compare among other suitable solutions
- Ideal solutions would have minimal assumptions along with a small granularity
  - techniques to move information on deleted data from the user's intentions to the lowest layers are therefore useful
- Other storage media types and interfaces exist
  - cloud storage, mixed-media data centers, etc.

Now, we'll look at two general user-level solutions: overwriting and filling.

User-Level Solutions

- overwriting
  - a targeted-scope per-file-granularity solution
  - opens specific files and overwrites them with zeros
  - assumes that in-place updates occur
  - e.g., srm, shred, wipe
- filling
  - an untargeted-scope per-block-granularity solution
  - opens a new file and writes into until the file system is full
  - assumes that when the file system is full, no deleted data remains
  - e.g., scrub, Mac Disk Utility

User-Level Secure Deletion

	File Data		File Metadata	
File System	**Updates In-place**	**Filling Works**	**Updates In-place**	**Filling Works**
btrfs	no	yes	no	no
exfat	yes	no	no	no
ext2	yes	yes	no	no
ext3/ext4	no	yes	no	no
f2fs	no	yes	no	yes
hfs	yes	yes	no	no
hfsplus	no	yes	no	no
jffs2	no	no	no	yes
reiserfs	yes	yes	no	no
ubifs	no	yes	no	yes
vfat	yes	no	no	no
yaffs	no	yes	no	yes