Traceless Execution Support for Privacy Enhancing Technologies

Daniela Lopes
Advisor: Prof. Nuno Santos

24 November 2021

Presentation outline

1. Introduction and motivation
2. Related work
3. Design
4. Evaluation
5. Conclusions

Why people need to use PETs

Privacy Enhancing Technologies (PETs):

Secure Messaging

VPNs

Anonymizing networks

Anti tracking

People use PETs to overcome several challenges:

Censorship

Lack of privacy

Surveillance

However, people are afraid to use PETs

Drawback of using PETs:

- Large amount of local resources
- Non-trivial meta-data and cryptographic artifacts

➡ **Extensive digital footprint**

Why existing PETs are not enough to protect users:

- Forensic inspection of the devices of endangered users such as journalists, whistleblowers and travelers passing border controls
- PETs do not provide plausible deniability

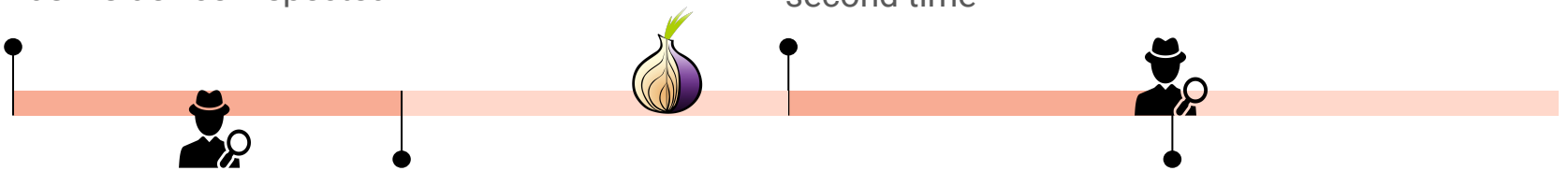
Example of a threatening usage scenario

John passes border control to enter the repressive country and has his device inspected

➡ **Snapshot 1**

John passes border control to leave the repressive country and has his device inspected a second time

➡ **Snapshot 2**



John makes publications about his sensitive findings via the Tor browser on Wikileaks

The investigator compares **Snapshot 1** and **Snapshot 2** and **observes** that John was the author of the publications

Our goal: plausible deniable PET sessions

John passes border control to enter the repressive country and has his device inspected

➡ **Snapshot 1**

John initiates a **plausible deniable PET session**

John makes publications about his sensitive findings via the Tor browser on Wikileaks

John closes the **plausible deniable PET session**

John passes border control to leave the repressive country and has his device inspected a second time

➡ **Snapshot 2**

The investigator compares **Snapshot 1** and **Snapshot 2** and **cannot observe** changes that lead to suspecting John

Plausible deniability: ability to deny knowledge or responsibility of having installed and executed PETs on their device

Challenges of enforcing deniable PET sessions

Challenges:

1. Where is the PET data going to be stored?
2. How to make that data non observable?
3. How to maintain the PET functionality?

Contributions

Design: **Calypso**, a steganographic storage system

- Offer the abstraction of a hidden **shadow partition**, to be used as a regular partition
- The shadow partition parasites on the free blocks of the native system to conceal data
- Preserves the initial entropy characteristics of the disk

Implementation: Fully functional prototype as a Linux kernel module for version 5.4

Evaluation: Extensive evaluation of the prototype

- Functionality
- Isolation
- Security
- Performance

Related work

Private program execution

👍 Processes are isolated from the rest of the system, preventing leaks

👎 Not deniable by design, the system itself is not concealed

Deniable file systems

👍 Data hiding in persistent storage

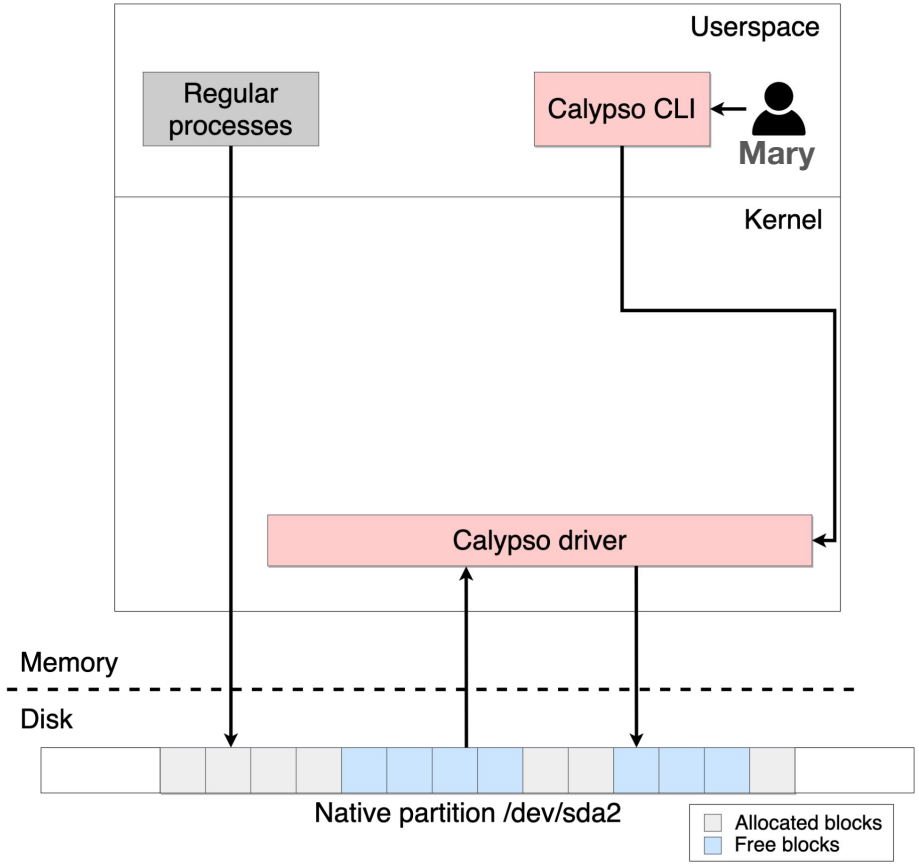
👎 Writing random patterns abnormally increases the disk's entropy, disclosing the existence of hidden data

Defending from multi-snapshot attacks

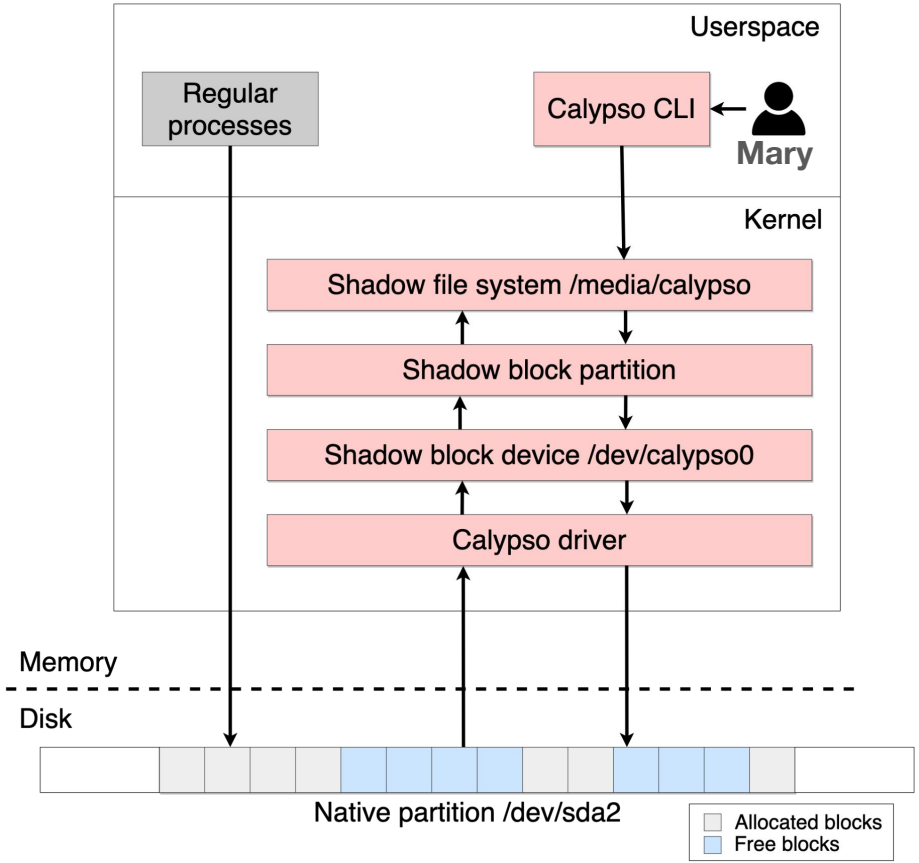
👍 Focus on preventing multi-snapshot attack, thereby providing stronger deniability

👎 Performance penalties and unnatural disk write-patterns

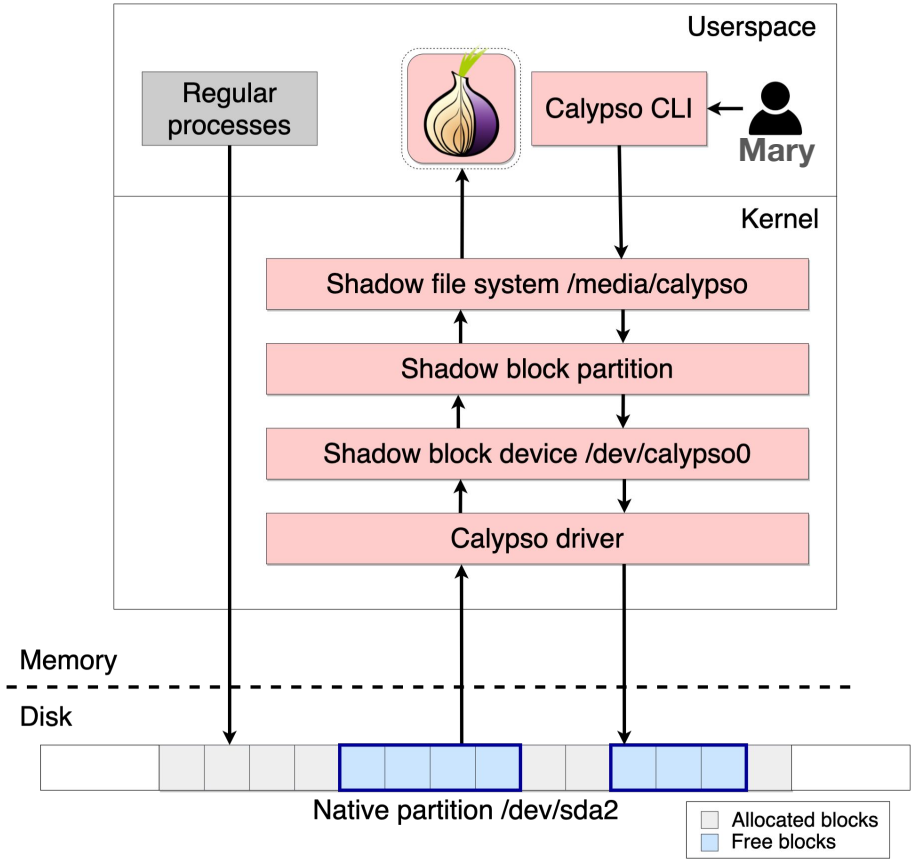
Calypso's architecture: inserting the module



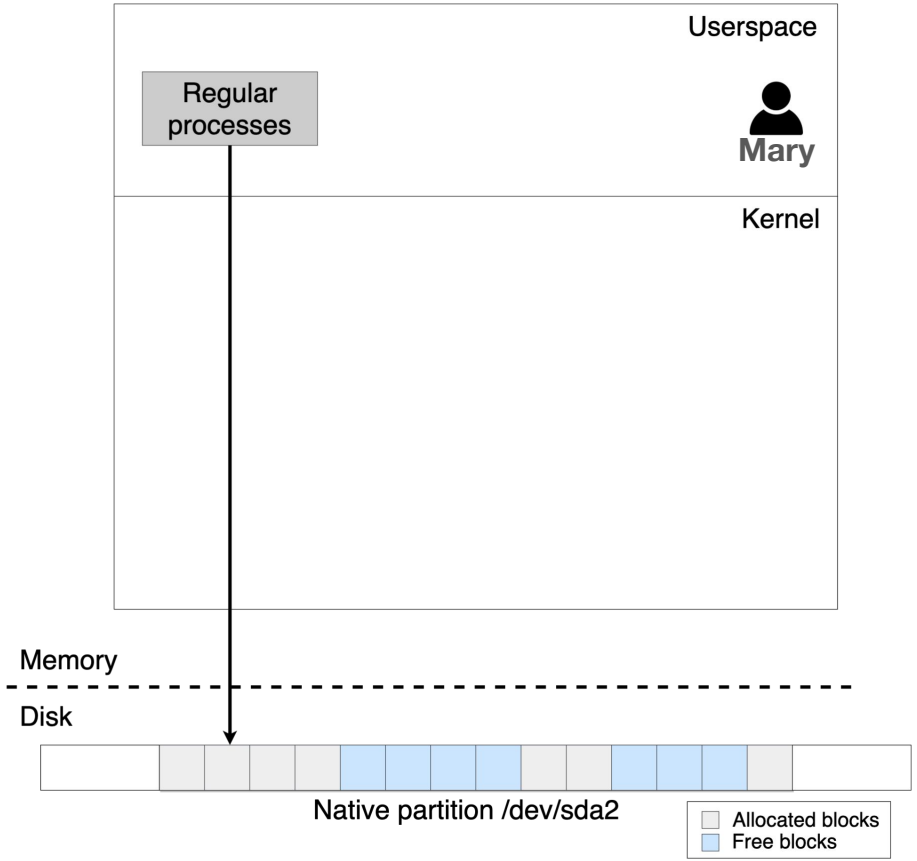
Calypso's architecture: shadow partition



Calypso's architecture: executing PETs



Calypso's architecture: state after Calypso



Calypso: design obstacles

Traceless bootstrap

- Prevent or erase traces resulted from executing Calypso and inserting its module

Block allocation

- Find usable blocks without disrupting the native system or causing corruption to Calypso

Monitoring native changes to blocks

- Track free blocks and prevent overwrites to Calypso data

Deniably encoding data blocks

- Ensure Calypso only makes deniable changes to native blocks

Calypso: design solutions

Traceless bootstrap

- Erase all persistent traces of bootstrap using rootkit techniques

Block allocation

- Use the free blocks of the native file system, mapped with in-memory data structures

Monitoring native changes to blocks

- Intercept all write requests to the native system to check and update the allocation in-memory data structures accordingly

Deniably encoding data blocks

- Choose only blocks with initial entropy above an established threshold

Evaluation goals

Functionality: Can Calypso support the execution of multiple PETs without disrupting the system or disturbing the functionality of PETs?

Isolation: Does the native file system present detectable changes after executing Calypso?

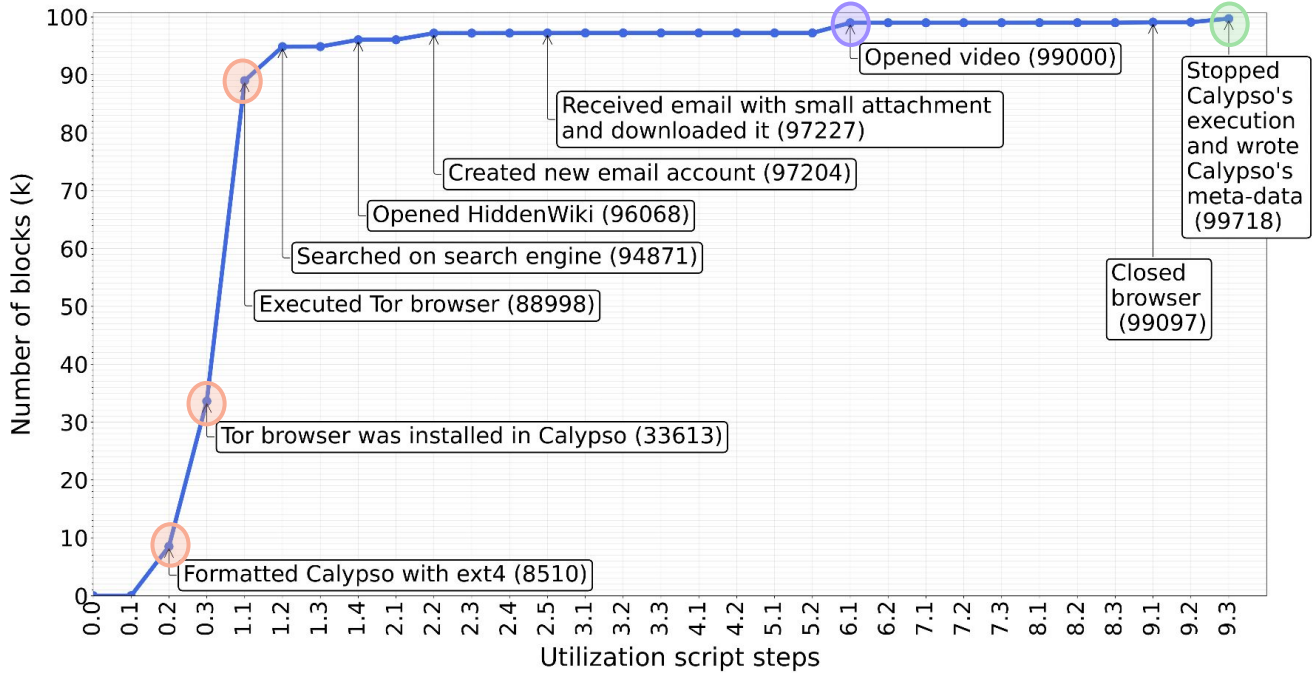
Security: Does Calypso make only deniable changes that preserve the initial entropy of the disk while ensuring practical storage capacity?

Performance: Is Calypso performant enough to maintain the usability of PETs and the native system?

Functionality evaluation

- Test Calypso's ability to sustain the execution of PET applications without file system errors
- Monitor the execution of PETs with multiple disk workloads, throughout the execution of multiple utilization steps

Functionality results

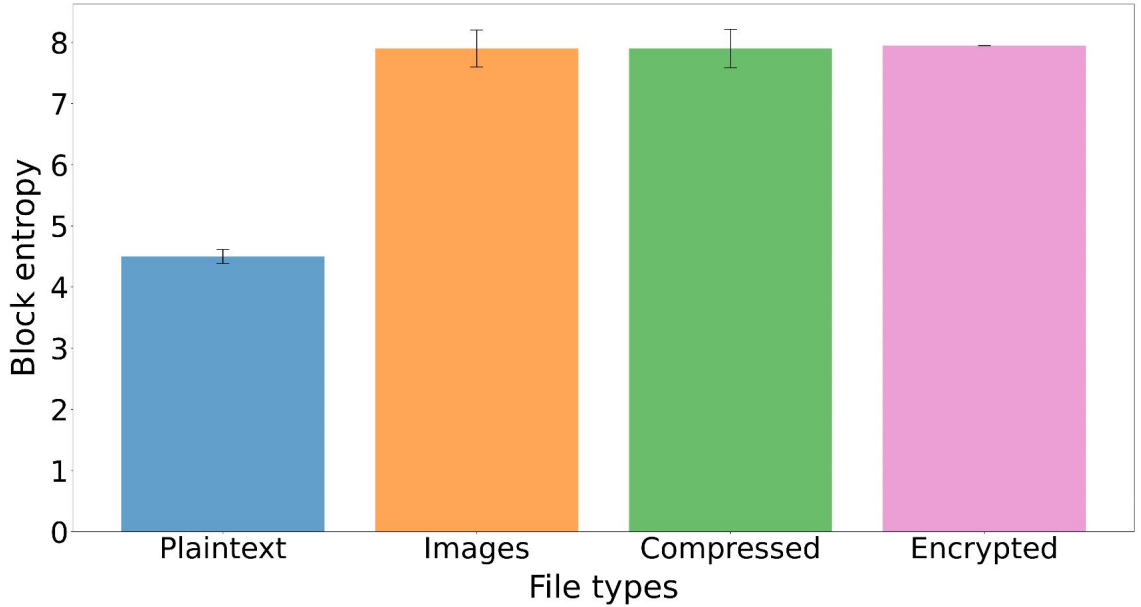


Calypso is able to sustain the execution of PETs through multiple utilization steps, without disrupting the system or the PET

Security evaluation

- Assess Calypso’s capability to make only non-observable changes to the native file system
- Maintain a practical storage capacity to execute PETs
- During a physical inspection, the inspector should not be able to distinguish between free blocks that store Calypso’s data and meta-data, from blocks that do not

Security results: entropy of common files



Several files erased in our computers can result in usable blocks for Calypso, without changing the initial entropy characteristics

Security

Entropy threshold: all blocks with entropy above this value all eligible to be used

Entropy difference: quantifies how much the entropy of a block has changed

Entropy threshold	Usable blocks (%)	Storage capacity (GB)	Maximum entropy difference
0	100%	4.00	1.000
5	74.91%	3.00	0.372
7	74.67%	2.99	0.120

Calypso can produce only deniable changes with higher entropy values while preserving sufficient storage capacity

Conclusions

- State-of-the-art PETs are vulnerable to forensic analysis
- Calypso makes non-observable changes to disk
- Give protection against forensic analysis

Future work:

- Enhance isolation by leveraging containerization techniques
- Explore the implications of purposely executing plausible actions as a decoy to justify the changes to disk and to increase storage capacity

Q&A

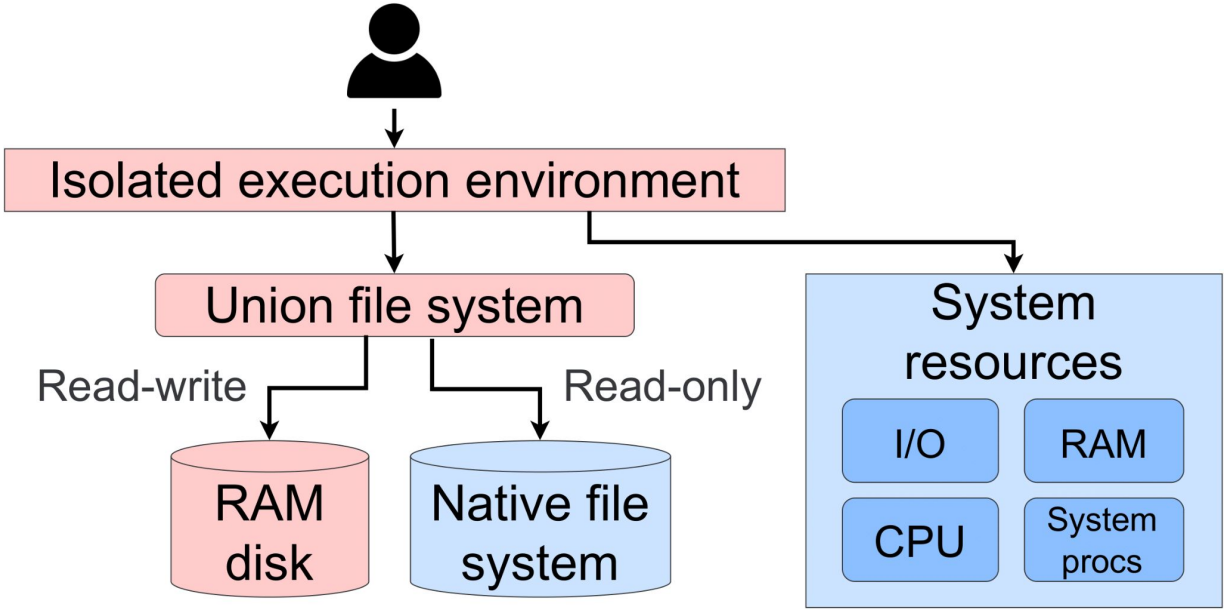
Extra slides

Main findings

Functionality: Calypso can sustain the execution of PETs without disrupting the native system or being corrupted

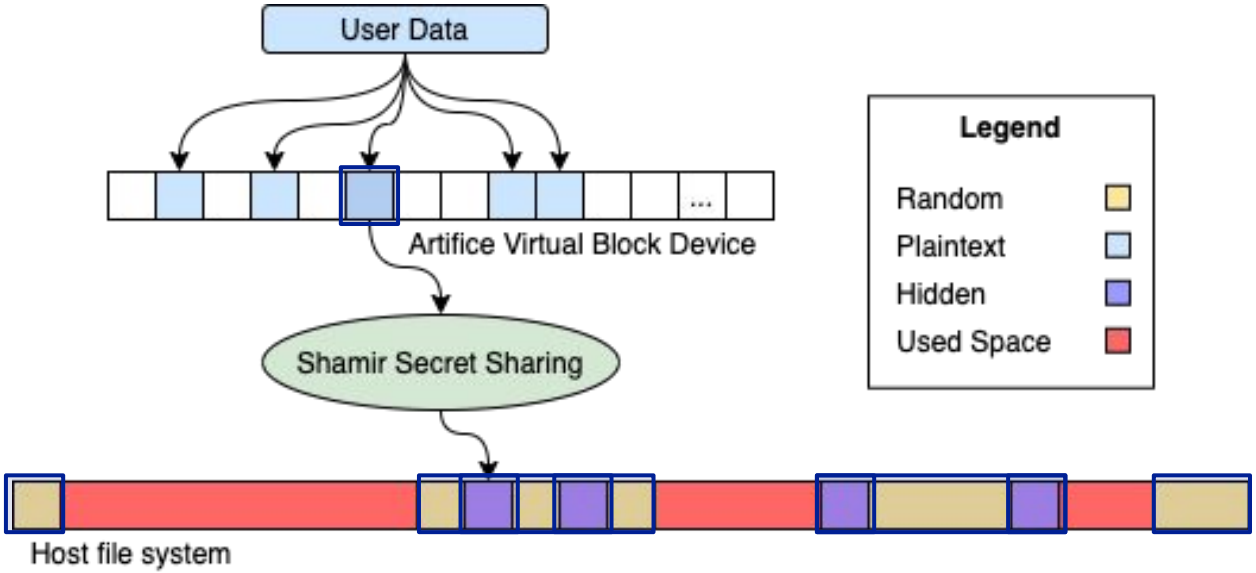
Security: Calypso can take advantage of several native blocks generated by the erasure of common file types to make only deniable changes, while preserving sufficient storage capacity

Residue-free computing



- No persistence or plausible deniability
- Lack of application support
- Changes the file system

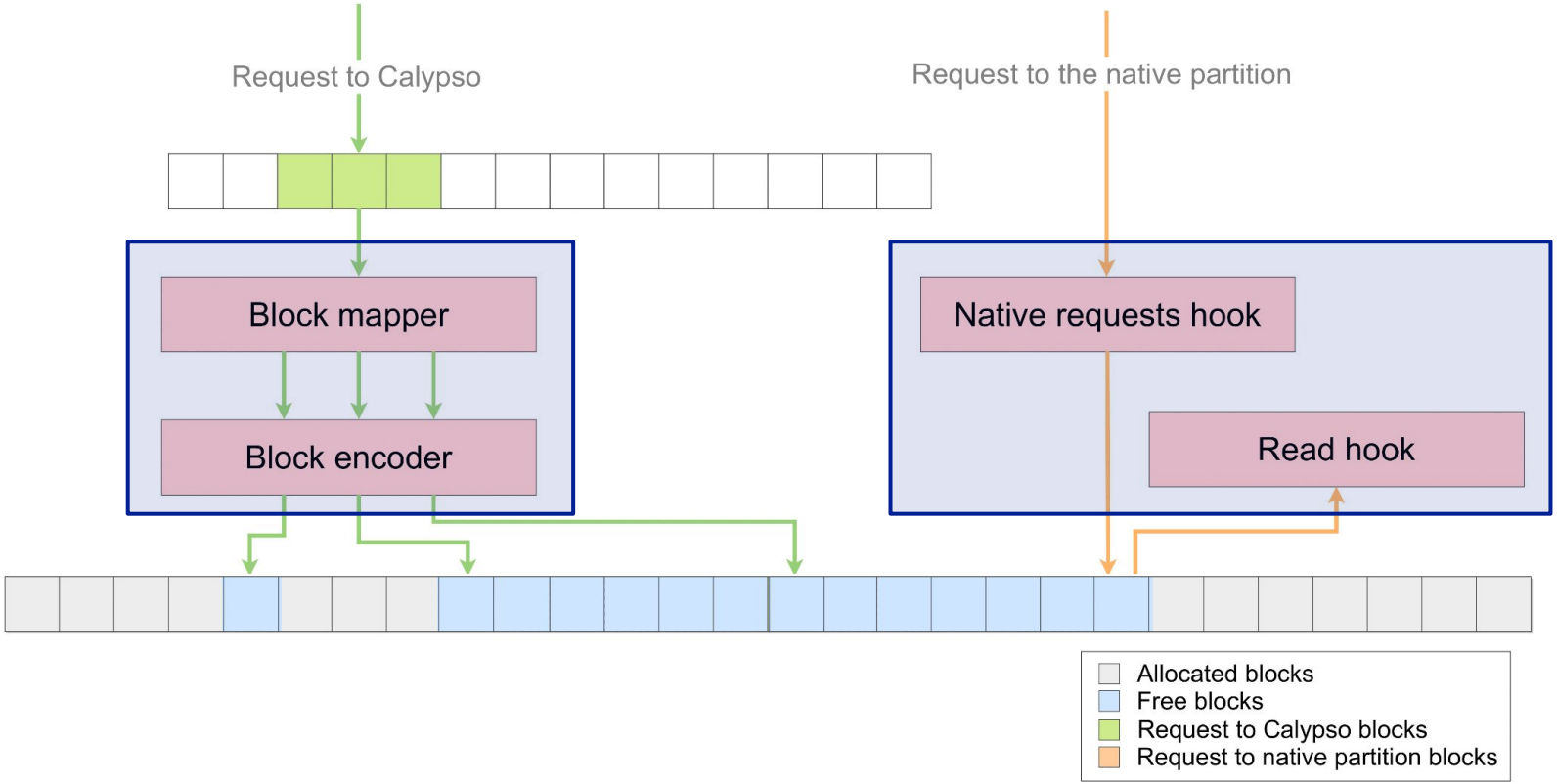
Artifice



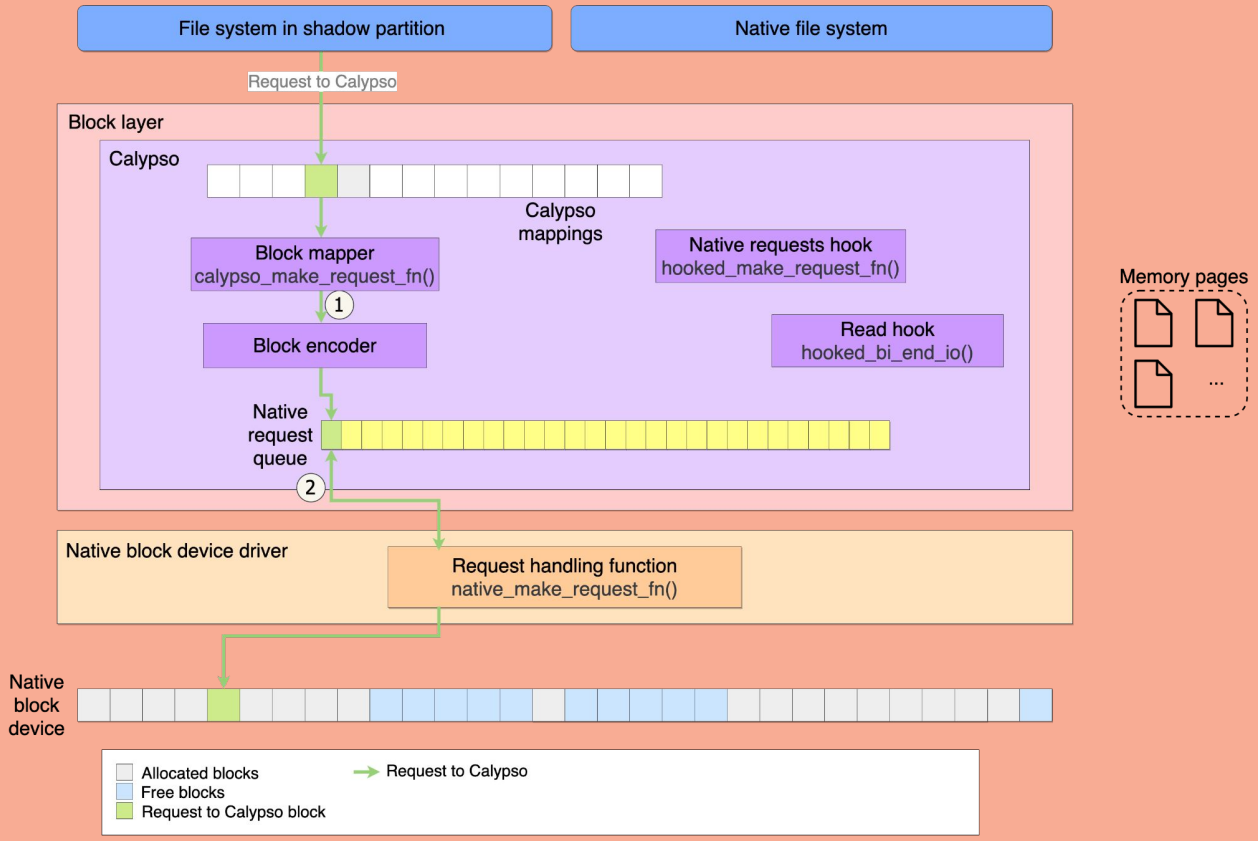
Pseudo-random blocks change the bit entropy of the disk

Dependency on external hardware components

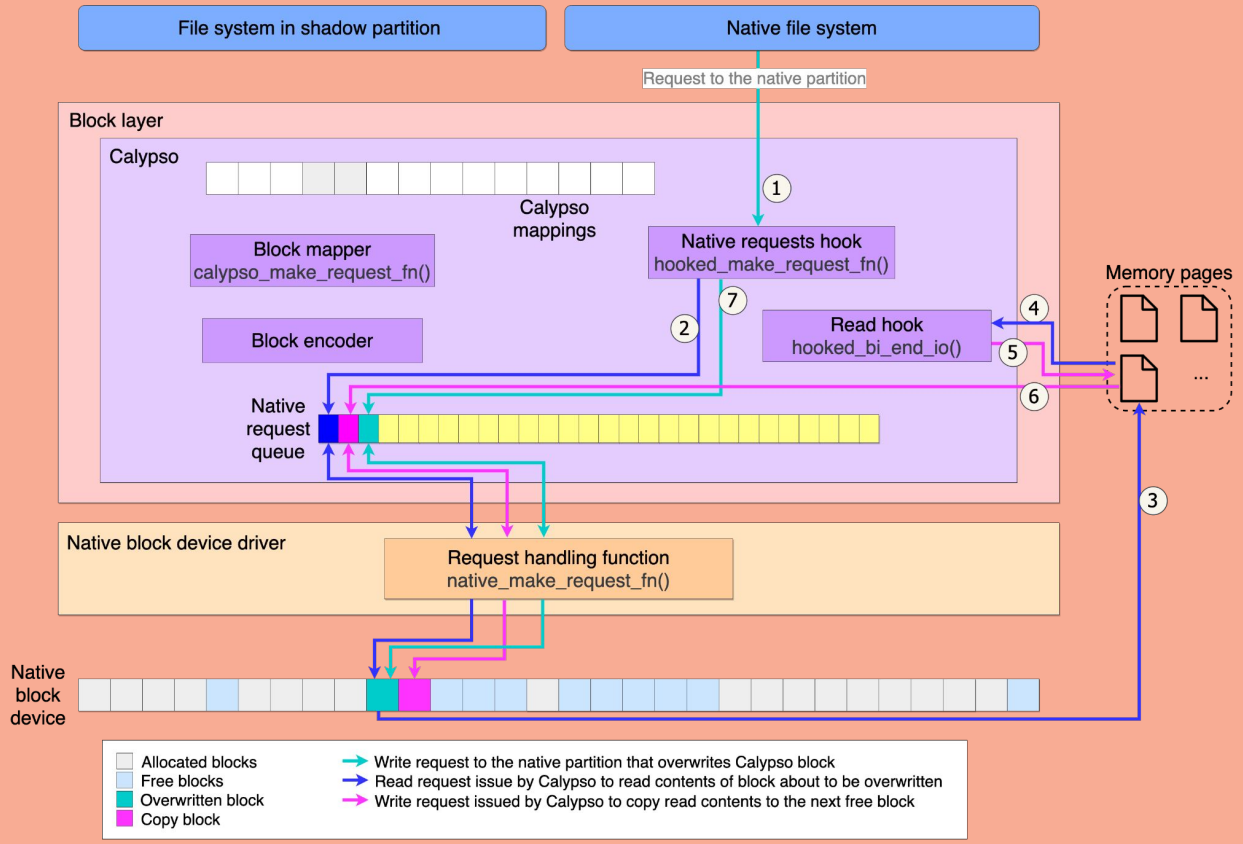
Calypso: driver



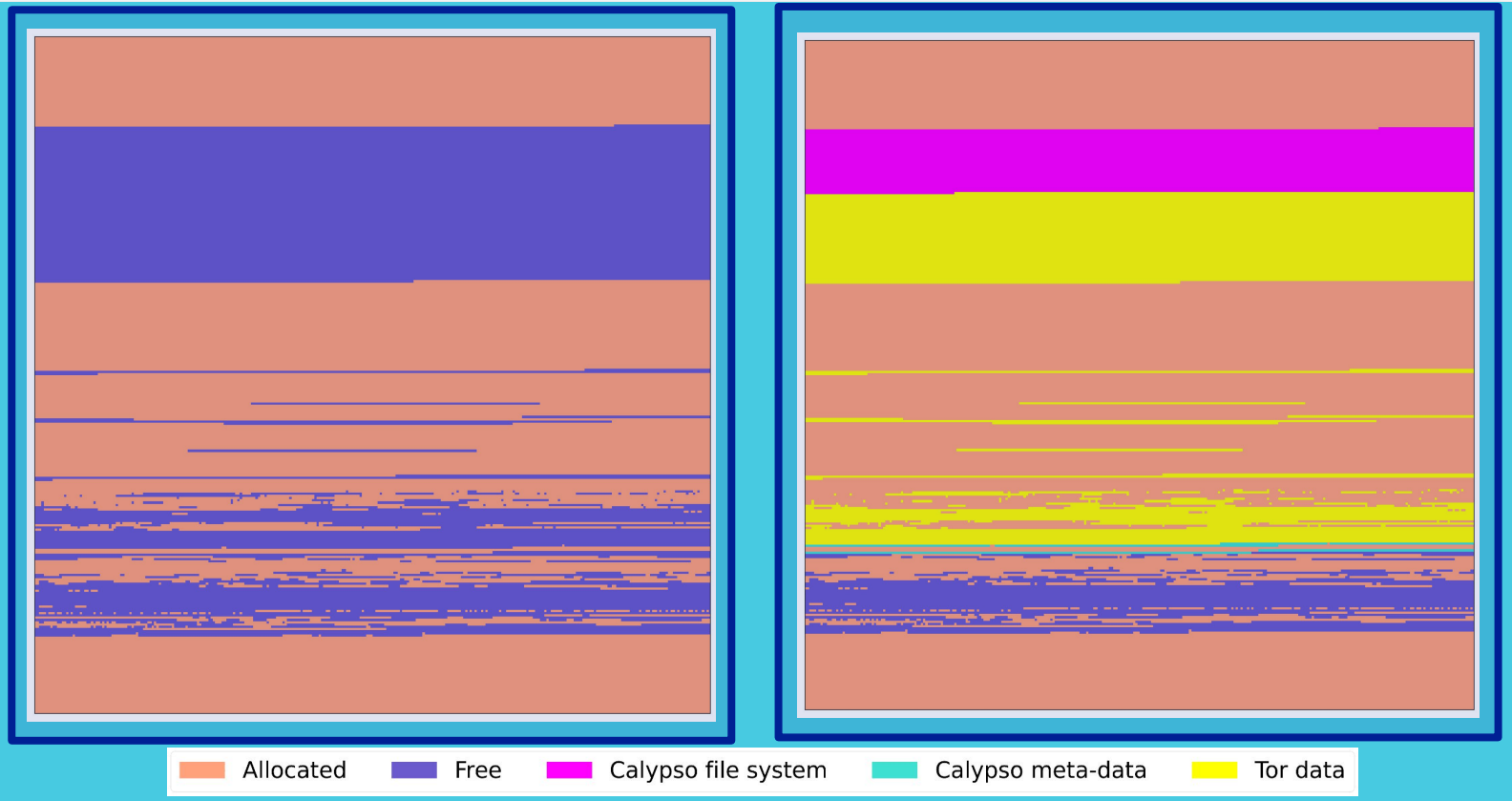
Redirecting requests



Intercepting requests and handling overwrites



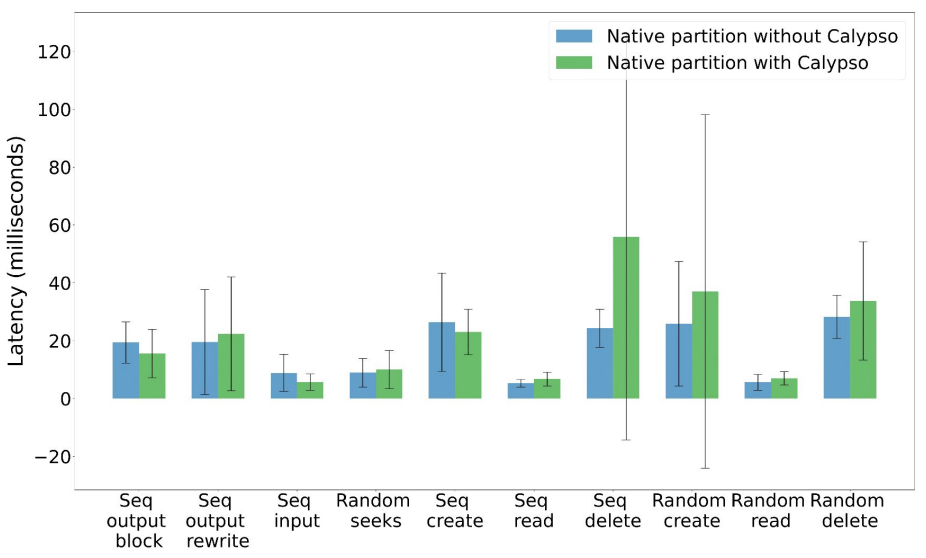
Functionality

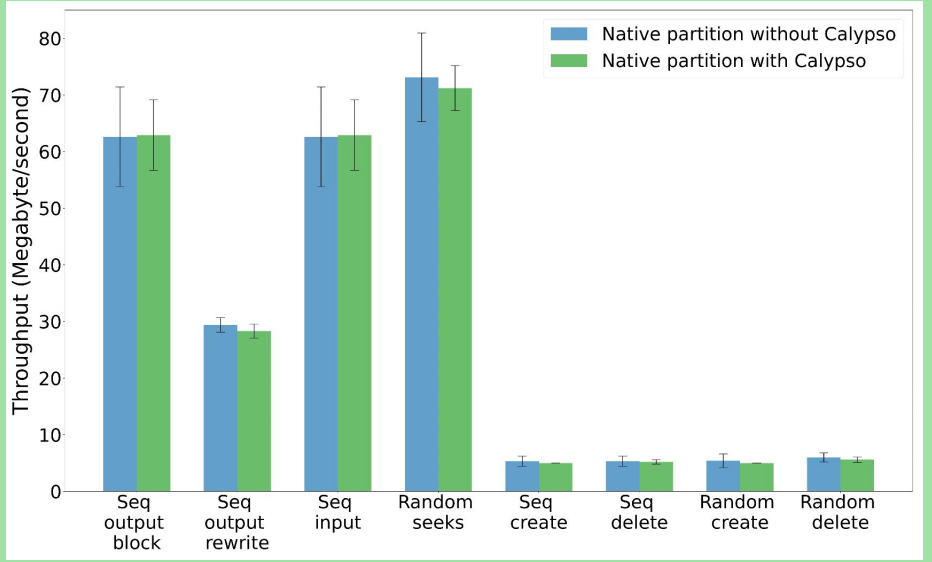


Security

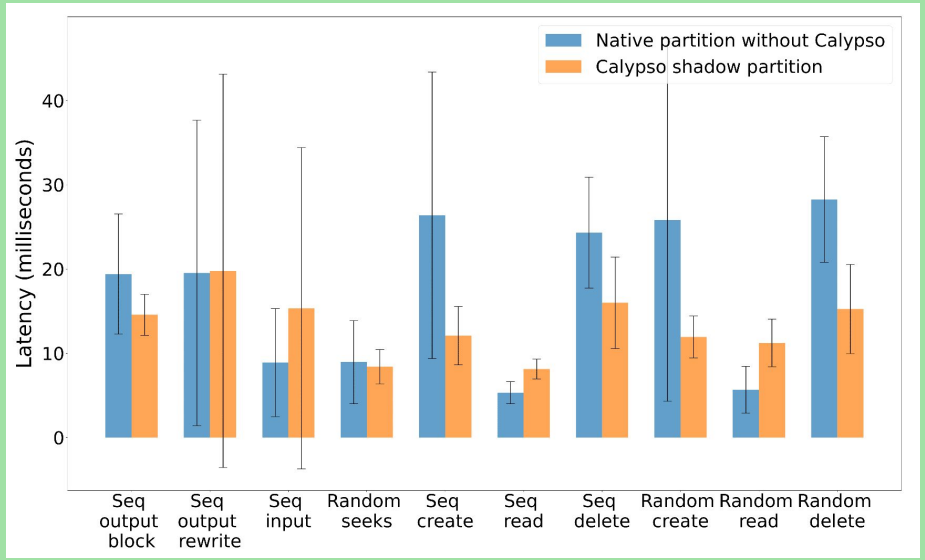
Threshold	Usable blocks count	Usable blocks (%)	Storage capacity (GB)	Average differential	Minimum differential	Maximum differential
0	1048576	100.00%	4.00	0.11225	0.00000	1.00000
1	1048291	99.97%	4.00	0.11198	0.00000	0.87262
2	1048141	99.96%	4.00	0.11187	0.00000	0.74745
3	1047967	99.94%	4.00	0.11175	0.00000	0.62262
4	1047261	99.87%	3.99	0.11139	0.00000	0.49616
5	785484	74.91%	3.00	0.00306	0.00000	0.37172
6	784098	74.78%	2.99	0.00262	0.00000	0.24590
7	782983	74.67%	2.99	0.00244	0.00000	0.12039

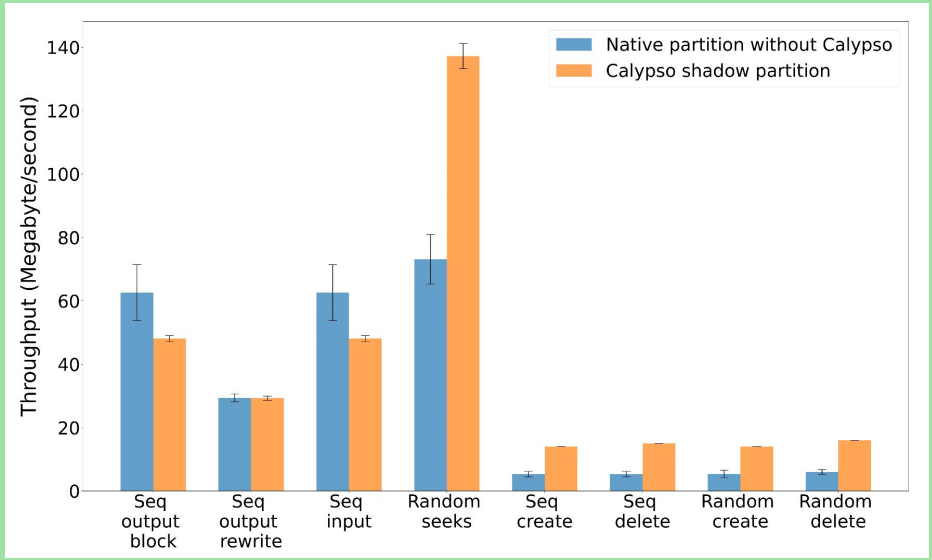
Performance



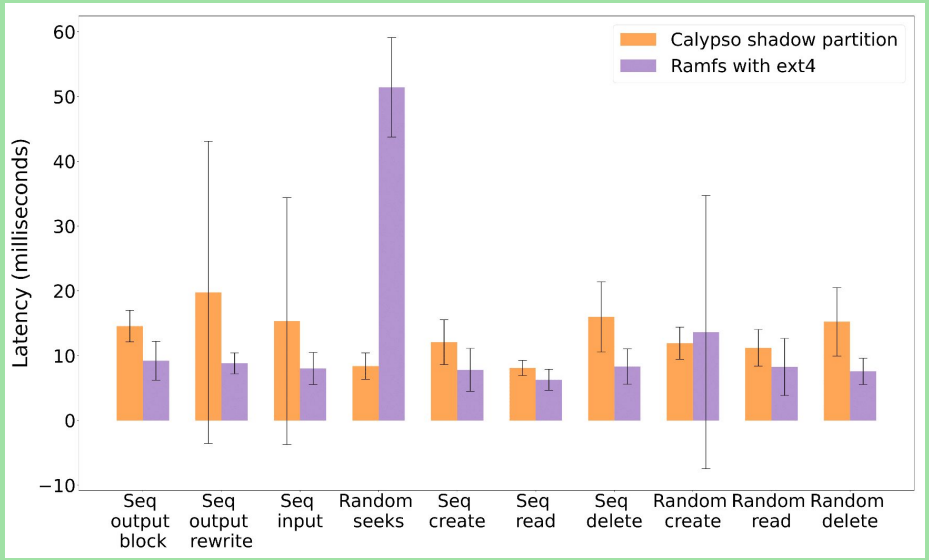


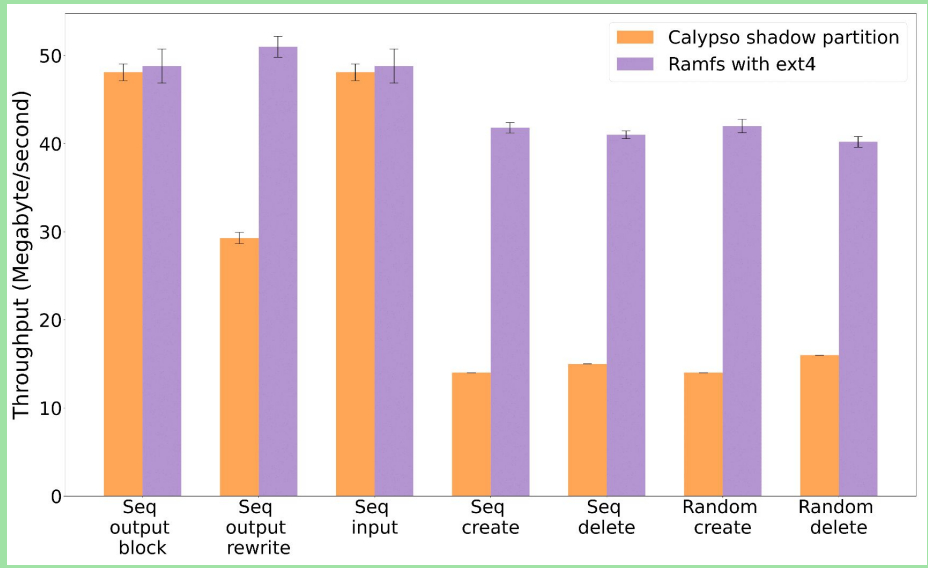
Performance





Performance





Fix slides

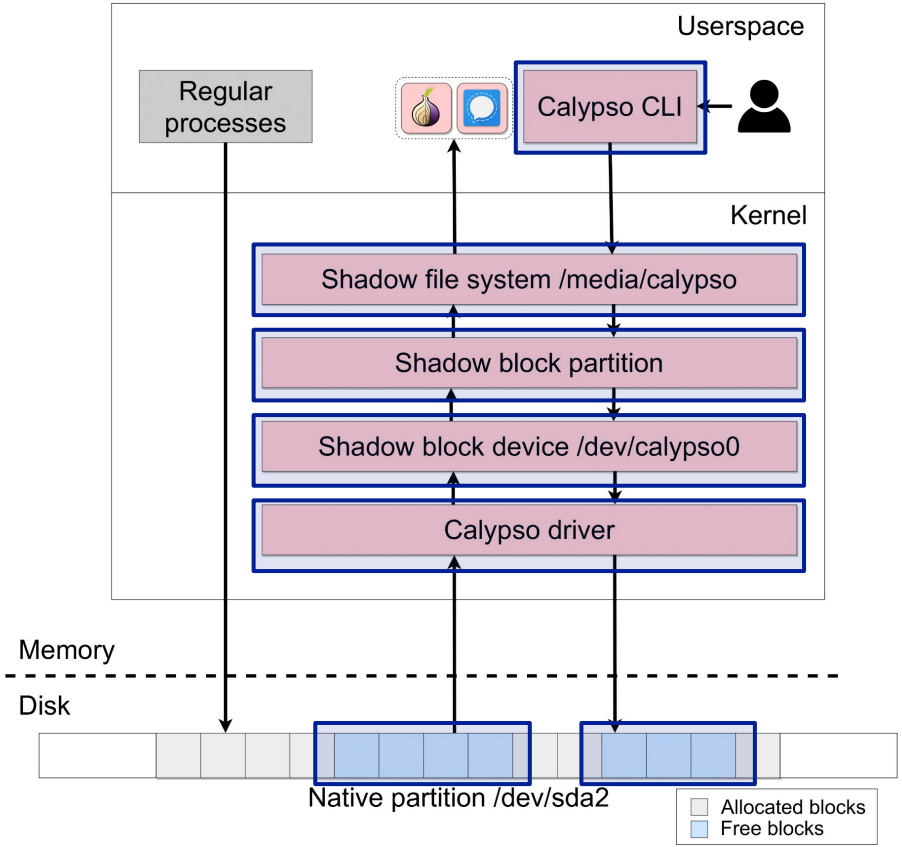
Contents

1. Introduction and motivation
2. **Related work**
3. Design
4. Evaluation
5. Conclusions

Contents

1. Introduction and motivation
2. Related work
3. **Design**
4. Evaluation
5. Conclusions

Calypso: architecture



Contents

1. Introduction and motivation
2. Related work
3. Design
4. **Evaluation**
5. Conclusions

Calypso: design obstacles

Traceless bootstrap

- Encode all actions required to make Calypso available to the user
- Prevent or erases traces from executing Calypso
- Single point of failure that can compromise the system's deniability

Block allocation

- Find usable blocks without disrupting the native system or causing corruption to Calypso

Monitoring native changes to blocks

- Prevent overwrites to Calypso data
- Keep track of the native allocation of free blocks that stop being usable

Deniably encoding data blocks

- Ensure Calypso only makes deniable changes to native blocks
- Encrypted data has particularities such as high entropy. Increasing the disk entropy discloses the existence of hidden data

Calypso: design solutions

Traceless bootstrap

- Hide bootstrap files in unpredictable locations in the disk, only recoverable by the rightful user
- Erase all persistent traces of bootstrap using rootkit techniques

Block allocation

- Use the free blocks of the native file system
- Map Calypso allocated blocks with in-memory data structures:
- Keep the free or allocated status for every native block

Monitoring native changes to blocks

- Intercept all write requests to the native system to check and update the allocation in-memory data structures accordingly

Deniably encoding data blocks

- Choose only blocks with initial entropy above an established threshold

Contents

Calypso: implementation obstacles

Retrieving usable blocks for storage

- Technically challenging and file system dependant because the Linux Virtual File System does not provide an interface to retrieve the free blocks

Adapting read and write requests functionality

- **Redirecting:** passing adapted requests to the native blocks
- **Intercepting:** monitoring and preventing changes to blocks mapped by Calypso, as well as keeping track of free blocks that become allocated

Cryptography and data hiding

- Prevent adversaries from correctly retrieving Calypso's data or meta-data
- **Hiding data blocks:** generating an encryption key deterministically only with the right user's secret
- **Storing and retrieving meta-data:** finding its location without in-memory structures

Calypso: implementation solutions

Retrieve usable blocks for storage

- Iterate through the Ext4 native file system meta-data structures to retrieve all the block bitmaps into a single one.

Adapting read and write requests functionality

- **Redirecting:** adapting the received requests to map the corresponding native blocks and passing them directly to the native request queue
- **Intercepting:** hooking the native block request handling function

Cryptography and data hiding

- **Hiding data blocks:** HMAC-based extract and expand key derivation function to generate an encryption key depending on the user's password
- **Storing and retrieving meta-data:** passing a seed to a pseudo-random number generator to obtain the same sequence of numbers for a user password

Related work

Private program execution

Isolate and conceal all data generated by a program, without providing the user with plausible deniability

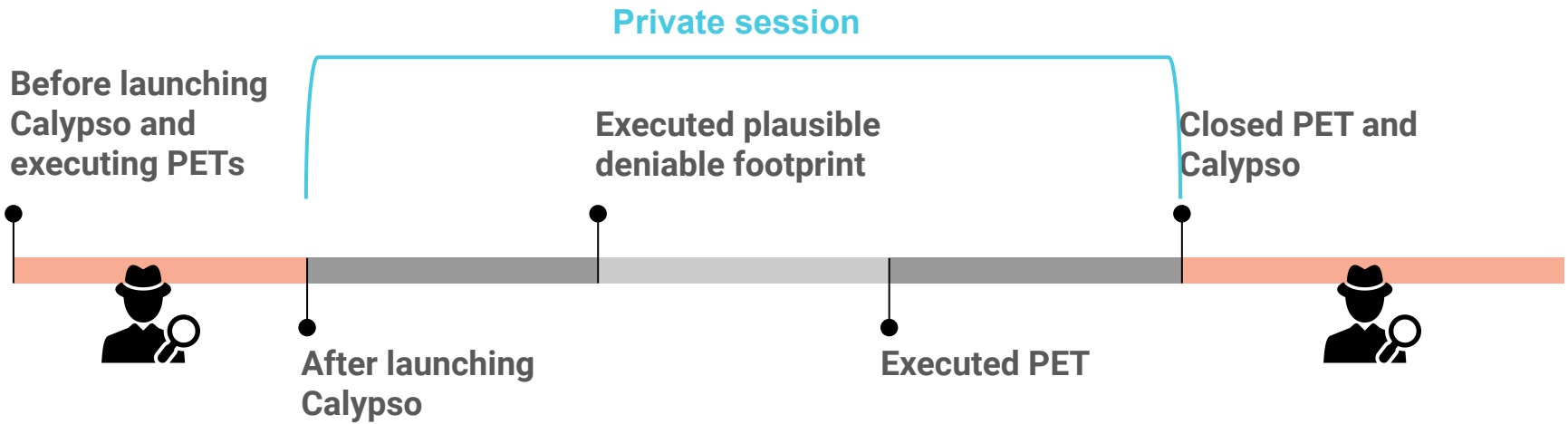
Deniable file systems

Deniable data hiding on persistent storage and single-snapshot attack prevention

Defending from multi-snapshot attacks

Multi-snapshot attack prevention by hiding modifications that indicate the existence of hidden data

Threat model



Isolation + Non-observability → Plausible deniability

Our goal: plausible deniable PET sessions

Plausible deniability: ability to deny knowledge or responsibility of having installed and executed PETs on their device

Conclusions

- State-of-the-art PETs are vulnerable to forensic analysis
- Calypso makes non-observable changes to disk
- Give protection against forensic analysis

Why people need to use PETs

Challenges that Internet users face:

Censorship

Lack of privacy

Surveillance

Privacy Enhancing Technologies (PETs):

Secure Messaging

VPNs

Anonymizing networks

Anti tracking

Traceless Execution Support for Privacy Enhancing Technologies

Student: Daniela Lopes
Advisor: Nuno Santos
Chairperson:
Member of the committee:
Instituto Superior Técnico

Many people are afraid of using PETs

Forensic inspection of devices + **Current PETs are not designed to remain unobservable during forensic investigations**

- Journalists
- Whistle blowers
- Travelers crossing border control

Many people fear the consequences of using PETs

Using PETs and their implications

Censorship

Lack of privacy

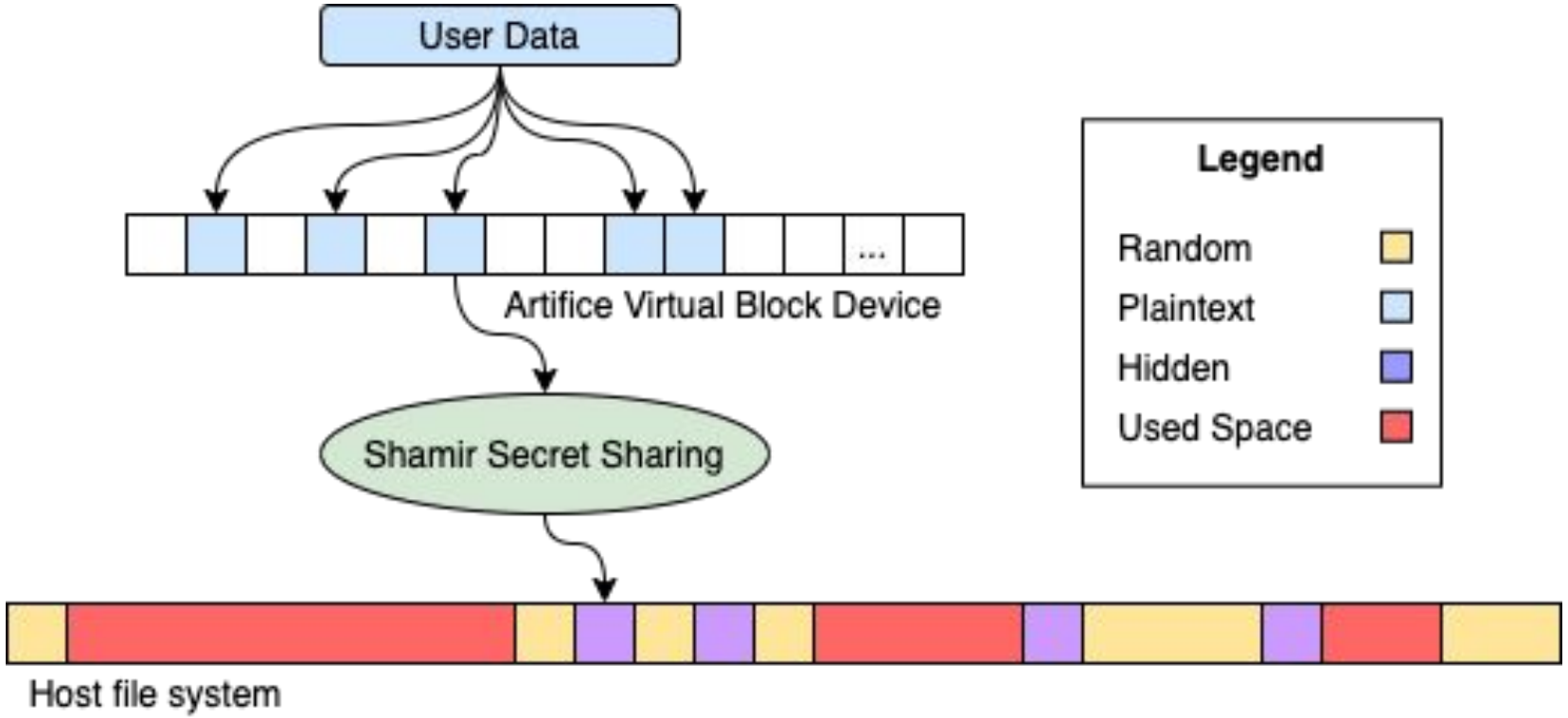
Surveillance

Lack of usability + Greater usage local resources + Meta-data and cryptographic artifacts

- Journalists
- Whistle blowers
- Border controls

Forensic inspection of devices + PETs do not provide plausible deniability ▶ **Many people fear the consequences of using PETs**

Artifice: encoding user data



Artifice: overview

Protect users in life-threatening situations

+

Provide a deniable way to store sensitive data

Adversary is unable to distinguish blocks encoding hidden data from blocks with random data

Artifice is not deniable

- Pseudo-random blocks change the bit entropy of the disk
- Dependency on external hardware components

Artifice is not deniable!

Proposed solution: Calypso

Shadow drive

- Secondary file system that uses the unused blocks of the native file system

+

Shadow Containers

- Isolated environment to execute programs deniably

No observable changes are made to the persistent state

Calypso: design obstacles

Traceless bootstrap

Block allocation

- Finding and maximizing usable space
- Encoding data within the blocks while maintaining the entropy of the disk
- Fault tolerance and data loss

Monitoring native changes to blocks

-

Conclusions

State-of-the-art PETs are vulnerable to forensic analysis

Calypso offers isolation and maintains the entropy of the disk

Provide a traceless execution environment to increase PET usage by censored users

Give protection against forensic analysis

Using PETs and their implications

Censorship

Lack of privacy

Surveillance

Forensic inspection of devices

- Journalists
- Whistle blowers
- Travelers passing border controls

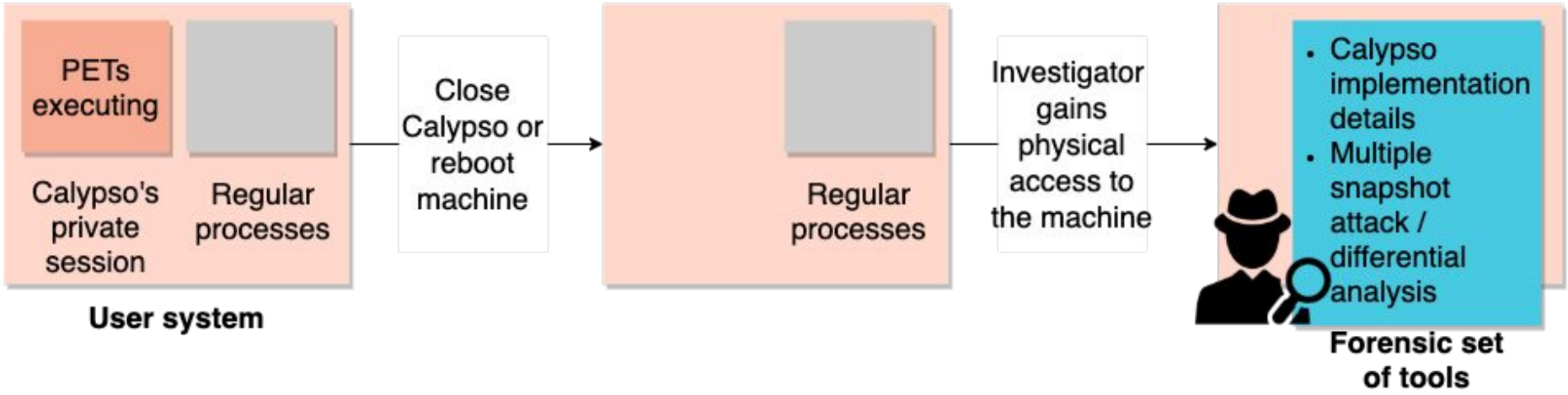
+

PETs do not provide plausible deniability

▶

Many people fear the consequences of using PETs!

Threat model



Using PETs and their implications

Censorship

Lack of privacy

Surveillance

Forensic inspection of devices

- Journalists
- Whistle blowers
- Travelers passing border controls

+

PETs do not provide plausible deniability

▶

Many people fear the consequences of using PETs!

Goals

Challenges	Solutions
Environment to execute PETs deniably with persistent state	Parasite on the free blocks of the native system
Maintain a PET's functionality and performance	Preserve the initial entropy characteristics of the disk

Retrieving usable blocks for storage

Cryptography and data hiding

Contents

Why people are afraid of using PETs

- Lack of usability
- Require large amounts of local resources
- Leaves an extensive footprint

Forensic inspection of devices

Why people are afraid of using PETs

Forensic inspection of devices

- Journalists
- Whistle blowers
- Travelers passing border controls

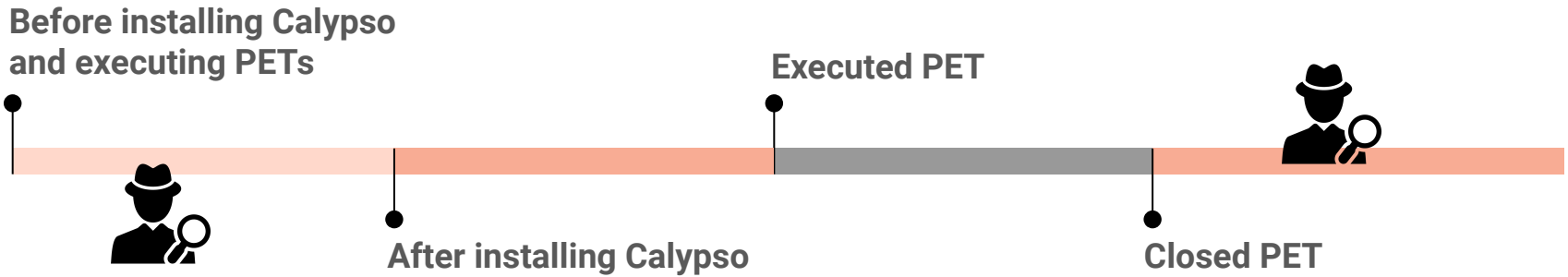
+

PETs do not provide plausible deniability

▶

Many people fear the consequences of using PETs!

Threat model



Isolation + Non-observability → Plausible deniability

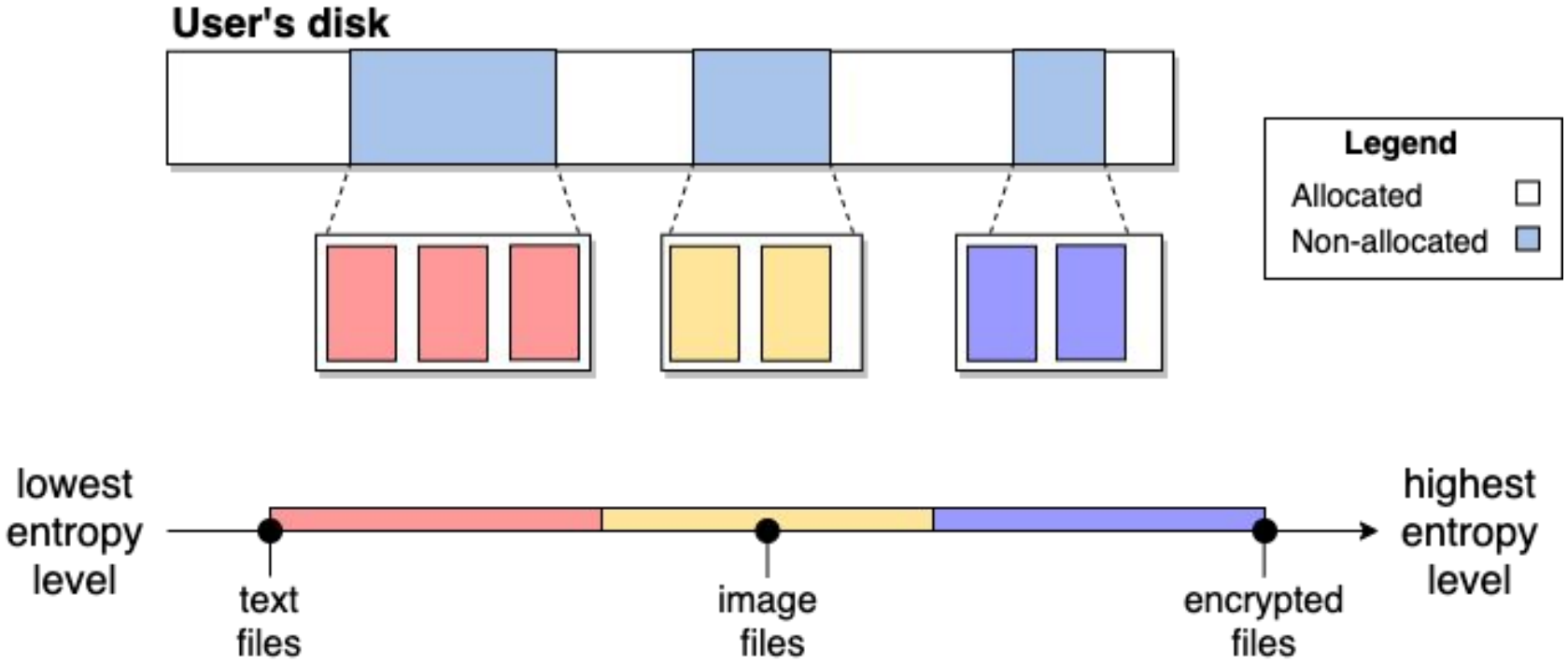
Traceless bootstrap

Block allocation

Monitoring native changes to blocks

Deniably encoding data blocks

Profiling the user's disk



Isolation

File category	Initial state		Tor native		Tor Calypso	
	Accesses count	Different file count	Accesses count	Different file count	Accesses count	Different file count
Temporary memory only	2138	67	4268	742	11377	5298
Temporary in disk	124	2	99	54	618	26
System config and OS re-sources	448	27	3015	161	2913	145
Binaries, libs and apps	239	62	3276	1389	1516	370
User data	0	0	2553	510	109	63
Calypso	0	0	0	0	318	75
Other	200	1	240	37	421	27

Evaluation

Functionality

Isolation

Security

Performance

Calypso: design obstacles

Functionality

- Test Calypso's ability to sustain the execution of PET applications without file system errors

Isolation

- Ensure Calypso leaves no persistent traces

Security

-

Performance

-

Why people are afraid of using PETs

Lack of usability

Large amounts of local resources

Extensive digital footprint

Forensic inspection of devices

PETs do not provide plausible deniability

People are afraid of using PETs

Goals and challenges

Goal

Build a system to execute PETs with plausible deniability, allowing users to perform private sessions

Challenges

- Make all changes to persistent state non-observable
- Maintain PETs usability, functionality and performance

Solutions

- Shadow partition that uses the free blocks of the native file system
- Encoded data can only be recovered by the rightful user

Contributions

Design

Calypso, a steganographic storage system
- Parasites on the free blocks of the native system to conceal data
- Preserves the initial entropy characteristics of the disk

Implementation

Fully functional prototype as a Linux kernel module for version 5.4

Evaluation

Extensive evaluation of the prototype

Conclusions

State-of-the-art PETs are vulnerable to forensic analysis

Calypso makes non-observable changes to disk

Give protection against forensic analysis