Where Theory Meets Practice for Privacy Enhancing Technologies

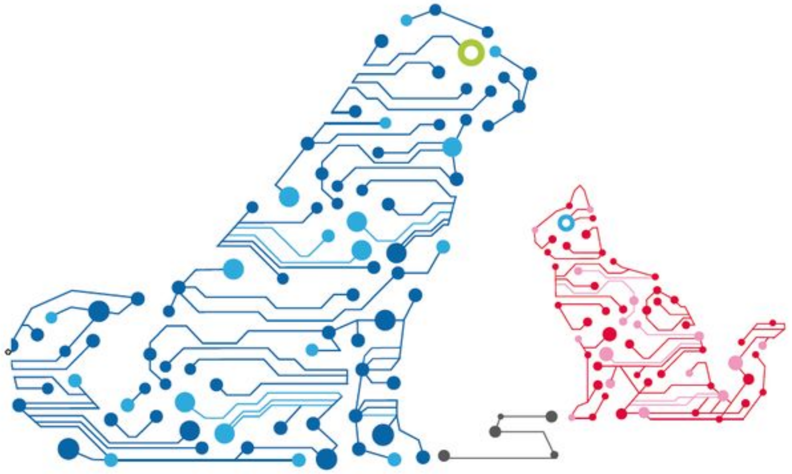
Presented by Chelsea H. Komlo

About me

- Software/security engineer
- HashiCorp, ThoughtWorks, Tor
- Worked in 5 countries and 2 languages

PETs rely on the research community

- Identifying vulnerabilities
- Creating new solutions to existing problems
- Reviewing/verifying designs
- Many more!



Source: https://www.enisa.europa.eu/news/enisa-news/if-you-want-a-friend-in-technology-get-a-pet-a-pet-maturity-assessment-tool-by-enisa

Thank you!

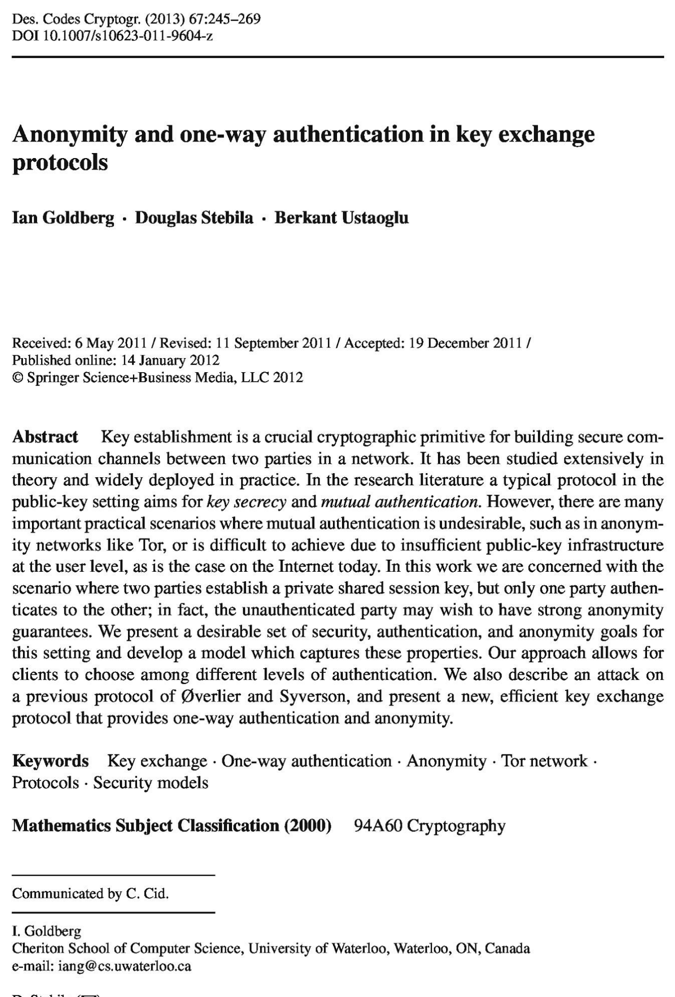
- To CrySP for hosting us!
- To the Tor team who provided feedback and ideas for this talk

In an ideal world

There would be a symbiotic relationship between those doing research and those whose projects would benefit from research.

Source: https://www.cypherpunks.ca/~iang/pubs/ntor.pdf

NTor Protocol


Des. Codes Cryptogr. (2013) 67:245–269
DOI 10.1007/s10623-011-9604-z

Anonymity and one-way authentication in key exchange protocols

Ian Goldberg · Douglas Stebila · Berkant Ustaoglu

Received: 6 May 2011 / Revised: 11 September 2011 / Accepted: 19 December 2011 /
Published online: 14 January 2012

Abstract Key establishment is a crucial cryptographic primitive for building secure communication channels between two parties in a network. It has been studied extensively in theory and widely deployed in practice. In the research literature a typical protocol in the public-key setting aims for *key secrecy* and *mutual authentication*. However, there are many important practical scenarios where mutual authentication is undesirable, such as in anonymity networks like Tor, or is difficult to achieve due to insufficient public-key infrastructure at the user level, as is the case on the Internet today. In this work we are concerned with the scenario where two parties establish a private shared session key, but only one party authenticates to the other; in fact, the unauthenticated party may wish to have strong anonymity guarantees. We present a desirable set of security, authentication, and anonymity goals for this setting and develop a model which captures these properties. Our approach allows for clients to choose among different levels of authentication. We also describe an attack on a previous protocol of Øverlier and Syverson, and present a new, efficient key exchange protocol that provides one-way authentication and anonymity.

Keywords Key exchange · One-way authentication · Anonymity · Tor network · Protocols · Security models

Mathematics Subject Classification (2000) 94A60 Cryptography

Communicated by C. Cid.

I. Goldberg
Cheriton School of Computer Science, University of Waterloo, Waterloo, ON, Canada
e-mail: iang@cs.uwaterloo.ca

Lots of good ideas, difficult to incorporate

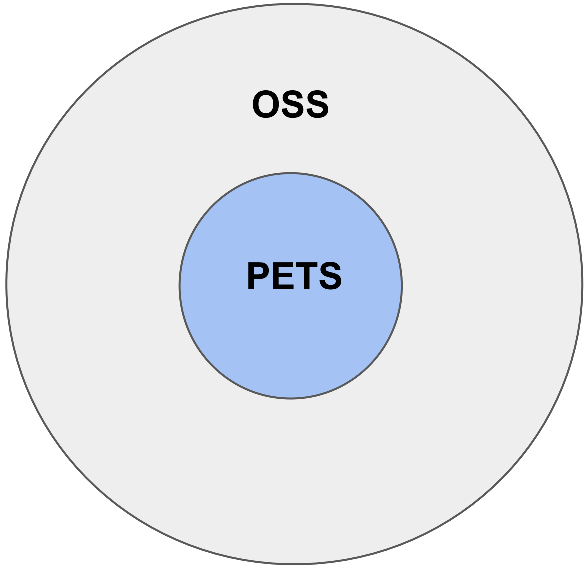
- Integrating new research findings isn’t seamless or predictable
- Good ideas get lost or forgotten about

In this talk, we will:

- Examine why good ideas fail to be incorporated into privacy enhancing technologies
- Use a hypothetical case study to examine these issues
- Look at positive cases and overall takeaways

Focus on OSS and PETS

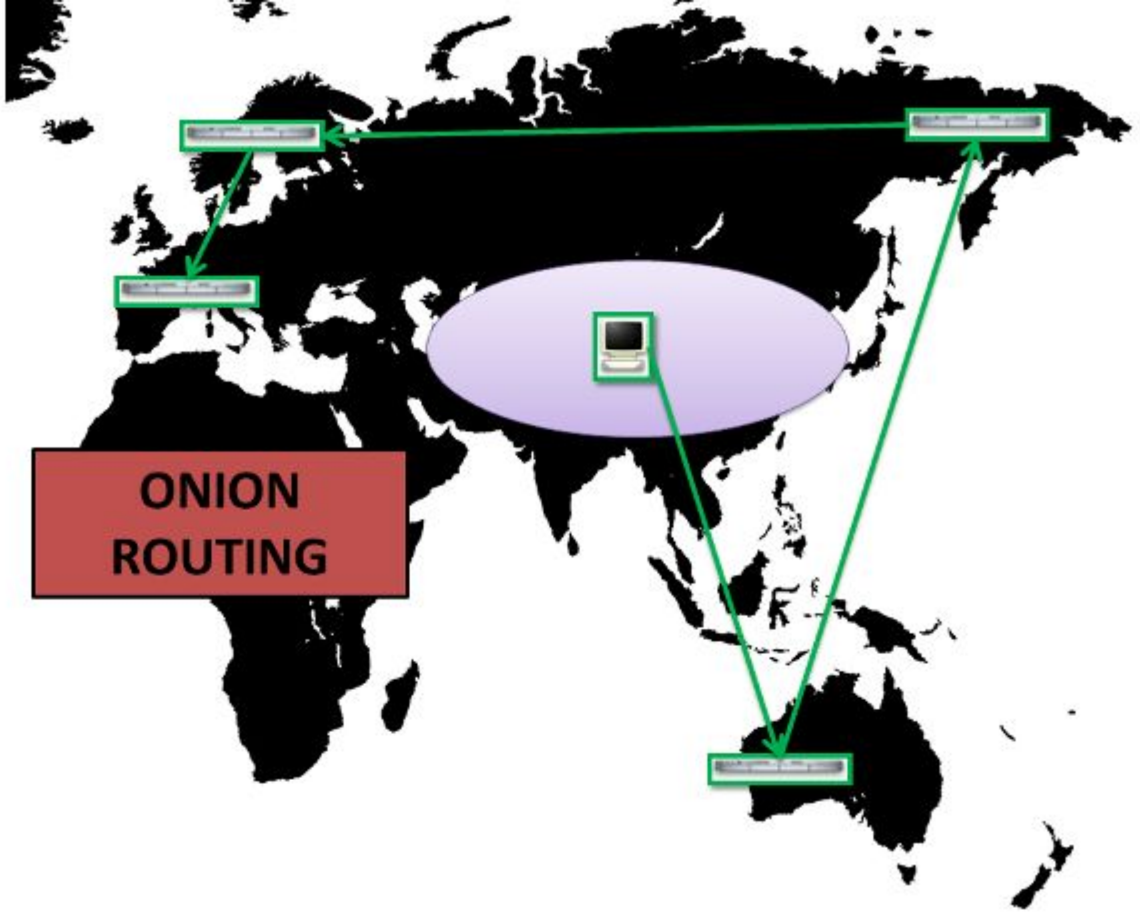
Common to open-source projects and PETS-specific



Contrived case study: Tor should migrate to use a mixnet protocol

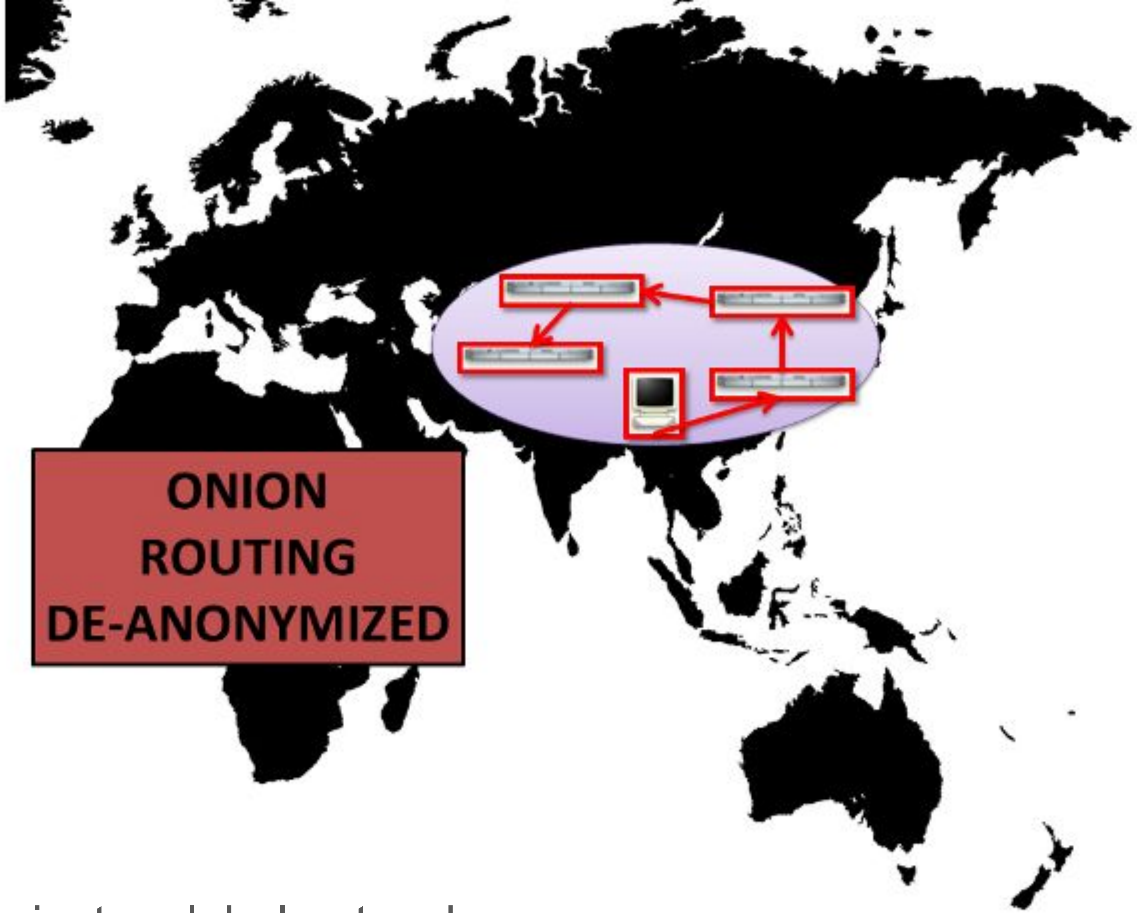
- Onion routing uses a connection-based protocol to hide the client's IP address via a multi-hop path between source and destination.
- Mixnet Protocol A: a connectionless protocol that uses permutations at each stage along a given path between source/destination.

Source: https://www2.ee.washington.edu/research/nsl/papers/proceedings-06.pdf



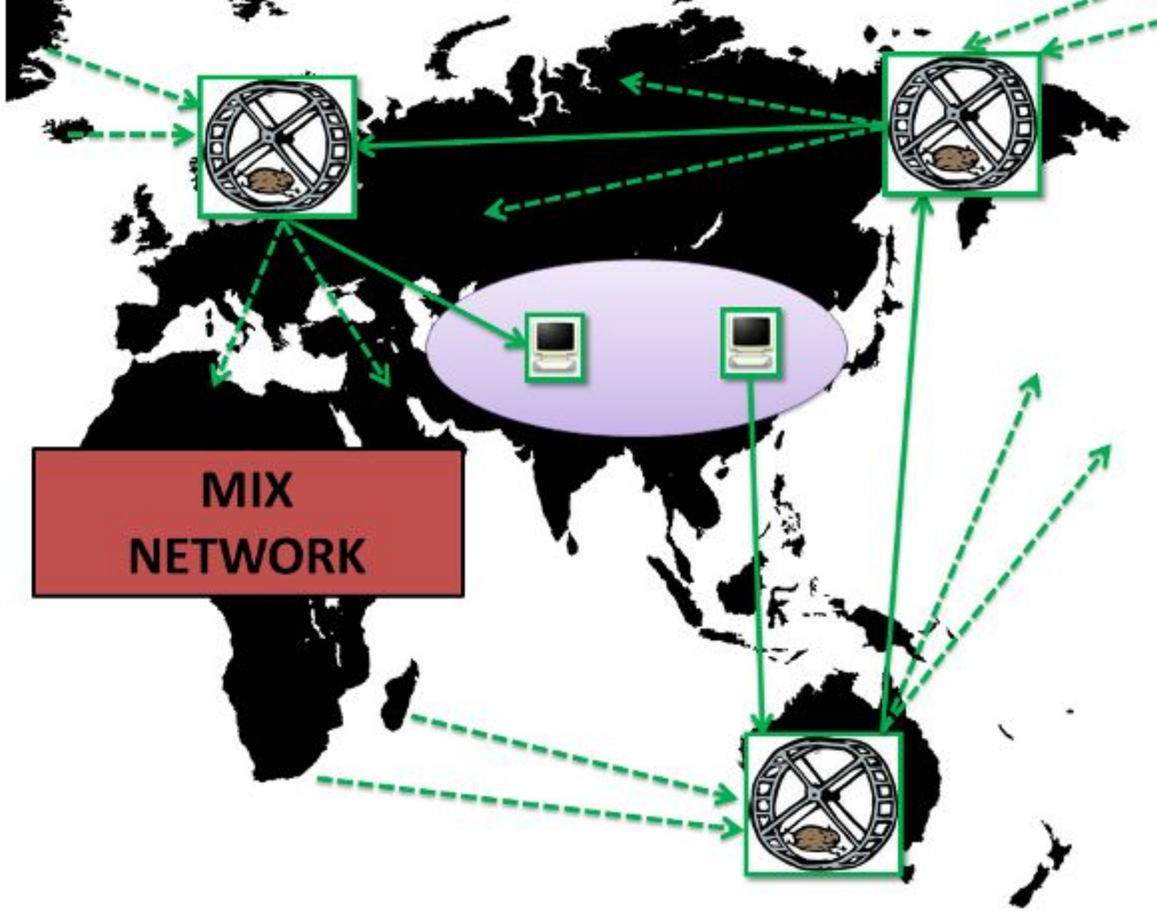
Onion Routing protects against local adversaries

Source: https://ritter.vg/blog-mix_and_onion_networks.html



Onion routing doesn’t protect against a global network adversary

Source: https://ritter.vg/blog-mix_and_onion_networks.html



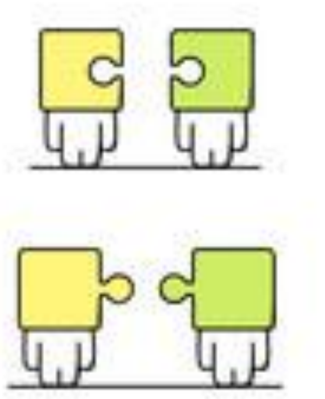
Mixnets create uncertainty by “mixing” at each hop

Source: https://ritter.vg/blog-mix_and_onion_networks.html

Why good ideas fail to be incorporated (short list)

- Incompatible design
- Inconclusive analysis
- Insufficient information/reproducibility
- Sub-par implementation
- People difficulties

Why good research ideas are not incorporated: Incompatible Design



Source: https://goo.gl/images/t8QA8K

Case study: Tor should move to a mixnet protocol

A research study showed that the security properties of Tor would improve by moving to use Mixnet Protocol A.

Doing so would provide resistance to a global network adversary.

Tor should move to a mixnet protocol: Analysis

Certain properties could improve, but must be weighed against tradeoffs:

- Resources required for the engineering effort?
- Completing the engineering effort with zero network downtime?

Incompatible design examples

- Incompatible with current/future architecture

Incompatible design examples

- Incompatible with current/future architecture
- Incompatible with the project’s goals and purpose

Incompatible design examples

- Incompatible with current/future architecture
- Incompatible with the project’s goals and purpose
- Incompatible with day-to-day operations (providing network connectivity)

Incompatible Design: How to Improve

- Along with one big design, offer incremental changes and MVP

How would you incrementally roll out transitioning from onion routing to a mixnet protocol?

Incompatible Design: How to Improve

- Recommend an iterative solution

Goal: Do not submit a “switch Tor to use mixnets instead of onion routing” patch!

Incompatible Design: How to Improve

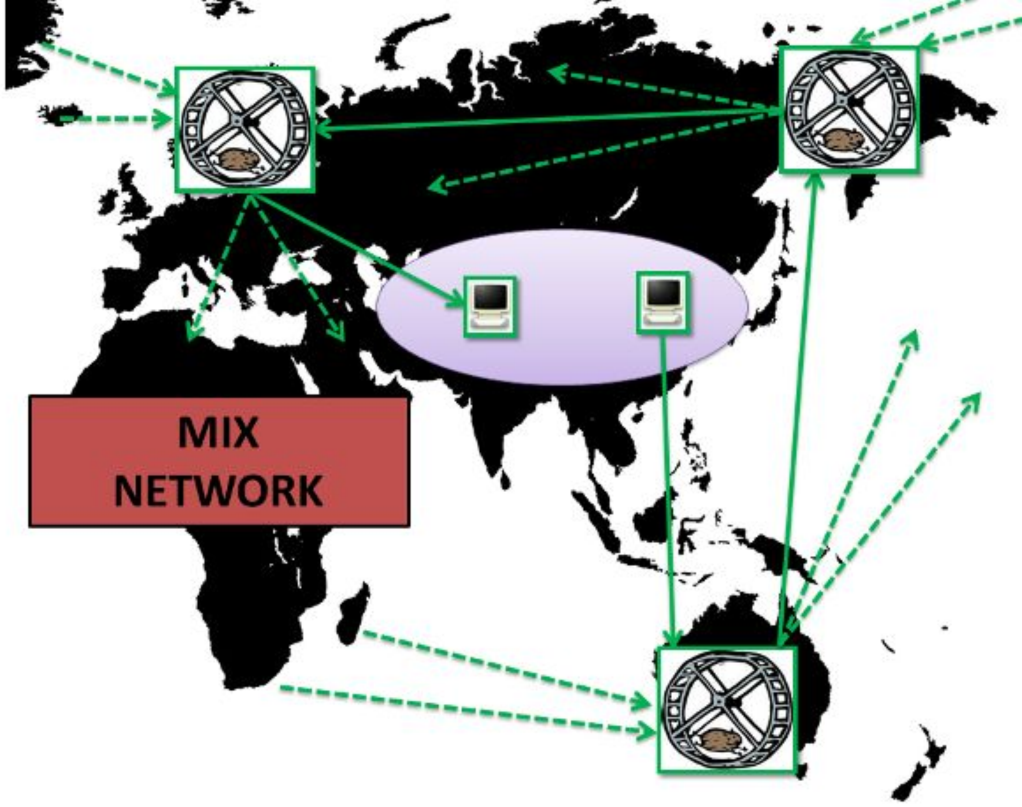
- Avoid breaking upgrades/downgrades

What does a graceful downgrade path look like?

Why good research ideas are not incorporated: Inconclusive Analysis

Case study: Tor should move to a mixnet protocol

Adding latency and “batching” between receiving and sending packets would raise the bar for end-to-end timing attacks.



Mixnet Protocol A adds random “batching” at each hop

Source: https://ritter.vg/blog-mix_and_onion_networks.html

Tor should move to a mixnet protocol: Analysis

- How does the proposed design impact other engineering tradeoffs?

Lots of tradeoffs!

- Security/Tor’s threat model
- Performance (per relay and for the entire network)
- Scalability (does this scheme scale to thousands of nodes)
- User experience
- Network load
- Engineering complexity (will it take the entire team a year to build)
- Engineering maintainability (will this require a lot of custom hard-to-maintain functionality?)
- Protocol complexity
- Failure cases (nodes should be able to intermittently fail, etc)
- Corner cases
- Extreme cases (DDOS, etc)
- Compatibility with upgrades/downgrades/multi-version deployments

Inconclusive Analysis: Examples

- How to continue supporting the current uses of Tor:
  - Thousands of relays
  - Millions of users
  - Wide application usage: file sharing to mobile browsing

Inconclusive Analysis: How to Improve

- Compare tradeoffs to industry standards
    - How user experience degrades with increased latency

Inconclusive Analysis: How to Improve

- Compare tradeoffs to industry standards
    - How user experience degrades with increased latency
- Consider tradeoffs holistically
    - Does increased security properties outweigh the proposed solution?
    - Provide information in such a way to reduce cognitive load for maintainers.

Why good research ideas are not incorporated: Insufficient Information, Reproducibility

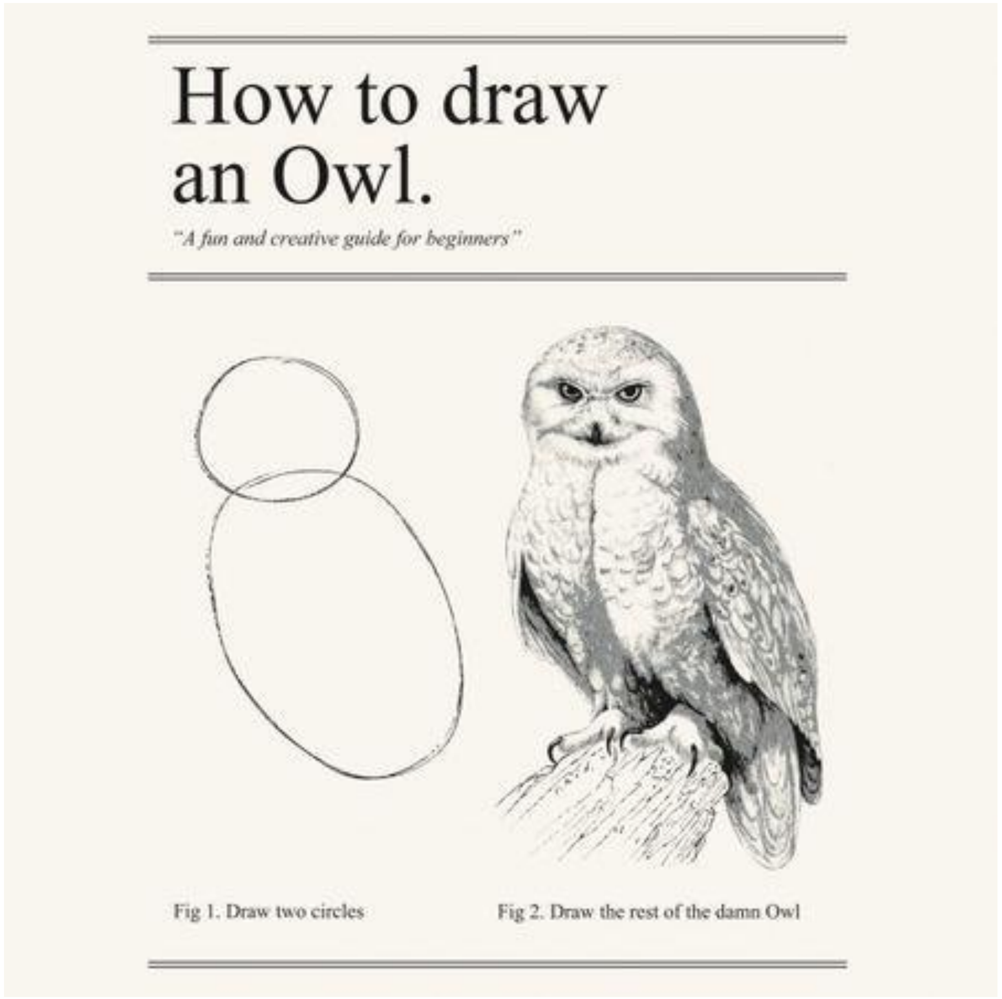

How to draw an Owl.
"A fun and creative guide for beginners"
Fig 1. Draw two circles
Fig 2. Draw the rest of the damn Owl

Image source: https://goo.gl/images/cJYNvp

Case study: Tor should move to a mixnet protocol

Add latency between receiving and sending packets to prevent end-to-end timing attacks.

This latency can be randomly chosen but should be influenced by certain properties depending on server load.

These results were validated on a test network of 10 nodes.

Tor should move to a mixnet protocol: Analysis

Several issues that would make reproducing this difficult:

- Not all parameters are published (for example, how to calculate mixing latency)
- Not clear what the upper/lower bands are for latency
- How would this experiment turn out differently on a larger network?

Insufficient Information/Reproducibility: Examples

- Unable to independently reproduce results

Insufficient Information/Reproducibility: Examples

- Unable to independently reproduce results
- Poor documentation

Insufficient Information/Reproducibility: Examples

- Unable to independently reproduce results
- Poor documentation
- Lack of explicitly-defined assumptions

Insufficient Information/Reproducibility: Examples

- Unable to independently reproduce results
- Poor documentation
- Lack of explicitly-defined assumptions
- Insufficient research rigor
    - Small data sets
    - Insufficiently controlled experiments

Insufficient Information/Reproducibility: How to Improve

- Peer review to assess “how independently reproducible are results?”
  - Are code samples provided?
  - Are test vectors provided?
  - Are library dependencies provided, and operating system tests were performed on?

Insufficient Information/Reproducibility: How to Improve

- Ensure implementation code doesn’t go away, even a few years later

Why good research ideas are not incorporated: Sub-Par Implementation

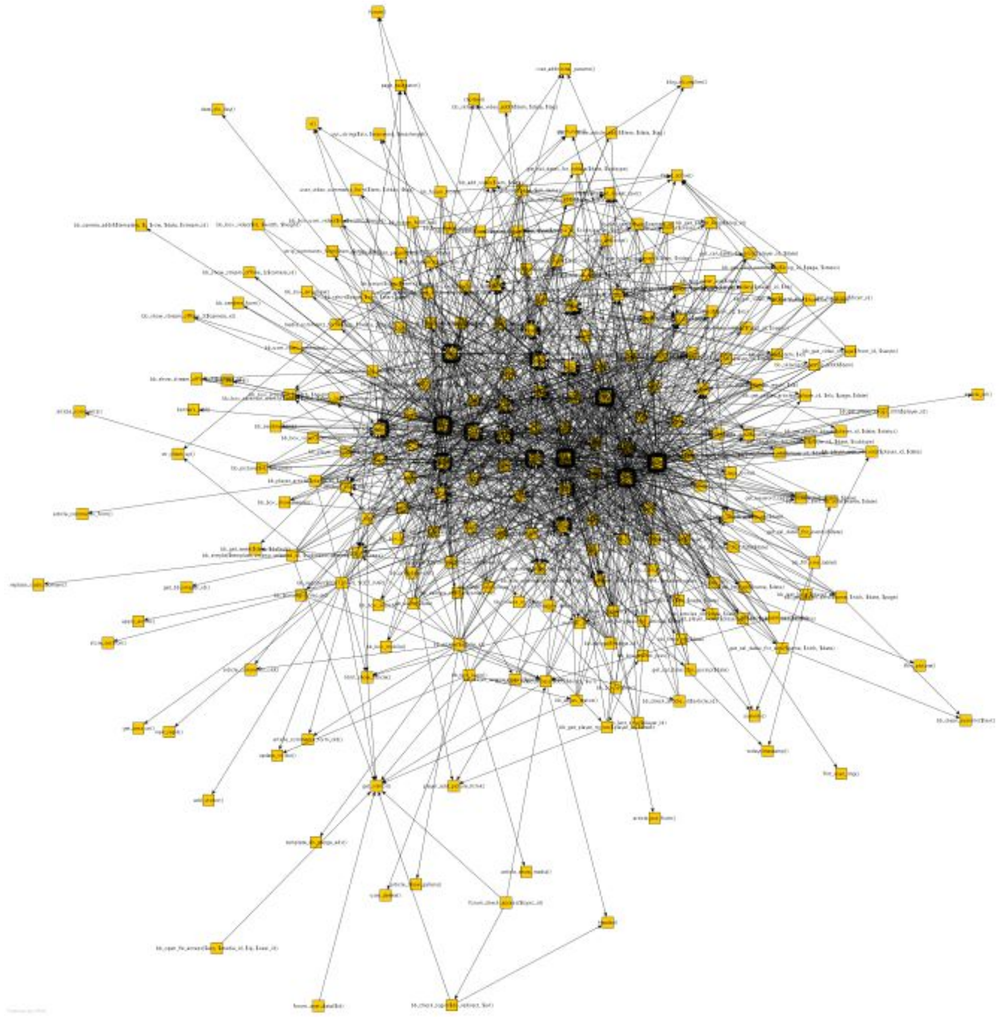


Image source: https://goo.gl/images/cdqHN3

Case study: Tor should move to a mixnet protocol

A paper has successful results, and a patch is ready to be integrated into Tor.

The patch is 10,000 lines of code without tests.

Case study: Tor should move to a mixnet protocol

This causes several difficulties for the team maintaining the project:

- How to sufficiently review the patch?
- How to know the patch does what is expected?
- How to know that this patch performs at scale?
- How to reconcile the patch with more recent changes to the codebase?

Sub-par Implementation: Examples

- Large blob

Sub-par Implementation: Examples

- Large blob
- Code drift

Sub-par Implementation: Examples

- Large blob
- Code drift
- Untested

Sub-par Implementation: Examples

- Large blob
- Code drift
- Untested
- Undocumented

Sub-par Implementation: Examples

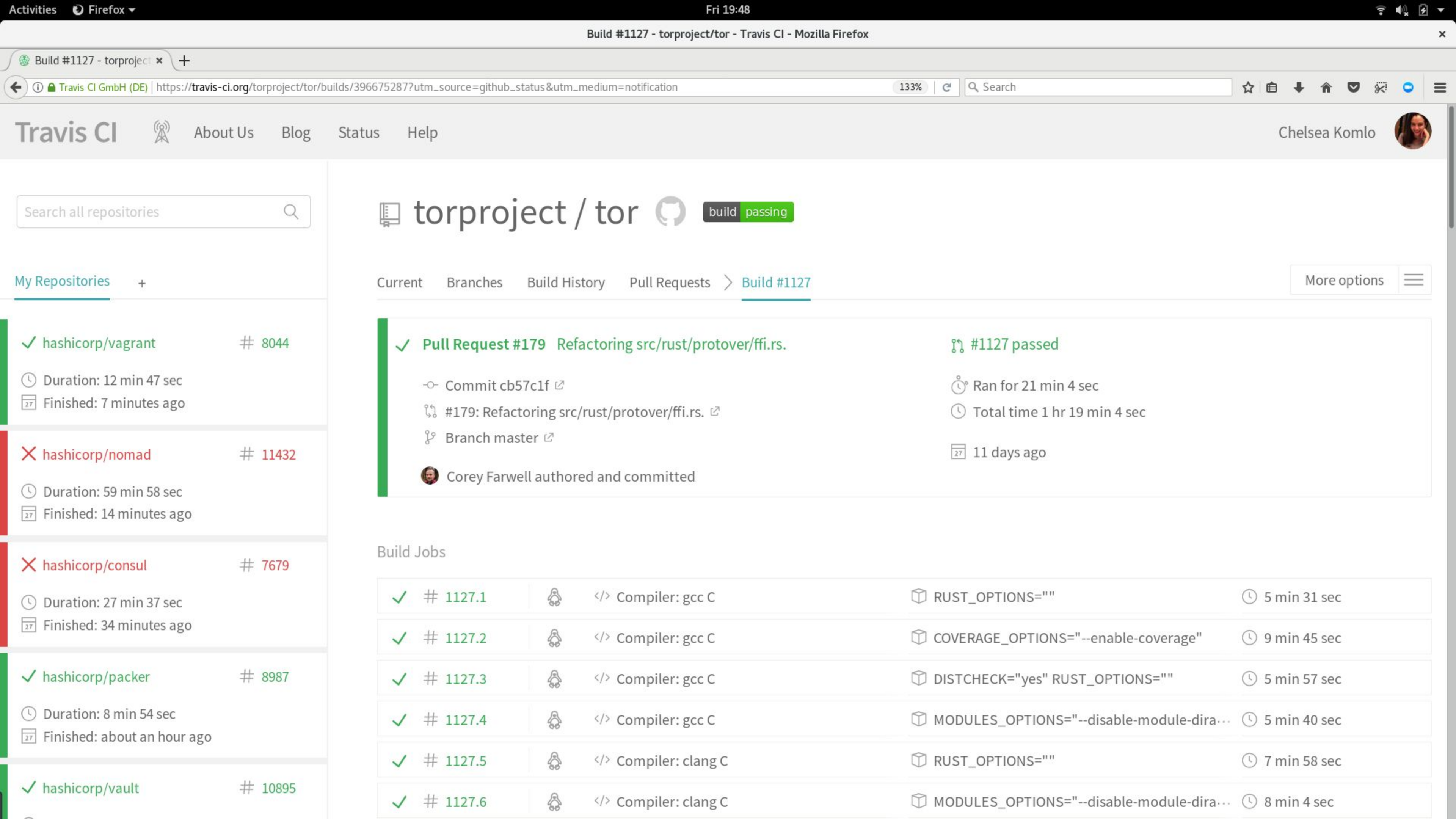
- Large blob
- Code drift
- Untested
- Undocumented
- Interwoven and cyclic dependencies

Sub-par Implementation: Examples

- Large blob
- Code drift
- Untested
- Undocumented
- Interwoven and cyclic dependencies
- Fragile

Sub-Par Implementation: How to Improve

- Write tests! Serves to both document and prove your code functions as expected



Build #1127 - torproject/tor - Travis CI - Mozilla Firefox

https://travis-ci.org/torproject/tor/builds/396675287?utm_source=github_status&utm_medium=notification

Travis CI   About Us   Blog   Status   Help   Chelsea Komlo

Search all repositories

My Repositories +

✓ hashicorp/vagrant # 8044
Duration: 12 min 47 sec
Finished: 7 minutes ago

✗ hashicorp/nomad # 11432
Duration: 59 min 58 sec
Finished: 14 minutes ago

✗ hashicorp/consul # 7679
Duration: 27 min 37 sec
Finished: 34 minutes ago

✓ hashicorp/packer # 8987
Duration: 8 min 54 sec
Finished: about an hour ago

✓ hashicorp/vault # 10895

torproject / tor   build passing

Current   Branches   Build History   Pull Requests > Build #1127   More options

✓ Pull Request #179 Refactoring src/rust/protover/ffi.rs.   #1127 passed

Commit cb57c1f   Ran for 21 min 4 sec
#179: Refactoring src/rust/protover/ffi.rs.   Total time 1 hr 19 min 4 sec
Branch master   11 days ago
Corey Farwell authored and committed

Build Jobs

✓	# 1127.1	Compiler: gcc C	RUST_OPTIONS=""	5 min 31 sec
✓	# 1127.2	Compiler: gcc C	COVERAGE_OPTIONS="--enable-coverage"	9 min 45 sec
✓	# 1127.3	Compiler: gcc C	DISTCHECK="yes" RUST_OPTIONS=""	5 min 57 sec
✓	# 1127.4	Compiler: gcc C	MODULES_OPTIONS="--disable-module-dira···	5 min 40 sec
✓	# 1127.5	Compiler: clang C	RUST_OPTIONS=""	7 min 58 sec
✓	# 1127.6	Compiler: clang C	MODULES_OPTIONS="--disable-module-dira···	8 min 4 sec

Sub-Par Implementation: How to Improve

- Meet with the team along the way, don’t just pass a big blob

Sub-Par Implementation: How to Improve

- Write code that is maintainable for the next 5-10 years

Sub-Par Implementation: How to Improve

- Fight code drift- continuously rebase your branch to the project’s master branch.

People Difficulties

Image Source: https://goo.gl/images/f3PtPh

Case study: Tor should move to a mixnet protocol

Research is done, paper published, researchers moving on.

The “publish or perish” model required of researchers doesn’t allow a lot of time after a publication to see through outcomes.

Case study: Tor should move to a mixnet protocol

After the research is done, a lot more needs to happen:

- Onboarding the team so they both understand the domain and the proposed solution
- Ensuring unexpected issues are resolved (does scaling to 100x have unexpected consequences?)

People Difficulties Examples

- Lack of domain knowledge within the team makes analysis and prioritizing the work difficult

People Difficulties Examples

- Lack of domain knowledge within the team makes analysis and prioritizing the work difficult
- Lack of communication with the team means questions/concerns aren’t asked

People Difficulties: How to Improve

- Have a champion within the project

People Difficulties: How to Improve

- Offer to pair during the implementation to ensure questions and remaining work are taken care of.

Positive example

Source: https://goo.gl/images/aL2d5s

Case study: KIST

- "Kernel Informed Socket Transport"
- Uses feedback from kernel congestion to inform how much data to transmit
- Changes priority scheduling from per-socket to all circuits.

Never Been KIST: Tor's Congestion Management Blossoms with Kernel-Informed Socket Transport

Rob Jansen[†] *John Geddes*[‡] *Chris Wacek*[*] *Micah Sherr*[*] *Paul Syverson*[†]

[†] *U.S. Naval Research Laboratory*
{rob.g.jansen, paul.syverson}@nrl.navy.mil

[‡] *University of Minnesota*
geddes@cs.umn.edu

[*] *Georgetown University*
{cwacek, msherr}@cs.georgetown.edu

Abstract

Tor's growing popularity and user diversity has resulted in network performance problems that are not well understood. A large body of work has attempted to solve these problems without a complete understanding of where congestion occurs in Tor. In this paper, we first study congestion in Tor at individual relays as well as along the entire end-to-end Tor path and find that congestion occurs almost exclusively in egress kernel socket buffers. We then analyze Tor's socket interactions and discover two major issues affecting congestion: Tor writes sockets sequentially, and Tor writes as much as possible to each socket. We thus design, implement, and test KIST: a new socket management algorithm that uses real-time kernel information to *dynamically compute the amount to write* to each socket while considering *all writable circuits* when scheduling new cells. We find that, in the medians, KIST reduces circuit congestion by over 30 percent, reduces network latency by 18 percent, and increases network throughput by nearly 10 percent. We analyze the security of KIST and find an acceptable performance and security trade-off, as it does not significantly affect the outcome of well-known latency and throughput attacks. While our focus is Tor, our techniques and observations should help analyze and improve overlay and application performance, both for security applications and in general.

1 Introduction

Tor [21] is the most popular overlay network for communicating anonymously online. Tor serves millions of users daily by transferring their traffic through a source-routed *circuit* of three volunteer relays, and encrypts the traffic in such a way that no one relay learns both its source and intended destination. Tor is also used to resist online censorship, and its support for hidden services, network bridges, and protocol obfuscation has helped attract a large and diverse set of users.

While Tor's growing popularity, variety of use cases, and diversity of users have provided a larger anonymity set, they have also led to performance issues [23]. For example, it has been shown that roughly half of Tor's traffic can be attributed to BitTorrent [18, 43], while the more recent use of Tor by a botnet [29] has further increased concern about Tor's ability to utilize volunteer resources to handle a growing user base [20, 36, 37, 45].

Numerous proposals have been made to battle Tor's performance problems, some of which modify the mechanisms used for path selection [13, 59, 60], client throttling [14, 38, 45], circuit scheduling [57], and flow/congestion control [15]. While some of this work has or will be incorporated into the Tor software, none of it has provided a comprehensive understanding of *where* the most significant source of congestion occurs in a complete Tor deployment. This lack of understanding has led to the design of uninformed algorithms and speculative solutions. In this paper, we seek a more thorough understanding of congestion in Tor and its effect on Tor's security. We explore an answer to the fundamental question—"*Where* is Tor slow?"—and design informed solutions that not only decrease congestion, but also improve Tor's ability to manage it as Tor continues to grow.

Congestion in Tor: We use a multifaceted approach to exploring congestion. First, we develop a shared library and Tor software patch for measuring congestion *local to relays* running in the public Tor network, and use them to measure congestion from three live relays under our control. Second, we develop software patches for Tor and the open-source Shadow simulator [7], and use them to measure congestion along the *full end-to-end path* in the largest known, at-scale, private Shadow-Tor deployment. Our Shadow patches ensure that our congestion measurements are accurate and realistic; we show how they significantly improve Shadow's TCP implementation, network topology, and Tor models.[1]

[1]We have contributed our patches to the Shadow project [7] and they have been integrated as of Shadow release `1.9.0`.

1

Case study: Kist

- Worked closely with the development team

Case study: Kist

- Worked closely with the development team
- Several rounds of iteration for performance

Case study: Kist

- Worked closely with the development team
- Several rounds of iteration for performance
- Sufficient funding/man hours for implementation and code revision

Overall Takeaways

Source: https://goo.gl/images/uBPEjW

Takeaways: Research

- Holistic and rigorous analysis

Takeaways: Research

- Holistic and rigorous analysis
- Consideration for engineering tradeoffs

Takeaways: Research

- Holistic and rigorous analysis
- Consideration for engineering tradeoffs
- Sufficient information and ease of reproducibility

Takeaways: Implementation

- Maintainable code

Takeaways: Implementation

- Maintainable code
- Sufficient documentation

Takeaways: Implementation

- Maintainable code
- Sufficient documentation
- Sufficient knowledge transfer for project maintainers

Thank you!

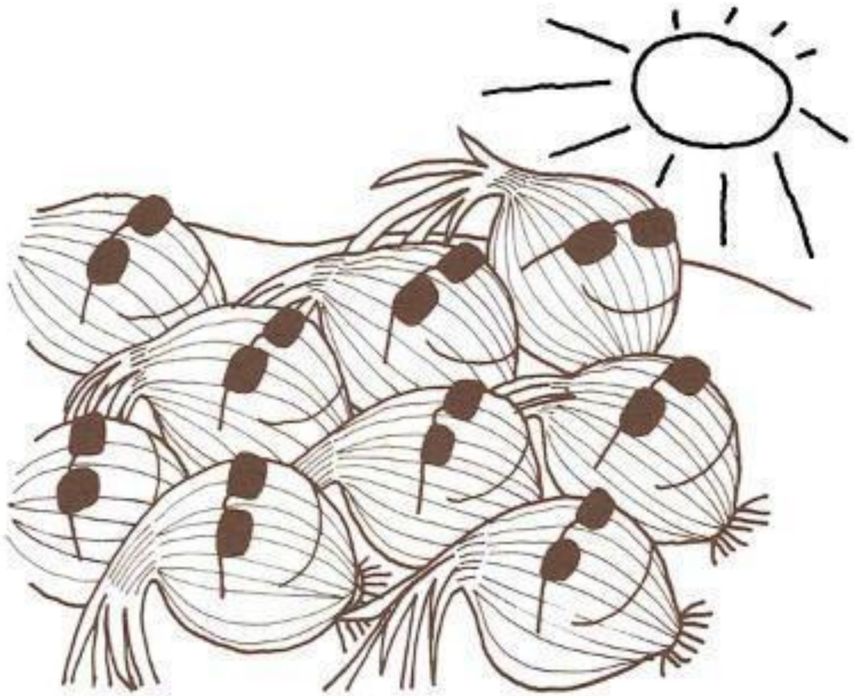


Image source: https://goo.gl/images/YSnGra