

Design and implementation of a multi-layer privacy-preserving router based on raspberry Pi platform: Integration of the Onion router network with advanced DNS protection mechanisms

Michał Kalisz¹, Konrad Hukowski², Michalina Gryniewicz-Jaworska^{3*} 

¹ Department of the Foundations of Computer Science, Faculty of Philosophy, The John Paul II Catholic, University of Lublin, Aleje Racławickie 14, 20-950 Lublin, Poland

² Ministry of Internal Affairs and Administration, ul. Stefana Batorego 5, 02-591 Warsaw, Poland

³ WSEI University, Projektowa 4, 20-209 Lublin, Poland

* Corresponding author's e-mail: michalina.gryniewicz-jaworska@wsei.pl

ABSTRACT

This paper presents the design and implementation of a multi-layer privacy-preserving network router built on the Raspberry Pi 5 platform. The proposed system integrates several complementary privacy-enhancing technologies, namely the TOR anonymization network, encrypted DNS protocols (DNSCrypt, DNS over HTTPS, DNS over TLS, and Oblivious DNS over HTTPS), a local recursive DNS resolver (Unbound), and DNS-level filtering via Pi-hole. A total of nine distinct operational modes were developed, each offering a different trade-off between the level of privacy protection and network performance — ranging from basic advertisement filtering to full traffic anonymization combined with multi-layer DNS encryption. A physical control interface featuring an OLED display and push buttons was implemented, enabling mode switching without remote shell access. Performance evaluation using the DNS Performance Testing Tool (dnssperf) demonstrated that even the most advanced configurations maintain acceptable response latencies and high query resolution rates. The proposed solution addresses the gap between expensive, closed-source commercial devices and complex open-source systems that require advanced technical expertise, offering an accessible and cost-effective privacy protection tool for individual users and high-risk groups. The total hardware cost does not exceed 200 USD, representing a fraction of the price of commercial solutions with comparable functionality.

Keywords: TOR network, Raspberry Pi, DNS encryption, DNSCrypt, DNS over HTTPS, Pi-hole, Unbound, privacy-preserving router, network anonymization, DNS filtering.

INTRODUCTION

The contemporary digital landscape is characterized by an unprecedented scale of surveillance and analysis of users' online activity. The proliferation of digital monitoring and mass-scale traffic analysis poses a significant threat to individual privacy. According to the Freedom House report, approximately 70% of the global population resides in countries where Internet traffic is subject to systematic monitoring [1].

Concurrently, research by the Electronic Frontier Foundation indicates that an average Internet user is tracked by approximately 72 trackers during a single browsing session [2]. This issue is particularly relevant given the growing number of Internet of Things (IoT) devices in household environments — Gartner projected that the global number of connected IoT devices would surpass 75 billion by 2025 [3].

Existing privacy-protection solutions can be broadly categorized into three groups: commercial

VPN appliances (often based on proprietary firmware and requiring expensive subscriptions), privacy-oriented operating systems (such as Tails or Whonix, which are not designed to function as persistent network infrastructure components), and software-based solutions that demand considerable technical expertise for proper configuration. None of these categories provides an affordable, modular, and easily manageable hardware-based solution suitable for home network environments.

The TOR network (The Onion Router), utilized by over two million users daily [4], remains one of the most effective technologies for traffic anonymization. However, its practical deployment encounters several technical challenges, including DNS leaks that, according to empirical research [5], affect up to 40% of improperly configured installations. Furthermore, the increasing sophistication of correlation attacks and traffic timing analysis represent substantial threats to user anonymity.

In response to these identified challenges, this paper presents the design and implementation of a multi-layer privacy-preserving router based on the Raspberry Pi 5 platform. The primary research objective was to develop an integrated system combining anonymization technologies (TOR), advanced DNS encryption protocols (DNSCrypt, DoH, DoT, ODoH), a local DNS resolver (Unbound), and a DNS-level filtering system (Pi-hole). This paper describes a practical solution to the aforementioned problems through the construction of a cost-effective anonymizing router.

The following research questions guided this study:

- RQ1: Is it feasible to effectively integrate multiple layers of privacy protection on a platform with limited computational resources?
- RQ2: What is the impact of individual security configurations on overall system performance?
- RQ3: Does the modular architecture enable dynamic adjustment of the protection level according to user requirements?

The principal design objectives of the project were as follows:

Design and implementation of discrete operational modes offering varying levels of privacy protection, ranging from basic DNS filtering to full anonymization with multi-layer encryption.

Development of a physical control interface utilizing an OLED display and push buttons, enabling mode switching without remote access.

Integration of heterogeneous privacy-enhancing technologies within a single, compact, low-power-consumption device.

Documentation of a complete implementation methodology enabling solution replication.

STATE OF THE ART

Theoretical foundations of network traffic anonymization

The foundations of modern network traffic anonymization systems were established by Dingledine, Mathewson, and Syverson [6], who introduced the architecture of the TOR network — a protocol based on onion routing that ensures anonymity through multi-layer encryption and the routing of traffic through a chain of randomly selected relay nodes. The TOR network implements a distributed trust model in which no single node simultaneously possesses information about both the sender's identity and the communication content. This design builds upon the earlier Onion Routing concept, augmented with mechanisms for perfect forward secrecy, congestion control, and configurable exit policies.

The TOR architecture stipulates that a user's connection traverses at least three nodes: an entry guard, a middle relay, and an exit node. The exit node establishes the connection with the destination server, thereby constituting a potential vulnerability point — particularly in the context of unencrypted DNS queries. This problem was examined in depth by Greschbach et al. [5], who demonstrated that the Google DNS resolver observes nearly 40% of all DNS queries originating from the TOR network, enabling correlation attacks. Their study further showed that DNS queries frequently traverse autonomous systems (ASes) that the corresponding TCP connections do not, thereby creating additional deanonymization vectors.

DNS privacy protection protocols

In response to the identified threats associated with unencrypted DNS communication, a number of protective protocols have been developed. Mockapetris [7, 8] defined the foundational DNS architecture in RFC 1034 and RFC 1035, which operated for nearly three decades without built-in confidentiality mechanisms. The introduction

of DNSSEC extensions (RFC 4033, Arends et al. [9]) provided integrity and authenticity verification for DNS responses, yet did not address the confidentiality of queries.

A breakthrough in DNS privacy was achieved with the development of DNS over TLS (DoT, RFC 7858, Hu et al. [10]) and DNS over HTTPS (DoH, RFC 8484, Hoffman and McManus [11]), both of which introduce encryption of the communication channel between client and resolver. DoT utilizes a dedicated port 853 with a TLS connection, whereas DoH tunnels DNS queries through the HTTPS channel on port 443, thereby making it difficult to distinguish DNS traffic from regular web traffic. A further evolution of this concept is the Oblivious DNS over HTTPS protocol (ODOH, RFC 9230, Kinnear et al. [12]), which introduces a separation between the entity that knows the client's IP address and the entity that knows the content of the DNS query — the ODOH relay knows only the client's IP, while the target DNS resolver receives only the encrypted query without information about its source.

The most recent development is the DNS over QUIC protocol (DoQ), standardized in RFC 9250 (Huitema et al. [13]), which combines the advantages of encryption with lower latencies resulting from the elimination of the multi-step TCP+TLS handshake process. The integration of these protocols within a single network device constitutes one of the challenges addressed in this work.

Existing implementations of hardware-based TOR routers

The concept of a hardware router that anonymizes traffic through the TOR network has been the subject of numerous projects, both commercial and open-source. A review of the key solutions enables identification of the gap that this work aims to fill.

Anonabox was among the first commercial TOR routers to gain widespread attention following a Kickstarter campaign in 2014 that raised over 500,000 USD. The campaign was subsequently suspended after it was revealed that the device was based on off-the-shelf Chinese hardware with minimal firmware modification. Independent security audits identified critical vulnerabilities, including an open web interface accessible via IPv6 without authentication, an SSH server running on a random port, and the inability for users to change the Wi-Fi password. Although

later revisions (Anonabox Pro) addressed some of these issues — including effective tunneling of all traffic through TOR without DNS leaks — concerns regarding the proprietary source code and limited configurability persisted.

Safeplug ('TOR in a Box') was another commercial device subjected to a formal security audit (Edmundson et al. [14]). The researchers analyzed the hardware, firmware, network behavior, and usability of the device, identifying multiple vulnerabilities, including the absence of DNS leak protection and limited firmware update capabilities.

PORTAL (Personal Onion Router To Assure Liberty) is an open-source project developed by the security researcher known as 'grugq,' which utilizes the Raspberry Pi platform for transparent TCP traffic redirection through the TOR network. While functional, PORTAL is limited to basic traffic tunneling and does not offer additional DNS protection layers or filtering mechanisms.

Onion Pi is an educational project developed by Adafruit that transforms a Raspberry Pi into a portable Wi-Fi access point with a TOR proxy. This solution targets novice users and focuses on simplicity of configuration. Onion Pi does not offer advanced DNS protection, advertisement filtering, or mode-switching capabilities.

TorBox is an actively maintained open-source project that converts a Raspberry Pi into an anonymizing router with TOR support. TorBox supports clients via both Wi-Fi and Ethernet, offers a text-based menu interface, and supports Onion service publication. This project is the closest to the solution described in the present work; however, it differs in several key aspects: it does not integrate advanced DNS encryption protocols (DNSCrypt, DoH, DoT, ODOH), does not offer a local recursive resolver (Unbound), does not provide DNS-level filtering via Pi-hole, and lacks a physical control interface (OLED display with push buttons).

Commercial routers with TOR and VPN functionality

The market also offers commercial travel routers with built-in TOR support, such as devices from the GL.iNet series (e.g., GL-A1300 Slate Plus, GL-AXT1800 Slate AX). These routers offer ease of configuration and integration with VPN services; however, their TOR functionality is limited compared to dedicated solutions. As user analyses indicate, travel routers with TOR integration route traffic through the TOR network

but do not protect against browser-level threats such as fingerprinting, JavaScript vulnerabilities, or other tracking methods. Moreover, these devices are based on proprietary or partially open firmware, which limits the possibility of independent security auditing.

Privacy-oriented operating systems

An alternative approach to network traffic anonymization is implemented through dedicated operating systems. Tails (The Amnesic Incognito Live System) is a live distribution that routes all traffic through the TOR network and leaves no traces on the storage medium after shutdown. This system is designed to be booted from removable media and is not suitable for continuous operation as a permanent network infrastructure component.

Whonix employs an innovative two-machine architecture: Whonix-Gateway functions as a TOR router, while Whonix-Workstation is an isolated user environment that can communicate exclusively through the gateway. This approach effectively eliminates IP and DNS leaks; however, it requires a virtualization environment (e.g., VirtualBox) and is not designed to function as a physical network router serving multiple client devices.

Both systems, while highly effective at protecting a single user, do not implement the scenario of a network router serving multiple clients with a modular selection of protection levels — the functionality that constitutes the principal innovation of this work.

Identification of the research gap

The conducted review of existing solutions enables the identification of a distinct gap that this work aims to address. A comparison of key

features across existing solutions is presented in Table 1. None of the existing solutions simultaneously combines: (a) multi-layer DNS protection integrating DNSCrypt, DoH, DoT, and ODoH protocols; (b) a local recursive resolver (Unbound); (c) DNS-level sinkhole filtering (Pi-hole); (d) transparent traffic tunneling through the TOR network; (e) a physical control interface enabling dynamic switching between operational modes; and (f) open-source code on a low-cost hardware platform. The identified gap constitutes the primary motivation for this research.

SYSTEM DESIGN AND METHODOLOGY

Hardware platform and operating system selection

Designing an advanced yet compact network device with traffic anonymization capabilities requires careful selection of the hardware platform and operating system. The objective was to achieve a solution characterized by adequate computational power, stability, low power consumption, and high flexibility in terms of configuration and extensibility. The Raspberry Pi 5 single-board computer (8 GB variant) running Raspberry Pi OS (64-bit) was selected for this project. This choice was motivated by both practical and technological considerations.

The Raspberry Pi 5 represents the latest and most capable iteration of the widely adopted single-board computer (SBC) family. Compared to its predecessors, the Pi 5 delivers substantially higher performance through its quad-core ARM Cortex-A76 processor clocked at 2.4 GHz and 8 GB of RAM. These specifications enable the device to execute both basic networking tasks and

Table 1. Comparison of key features of existing privacy-preserving solutions

Solution	Type	Open source	Multi-layer DNS	DNS filtering	Physical interface	Mode modularity
Anonabox	Commercial	No	No	No	No	No
Safepug	Commercial	No	No	No	No	No
PORTAL	Open-source	Yes	No	No	No	No
Onion Pi	Open-source	Yes	No	No	No	No
TorBox	Open-source	Yes	No	No	Text menu	Partial
GL.iNet (Slate)	Commercial	Partial	No	No	Web UI	Limited
Whonix	OS	Yes	No	No	No	No
Tails	OS	Yes	No	No	No	No
This work	Open-source	Yes	Yes	Yes	OLED+buttons	Yes (9 modes)

more computationally demanding services, such as DNS encryption, recursive query handling, and full traffic tunneling through the TOR network. A notable innovation of the Raspberry Pi 5 is its PCIe Gen2 x1 interface, which enables connection of NVMe SSD drives. Employing such a storage medium significantly increases read and write speeds, thereby enhancing the device's reliability during continuous operation as a router or access point. Additionally, 802.11ac Wi-Fi support ensures fast and stable wireless connectivity for both local communication and Internet access. The complete assembly of the developed router is presented in Figure 1.

The Raspberry Pi 5 offers several advantages in networking applications. Its high CPU performance enables efficient execution of encryption and anonymization processes with minimal latency overhead. The device's low power consumption is particularly relevant for continuous 24/7 operation. The extensive and active user community, together with broad open-source support, ensures readily available documentation, libraries, and technical assistance. The compact form factor and modular design facilitate installation in custom enclosures and deployment in mobile applications. Full compatibility with a wide

range of accessories — including HAT expansion boards, OLED displays, and GPIO interfaces — renders the platform highly versatile for networking projects requiring high functionality at low deployment cost.

The Raspberry Pi OS 64-bit distribution, an official Debian-based Linux distribution optimized for the ARM architecture, was selected as the operating system. This choice was primarily motivated by the system's high stability and full compatibility with dedicated Raspberry Pi hardware, which translates directly into reliable platform operation. The 64-bit variant enables more efficient RAM management and supports applications with high resource demands — a critical consideration for a router operating in continuous mode and running multiple network services. Raspberry Pi OS provides extensive package availability and repository access, which considerably simplifies the installation of essential system components such as TOR, Unbound, dnscrypt-proxy, Pi-hole, and fail2ban.

As part of the deployment, a standard system installation was performed on a microSD card, which was subsequently cloned to a high-speed NVMe SSD via the M.2 HAT adapter. Migrating the system to the faster storage medium significantly improved operational performance and enhanced the stability of device operation in 24/7 mode.

Hardware and system configuration

Transforming the Raspberry Pi 5 into a fully functional router with traffic anonymization capabilities required comprehensive hardware and operating system configuration. The process began with the assembly of the hardware kit, including installation of a heatsink with active cooling, connection of the NVMe M.2 HAT expansion board, and mounting of the SSD that served as the primary storage medium. Through the PCIe interface and Raspberry Pi OS 64-bit, the operating system was booted directly from the high-speed SSD, significantly enhancing the stability and performance of the entire system. The hardware configuration, including the NVMe M.2 HAT expansion board, is shown in Figure 2.

Following the initial system installation using RPi Imager, a software update was performed, the user account was configured, and network services were enabled. Subsequently, SSH access was activated, the default port was changed, and authentication was restricted to cryptographic



Figure 1. The TOR router assembly with a USB Wi-Fi adapter, NVMe SSD drive, OLED control display, and heat sink



Figure 2. Raspberry Pi 5 with the NVMe M.2 HAT expansion board

key-based login exclusively. The fail2ban intrusion prevention tool was also configured to monitor system logs for the purpose of detecting and blocking potential brute-force attacks (Figure 3).

The device was then configured as a router and Wi-Fi access point. An external USB Wi-Fi adapter was installed and configured, assigned to a new wlan1 interface. Using the NetworkManager utility, a Wi-Fi access point was created with a custom SSID and WPA2-PSK password (Figure 4).

The dnsmasq service was deployed as a DHCP and DNS server, enabling IP address assignment to clients and domain name resolution (Figure 5). The configuration was completed with NAT and firewall rules in iptables, which provided address translation and packet routing between the local network and the Internet (Figure 6).

Implementation of DNS services and traffic filtering

In the subsequent development phase, particular attention was devoted to the DNS handling and network traffic filtering subsystem. First, the Pi-hole service was deployed as a local DNS sinkhole server. Pi-hole blocks queries to known advertising, tracking, and malicious domains, thereby significantly enhancing the privacy of users connected to the router’s Wi-Fi network. Custom blocklists sourced from the Cert.pl and Firebog.net repositories were also incorporated, enabling precise adaptation of the system to current threats (Figure 7). Unbound was deployed as the recursive DNS layer, functioning as a fully independent resolver that retrieves data directly from root DNS servers, thereby eliminating the need to contact any third-party intermediaries. As a result, each DNS query is processed locally and independently, improving both privacy and response speed (Figure 8).

The next step involved the installation and advanced configuration of the DNSCrypt-proxy client, which introduced an additional layer of DNS query encryption and anonymization. The DNSCrypt, DNS over HTTPS (DoH), and Oblivious DoH (ODoH) protocols were enabled, along with enforced DNSSEC support. DNS query routing through the TOR network via a local

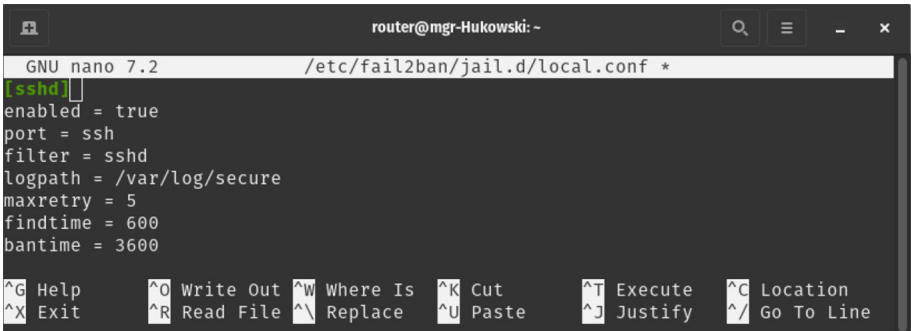


Figure 3. Configuration of the fail2ban intrusion prevention system

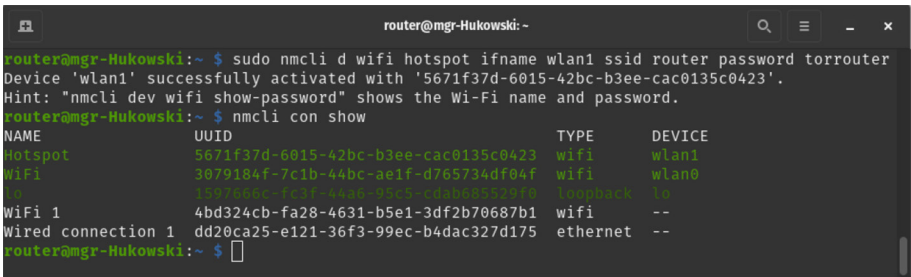
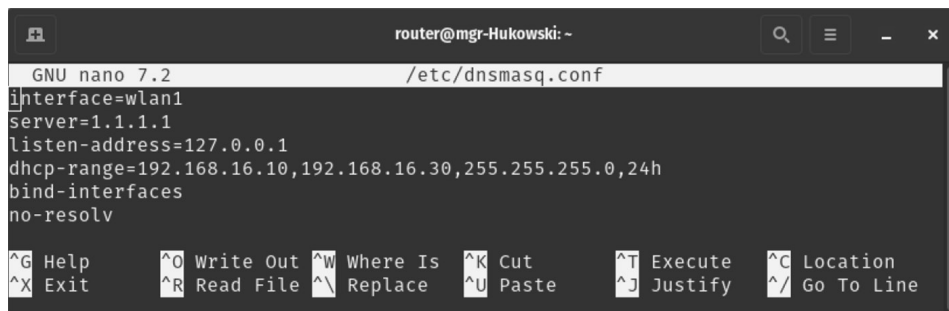


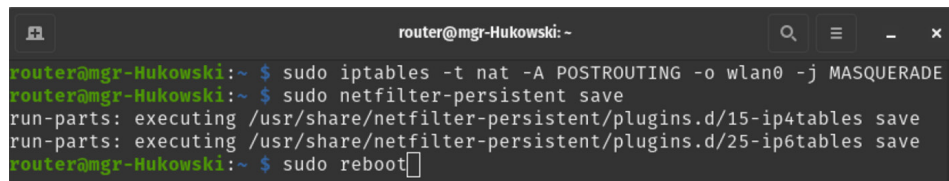
Figure 4. Network interface configuration



```

router@mgr-Hukowski: ~
GNU nano 7.2 /etc/dnsmasq.conf
interface=wlan1
server=1.1.1.1
listen-address=127.0.0.1
dhcp-range=192.168.16.10,192.168.16.30,255.255.255.0,24h
bind-interfaces
no-resolv
^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^_ Replace   ^U Paste     ^J Justify   ^_ Go To Line
  
```

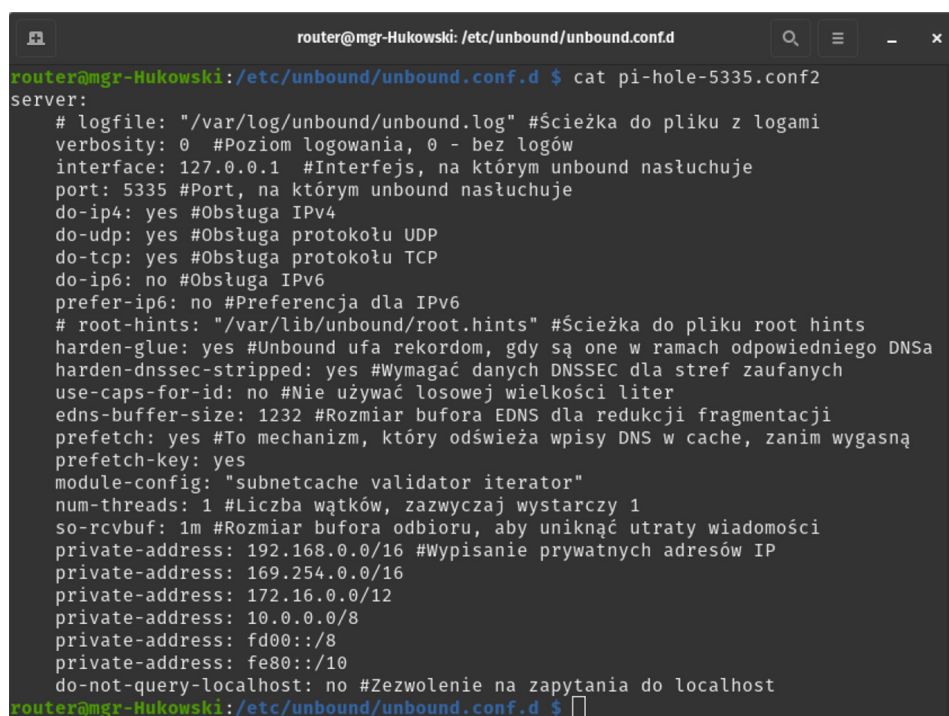
Figure 5. The dnsmasq configuration file



```

router@mgr-Hukowski: ~
router@mgr-Hukowski:~$ sudo iptables -t nat -A POSTROUTING -o wlan0 -j MASQUERADE
router@mgr-Hukowski:~$ sudo netfilter-persistent save
run-parts: executing /usr/share/netfilter-persistent/plugins.d/15-ip4tables save
run-parts: executing /usr/share/netfilter-persistent/plugins.d/25-ip6tables save
router@mgr-Hukowski:~$ sudo reboot
  
```

Figure 6. Firewall configuration for non-TOR operational modes



```

router@mgr-Hukowski: /etc/unbound/unbound.conf.d
router@mgr-Hukowski:/etc/unbound/unbound.conf.d$ cat pi-hole-5335.conf2
server:
# logfile: "/var/log/unbound/unbound.log" #Ścieżka do pliku z logami
verbosity: 0 #Poziom logowania, 0 - bez logów
interface: 127.0.0.1 #Interfejs, na którym unbound nasłuchuje
port: 5335 #Port, na którym unbound nasłuchuje
do-ip4: yes #Obsługa IPv4
do-udp: yes #Obsługa protokołu UDP
do-tcp: yes #Obsługa protokołu TCP
do-ip6: no #Obsługa IPv6
prefer-ip6: no #Preferencja dla IPv6
# root-hints: "/var/lib/unbound/root.hints" #Ścieżka do pliku root hints
harden-glue: yes #Unbound ufa rekordom, gdy są one w ramach odpowiedniego DNSa
harden-dnssec-stripped: yes #Wymagać danych DNSSEC dla stref zaufanych
use-caps-for-id: no #Nie używać losowej wielkości liter
edns-buffer-size: 1232 #Rozmiar bufora EDNS dla redukcji fragmentacji
prefetch: yes #To mechanizm, który odświeża wpisy DNS w cache, zanim wygasną
prefetch-key: yes
module-config: "subnetcache validator iterator"
num-threads: 1 #Liczba wątków, zazwyczaj wystarczy 1
so-rcvbuf: 1m #Rozmiar bufora odbioru, aby uniknąć utraty wiadomości
private-address: 192.168.0.0/16 #Wypisanie prywatnych adresów IP
private-address: 169.254.0.0/16
private-address: 172.16.0.0/12
private-address: 10.0.0.0/8
private-address: fd00::/8
private-address: fe80::/10
do-not-query-localhost: no #Zezwolenie na zapytania do localhost
router@mgr-Hukowski:/etc/unbound/unbound.conf.d$
  
```

Figure 7. Pi-hole configuration with the Unbound server

SOCKS5 proxy was also implemented. The system was equipped with advanced features including DNS caching and query routing through anonymous relay servers. The combination of Pi-hole, Unbound, and DNSCrypt-proxy provided the router with a sophisticated DNS protection system that minimizes the risk of leaks and eavesdropping while ensuring full control over DNS query routing.

TOR network integration

To ensure maximum user anonymity, the subsequent phase involved implementing a mechanism for tunneling all network traffic through the TOR network. The TOR service was installed from the system repositories and configured as a proxy server. The critical entries in the torrc configuration file defined the TransPort and SocksPort

```

router@mgr-Hukowski: ~
router@mgr-Hukowski:~$ cat /etc/unbound/unbound.conf.d/pi-hole-DoH-443.conf | grep
-v "# "
server:
  verbosity: 0 #Poziom logowania, 0 - bez logów
  interface: 127.0.0.1@443 #Interfejs, na którym unbound nasłuchuje
  do-ip4: yes #Obsługa IPv4
  do-udp: yes #Obsługa protokołu UDP
  do-tcp: yes #Obsługa protokołu TCP
  do-ip6: no #Obsługa IPv6
  prefer-ip6: no #Preferencja dla IPv6
  harden-glue: yes #Unbound ufa rekordom, gdy są one w ramach odpowiedniego DNSa
  harden-dnssec-stripped: yes #Wymagać danych DNSSEC dla stref zaufanych
  use-caps-for-id: no #Nie używać losowej wielkości liter
  edns-buffer-size: 1232 #Rozmiar bufora EDNS dla redukcji fragmentacji
  prefetch: yes #To mechanizm, który odświeża wpisy DNS w cache, zanim wygasną
  prefetch-key: yes
  module-config: "subnetcache validator iterator"
  num-threads: 1 #Liczba wątków, zazwyczaj wystarczy 1
  so-rcvbuf: 1m #Rozmiar bufora odbioru, aby uniknąć utraty wiadomości
  private-address: 192.168.0.0/16 #Wypisanie prywatnych adresów IP
  private-address: 169.254.0.0/16
  private-address: 172.16.0.0/12
  private-address: 10.0.0.0/8
  private-address: fd00::/8
  private-address: fe80::/10
  do-not-query-localhost: no #Zezwolenie na zapytania do localhost
  tls-cert-bundle: "/etc/ssl/certs/ca-certificates.crt"
  tls-service-key: "/etc/unbound/ssl/key.pem"
  tls-service-pem: "/etc/unbound/ssl/cert.pem"
  https-port: 443
router@mgr-Hukowski:~$

```

Figure 8. Pi-hole configuration with Unbound (DoH mode)

```

router@mgr-Hukowski: ~
router@mgr-Hukowski:~$ sudo nano /etc/tor/torrc
router@mgr-Hukowski:~$ cat /etc/tor/torrc | grep -v '#' | grep '.'
Log notice file /var/log/tor/notices.log
Log debug file /var/log/tor/debug.log
VirtualAddrNetworkIPv4 10.192.0.0/10
AutomapHostsSuffixes .onion,.exit
AutomapHostsOnResolve 1
TransPort 0.0.0.0:9040
DNSPort 0.0.0.0:9053
SocksPort 0.0.0.0:9050
ControlPort 9051
HashedControlPassword 16:F5E718707688C11F6067F466846FDAECCE2A39E2D5F6A9ACE19331A
740
CookieAuthentication 1
ExcludeNodes {Ru} StrictNodes 1
ExcludeExitNodes {Pl},{Cn} StrictNodes 1
router@mgr-Hukowski:~$ sudo systemctl restart tor

```

Figure 9. TOR service configuration

parameters, enabling traffic redirection at both the system and application levels (Figure 9).

Appropriate iptables rules were configured to redirect all incoming traffic from the wlan1 interface to port 9040 (TOR TransPort). This ensures that all packets leaving the local network are automatically anonymized without requiring TOR software installation on client devices. The configuration also incorporated the option to exclude TOR nodes located in countries with low trust levels through the ExcludeNodes and ExcludeExitNodes directives in the configuration file (Figure 10). To ensure persistence of the redirection

rules, the netfilter-persistent utility was employed, enabling automatic loading of the iptables configuration at each system startup. The entire setup was validated using diagnostic scripts, including DNS leak tests, which confirmed effective tunneling operation and the absence of direct Internet connections outside the TOR network (Figure 11).

Extended functionality

In the final development phase, a set of features enhancing the interactivity and usability of the router was implemented. A 128 × 128 pixel

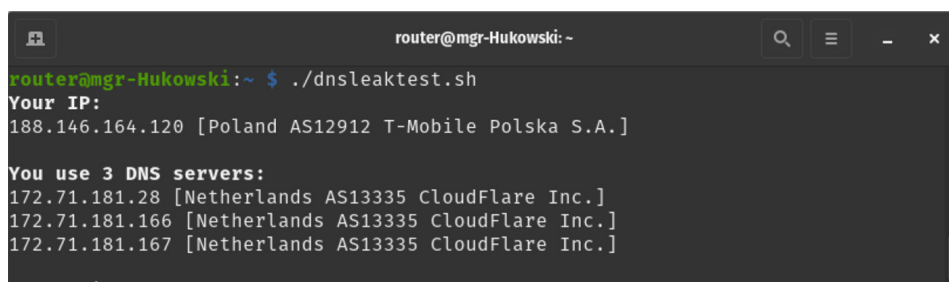


```

router@mgr-Hukowski:~$ cat iptables_rules.tor | grep -v "#"
sudo iptables -F
sudo iptables -t nat -F
sudo iptables -t nat -A POSTROUTING -o wlan0 -j MASQUERADE
sudo iptables -t nat -A PREROUTING -i wlan1 -p tcp --dport 22 -j REDIRECT --to-ports 22
sudo iptables -t nat -A PREROUTING -i wlan1 -p udp --dport 53 -j REDIRECT --to-ports 53
sudo iptables -t nat -A PREROUTING -i wlan1 -p tcp --dport 53 -j REDIRECT --to-ports 53
sudo iptables -t nat -A PREROUTING -i wlan1 -p tcp --syn -j REDIRECT --to-ports 9040
sudo iptables -A OUTPUT -o lo -j ACCEPT
sudo iptables -A INPUT -i lo -j ACCEPT
sudo iptables -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
sudo iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
sudo iptables -t nat -A OUTPUT -p tcp -m owner --uid-owner debian-tor -j RETURN
sudo iptables -t nat -A OUTPUT -p tcp -d 127.0.0.0/8 -j RETURN
sudo iptables -t nat -A OUTPUT -p tcp -d 192.168.0.0/16 -j RETURN
sudo iptables -t nat -A OUTPUT -p tcp -d 10.0.0.0/8 -j RETURN
sudo iptables -t nat -A OUTPUT -p tcp -d 172.16.0.0/12 -j RETURN
sudo iptables -t nat -A OUTPUT -p tcp --syn -j REDIRECT --to-ports 9040
router@mgr-Hukowski:~$

```

Figure 10. Firewall configuration for TOR-enabled operational modes



```

router@mgr-Hukowski:~$ ./dnsleaktest.sh
Your IP:
188.146.164.120 [Poland AS12912 T-Mobile Polska S.A.]

You use 3 DNS servers:
172.71.181.28 [Netherlands AS13335 CloudFlare Inc.]
172.71.181.166 [Netherlands AS13335 CloudFlare Inc.]
172.71.181.167 [Netherlands AS13335 CloudFlare Inc.]

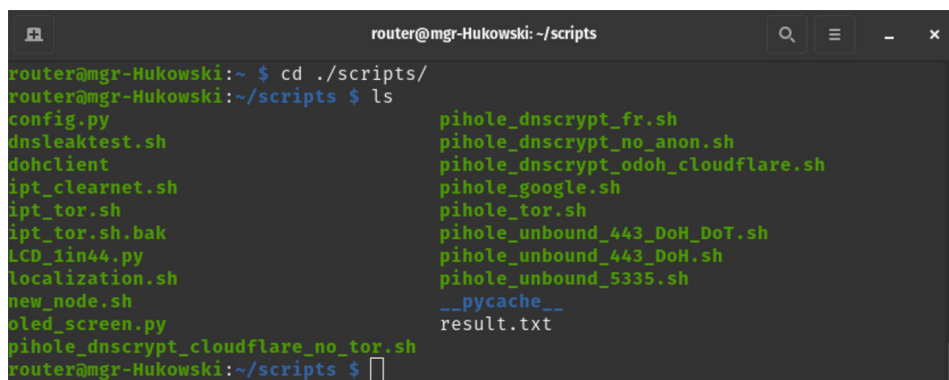
```

Figure 11. DNS leak test utility output

OLED display and a set of control buttons were connected to the Raspberry Pi, enabling physical management of the device's operational modes. After enabling the SPI interface and installing the requisite Python libraries (including I2C, spidev, and PIL), a graphical interface was programmed that displays system data — such as the IP address, CPU temperature, and the number of connected clients — and enables switching between the Clearnet and TOR modes (Figure 12). Additionally, a suite of automation scripts was implemented. These scripts handle rapid iptables

rule switching, start and stop DNSCrypt and Pi-hole services, and perform diagnostics of the current router state. Supplementary tools, such as the dnsleaktest.sh script, enable quick verification of DNS tunneling correctness and detection of potential leaks (Figure 13).

These extensions enabled the creation of a fully functional, flexible, and secure network system that not only protects user privacy but also allows intuitive management via a physical interface without requiring system login. The system was optimized for continuous 24/7 operation.



```

router@mgr-Hukowski:~/scripts$ ls
config.py                               pihole_dnscrypt_fr.sh
dnsleaktest.sh                          pihole_dnscrypt_no_anon.sh
dohclient                               pihole_dnscrypt_odoh_cloudflare.sh
ipt_clearnet.sh                         pihole_google.sh
ipt_tor.sh                             pihole_tor.sh
ipt_tor.sh.bak                         pihole_unbound_443_DoH_DoT.sh
LCD_iin44.py                           pihole_unbound_443_DoH.sh
localization.sh                        pihole_unbound_5335.sh
new_node.sh                            __pycache__
oled_screen.py                         result.txt
pihole_dnscrypt_cloudflare_no_tor.sh
router@mgr-Hukowski:~/scripts$

```

Figure 12. Python scripts for router control via the OLED display

```
router@mgr-Hukowski: ~/scripts
router@mgr-Hukowski:~/scripts $ cat pihole_dnscrypt_fr.sh
#!/bin/bash
NEW_DNS_SERVER="127.0.0.1#54"
SETUP_VARS_PATH="/etc/pihole/setupVars.conf"
DNS_VAR="PIHOLE_DNS_1"
SERVER_NAMES_NEW="doh-crypto-sx"
SERVER_NAMES=$(cat /opt/dnscrypt-proxy/dnscrypt-proxy.toml | grep ^server_names | cut -d '"' -f2)
PORT=$(cat /etc/pihole/setupVars.conf | grep "#" | cut -d '=' -f2)
IPT=$(sudo iptables -S -t nat | wc -l)
a=0
if [ "$IPT" == "5" ]; then
    sudo /home/router/scripts/ipt_tor.sh
    echo "Iptables zaaktualizowany"
fi
if [ "$SERVER_NAMES" != "$SERVER_NAMES_NEW" ]; then
    sudo cp /opt/dnscrypt-proxy/dnscrypt-proxy-fr.toml2 /opt/dnscrypt-proxy/dnscrypt-proxy.toml
    echo "DNSCrypt-proxy zaaktualizowany"
fi
if [ "$PORT" != "$NEW_DNS_SERVER" ]; then
    if grep -q "^$DNS_VAR=" "$SETUP_VARS_PATH"; then
        sudo sed -i "s/^$DNS_VAR=.*/$DNS_VAR=$NEW_DNS_SERVER/" "$SETUP_VARS_PATH"
    else
        sudo echo "$DNS_VAR=$NEW_DNS_SERVER" >> "$SETUP_VARS_PATH"
    fi
    echo "Pi-hole zaaktualizowany"
fi
if [ "$SERVER_NAMES" != "$SERVER_NAMES_NEW" ]; then
    sudo systemctl restart dnscrypt-proxy.service
    a=$((a+1))
fi
if [ "$PORT" != "$NEW_DNS_SERVER" ]; then
    pihole -a interface
    a=$((a+1))
fi
if [ "$a" != "0" ]; then
    sleep 3
    /home/router/scripts/localization.sh
fi
exit 0
```

Figure 13. Example Python script for the Pi-hole + TOR + DNSCrypt + relay mode

RESULTS

The router was equipped with a dynamic mode-switching mechanism, allowing adjustment of the privacy level, performance, and network security to the user’s current requirements. The router offers two fundamental operating regimes — Clearnet and TOR — each augmented with various DNS configurations, encryption protocols, and anonymization mechanisms.

Each operational mode represents a different trade-off between security and privacy on the one hand, and usability and system performance on the other. Configurations based solely on Pi-hole and public DNS servers are straightforward to implement but do not provide sufficient protection in environments susceptible to surveillance. Deployments incorporating Unbound, DNSCrypt, and TOR — particularly with ODoH and relays — provide a very high level of privacy but require advanced

Table 2. Comparison of router operational modes

Mode	DNS encryption	Anonymization	Filtering	Config. complexity
Pi-hole + Google DNS	No	No	Yes	Low
Pi-hole + Unbound	No	No	Yes	Medium
Pi-hole + Unbound DoH	Partial (to Unbound)	No	Yes	Medium
Pi-hole + Unbound → Cloudflare (DoT)	Yes	No	Yes	Medium
Pi-hole + DNSCrypt + Relay	Yes	Partial	Yes	Medium
Pi-hole + TOR	Yes (to TOR node)	Full	Yes	Medium
Pi-hole + DNSCrypt + TOR	Yes	Full	Yes	High
Pi-hole + DNSCrypt + TOR + Relay	Yes	Full+	Yes	High
Pi-hole + DNSCrypt + TOR + ODoH	Yes	Full+	Yes	High

Note: DNS encryption — No = no encryption; Partial = encryption on selected segments only; Yes = full encryption. Anonymization — No = none; Partial = IP concealment from the DNS server; Full = IP concealment via TOR; Full+ = TOR combined with an additional anonymization layer. Filtering — Yes = active blocking of advertisements and trackers via Pi-hole. Configuration complexity — Low = basic installation; Medium = requires service configuration; High = advanced multi-service configuration.

technical knowledge and more capable hardware. The modular architecture of the router enables seamless switching between modes, rendering it a versatile tool for both casual users and advanced users seeking maximum anonymity. A comparison of the developed configurations is presented in Table 2. Modes relying exclusively on Pi-hole represent a compromise between ease of deployment and basic protection, while configurations incorporating full TOR tunneling and multi-layer DNS encryption offer the maximum achievable level of anonymization, albeit requiring advanced technical expertise for proper implementation.

Performance evaluation was conducted using the DNS Performance Testing Tool (dnssperf) version 2.9.0, which enables measurement of key DNS server operational parameters under controlled load conditions. Tests were executed from the command line using the following command: `dnssperf -s 192.168.16.1 -p 53 -d queries.txt`. The input dataset, contained in the queries.txt file, comprised 77 DNS queries for records of various types (A, AAAA, MX, NS, TXT, SOA, SRV) targeting a set of domains representative of both global Internet services and specialized network resources. The tested domains included, among others, example.com, google.com, facebook.com, youtube.com, amazon.com, and cloudflare.com, as well as ftp.gnu.org, crates.io, and cdn.jsdelivr.net (the complete list of domains is provided in Appendix A). The sampling was purposive, designed to

reflect the diverse nature of real-world DNS queries encountered in a production environment.

The experiment encompassed eight distinct DNS server configurations, for which the following metrics were measured: average response latency, minimum and maximum latency, and standard deviation of latency. Each configuration was subjected to four independent trials, with the DNS cache being cleared prior to each measurement to ensure repeatability and eliminate the effect of cached queries. Additionally, using dnssperf, the system throughput was assessed by determining the number of queries handled per second (QPS), as well as the name resolution success rate, expressed as the percentage of queries completed successfully. All raw data were recorded and subsequently subjected to statistical analysis aimed at identifying significant performance differences among the individual DNS server configurations.

The results of the conducted measurements indicate that the analyzed DNS server configurations exhibit noticeable yet generally moderate differences in response latencies. As shown in Figure 14, the Pi-hole + Google DNS configuration achieves the lowest average latency with a relatively narrow range of minimum and maximum values, confirming its stable and predictable temporal characteristics. Configurations of greater complexity, such as Pi-hole + Unbound (DoH) + Cloudflare (DoT) or those involving DNSCrypt and the TOR network, generate somewhat higher average latencies and a wider value

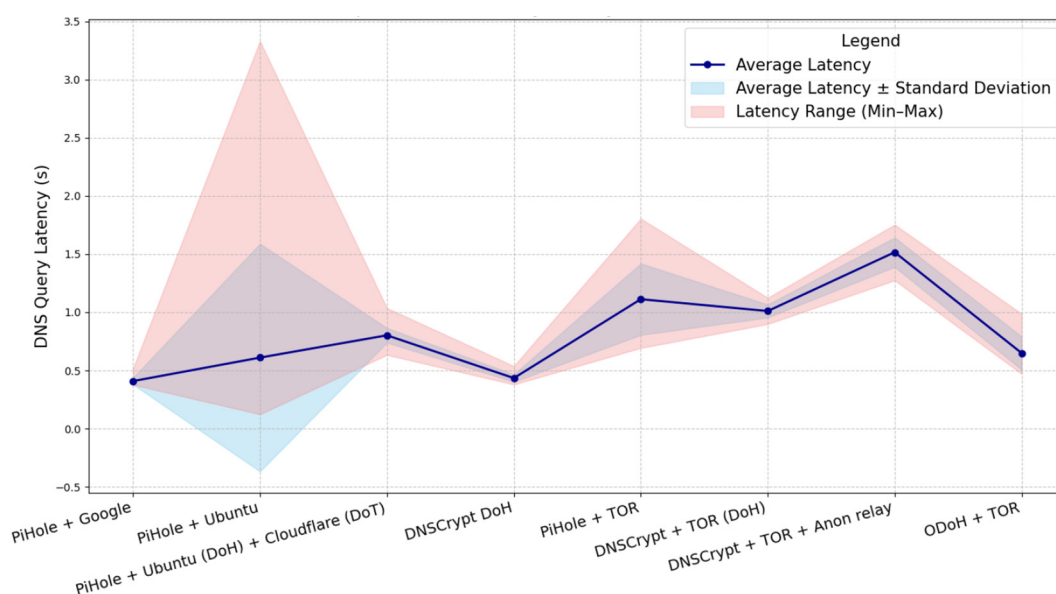


Figure 14. Comparison of DNS response latencies across operational modes (average, minimum, and maximum values)

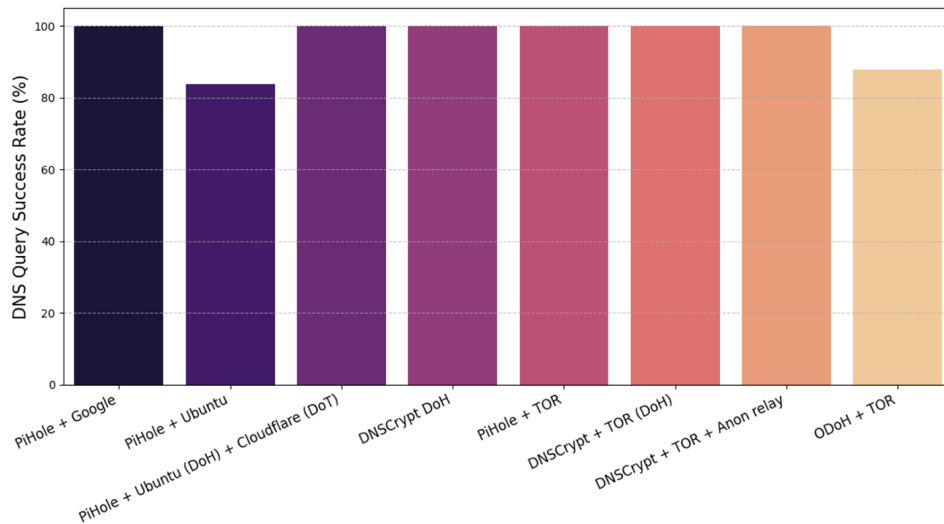


Figure 15. DNS query resolution success rate across operational modes

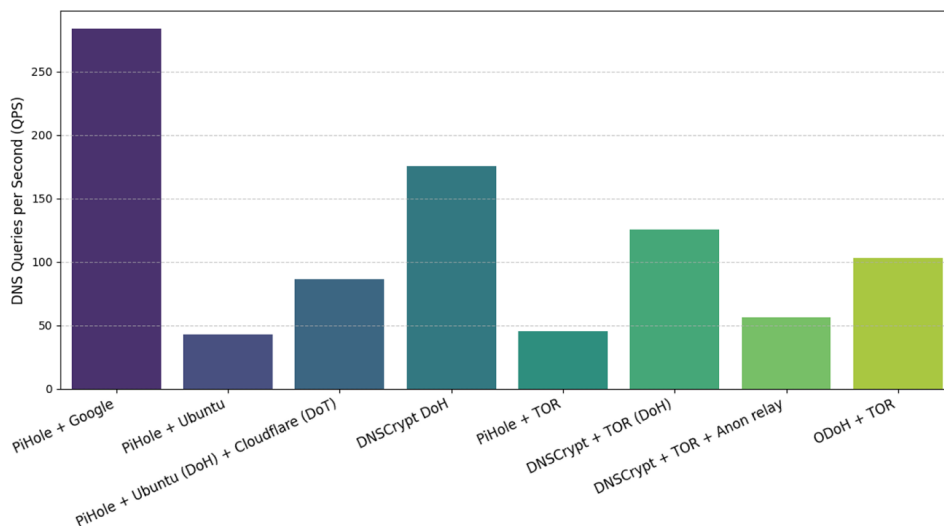


Figure 16. DNS query throughput (queries per second) across operational modes

range, which can be attributed to the additional overhead arising from encryption and multi-layer routing. However, this increase remains limited — in absolute terms, the differences are relatively small, suggesting that even when employing more advanced privacy and security mechanisms, DNS response times remain acceptable from the perspective of a typical end user. In terms of DNS name resolution effectiveness, all tested configurations maintained a very high percentage of successfully resolved queries, as illustrated in Figure 15. Minor decreases in success rate observed for certain configurations (e.g., Pi-hole + Unbound and ODoH + TOR) do not indicate significant reliability issues but rather reflect the inherent limitations associated with more complex traffic processing and potential sensitivity to timeouts.

Throughput analysis, measured as the number of DNS queries serviced per second (QPS) and presented in Figure 16, demonstrates that the highest QPS values are achieved by configurations with lower cryptographic complexity, such as Pi-hole + Google DNS and DNSCrypt DoH. Configurations employing TOR — particularly in combination with additional anonymization mechanisms — exhibit lower throughput; however, this decrease falls within the expected range of performance cost for enhanced privacy. Taken together, the results indicate that the more advanced methods, despite measurable overhead in terms of latency and QPS, maintain operating parameters at a practically useful level, offering an attractive compromise between privacy protection and DNS system performance.

CONCLUSIONS

This paper has presented the design and implementation of a multi-layer privacy-preserving router based on the Raspberry Pi 5 platform, integrating network traffic anonymization technologies (TOR), advanced DNS encryption protocols (DNSCrypt, DoH, DoT, ODoH), the Unbound local recursive resolver, and the Pi-hole filtering system. The developed solution offers nine operational modes enabling dynamic adjustment of the privacy protection level to the user's current needs — from basic advertisement filtering to full anonymization with multi-layer DNS query encryption.

It has been demonstrated that the Raspberry Pi 5 equipped with an ARM Cortex-A76 processor (4×2.4 GHz) and 8 GB of RAM is capable of simultaneously running all implemented security mechanisms. The successful integration of Pi-hole, Unbound, DNSCrypt-proxy, and TOR was confirmed across nine distinct operational configurations. A critical success factor proved to be the use of an NVMe SSD connected via the PCIe interface, which eliminated the I/O bottleneck characteristic of microSD cards.

The comparative analysis of operational modes revealed a correlation between the level of privacy protection and system performance parameters. The basic mode (Pi-hole with Google Public DNS) exhibits negligible impact on network performance. The introduction of the local Unbound resolver increases system resource utilization but eliminates dependence on external DNS providers. Configurations employing full traffic tunneling through the TOR network with multi-layer DNS encryption (DNSCrypt + ODoH) impose the highest computational demands, resulting in a noticeable increase in DNS response latencies. The throughput reduction in TOR mode is an inherent characteristic of the onion routing architecture and does not stem from hardware limitations of the Raspberry Pi 5 platform. A particularly noteworthy configuration proved to be the combination of DNSCrypt-proxy with the TOR network, which effectively eliminates the DNS leak problem at the exit node.

The developed system fully realizes the modularity assumption. The implementation of a physical control interface (128×128 px OLED display with push buttons) enables mode switching without the need for SSH access or command-line knowledge. The developed automation scripts

manage the reconfiguration process — stopping unnecessary services, modifying iptables rules, and launching the appropriate components for the selected mode. The switching time between modes is sufficiently short to render the system practical for everyday use.

REFERENCES

1. Freedom House. Freedom on the Net 2023: The Repressive Power of Artificial Intelligence. Washington, DC: Freedom House; 2023. Available from: <https://freedomhouse.org/report/freedom-net/2023/repressive-power-artificial-intelligence>
2. Electronic Frontier Foundation. Behind the One-Way Mirror: A Deep Dive Into the Technology of Corporate Surveillance. San Francisco, CA: EFF; 2019. Available from: <https://www.eff.org/wp/behind-the-one-way-mirror>
3. Gartner Inc. Leading the IoT: Gartner Insights on How to Lead in a Connected World. Stamford, CT: Gartner; 2017.
4. The Tor Project. Tor Metrics Portal. 2024. Available from: <https://metrics.torproject.org/>
5. Greschbach B., Pulls T., Roberts L.M., Winter P., Feamster N. The effect of DNS on Tor's anonymity. In: Proceedings of the Network and Distributed System Security Symposium (NDSS). San Diego, CA, USA; 2017. <https://doi.org/10.14722/ndss.2017.23456>
6. Dingledine R., Mathewson N., Syverson P. Tor: The second-generation onion router. In: Proceedings of the 13th USENIX Security Symposium. San Diego, CA, USA; 2004. p. 303–320.
7. Mockapetris P. Domain names – concepts and facilities. STD 13, RFC 1034. Internet Engineering Task Force; 1987. <https://doi.org/10.17487/RFC1034>
8. Mockapetris P. Domain names – implementation and specification. STD 13, RFC 1035. Internet Engineering Task Force; 1987. <https://doi.org/10.17487/RFC1035>
9. Arends R., Austein R., Larson M., Massey D., Rose S. DNS security introduction and requirements. RFC 4033. Internet Engineering Task Force; 2005. <https://doi.org/10.17487/RFC4033>
10. Hu Z., Zhu L., Heidemann J., Mankin A., Wessels D., Hoffman P. Specification for DNS over transport layer security (TLS). RFC 7858. Internet Engineering Task Force; 2016. <https://doi.org/10.17487/RFC7858>
11. Hoffman P., McManus P. DNS Queries over HTTPS (DoH). RFC 8484. Internet Engineering Task Force; 2018. <https://doi.org/10.17487/RFC8484>

12. Kinnear E., McManus P., Pauly T., Verma T., Wood C. Oblivious DNS over HTTPS. RFC 9230. Internet Engineering Task Force; 2022. <https://doi.org/10.17487/RFC9230>
13. Huitema C., Dickinson S., Mankin A. DNS over Dedicated QUIC Connections. RFC 9250. Internet Engineering Task Force; 2022. <https://doi.org/10.17487/RFC9250>
14. Edmundson A., Ensafi R., Feamster N., Rexford J. A security analysis of the Safeplug ‘Tor in a Box’. Technical Report. Princeton University; 2014.
15. The Tor Project. Tor Proposal 352: DNS Resolution in Tor. 2023. Available from: <https://gitlab.torproject.org/tpo/core/torspec/-/blob/main/proposals/352-dns-resolution.md>
16. Winter P., Lindskog S., Pulls T. How Do Tor Users Interact With Onion Services? In: Proceedings of the 27th USENIX Security Symposium. Baltimore, MD, USA; 2018. p. 411–428.
17. Dooley M., Rooney T. DNS Security Management. Hoboken, NJ: IEEE Press/Wiley; 2017. <https://doi.org/10.1002/9781119328414>