applied sciences

MDPI

Article

Enhancing the Privacy of Network Services through Trusted Computing

Denghui Zhang [1,2], Lijing Ren [2] and Zhaoquan Gu [2,3,*]

[1] Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou 510006, China
[2] Department of New Networks, Peng Cheng Laboratory, Shenzhen 518055, China
[3] Department of Computer Science and Technology, Harbin Institute of Technology, Shenzhen 518055, China
* Correspondence: guzhaoquan@hit.edu.cn

Abstract: The addressing and discovering service is a vital infrastructure of the Internet. New applications and scenarios in next-generation networks rely on the secure and stable operation of domain name services, which puts forward new security challenges for the original domain name mechanism. While previous security enhancements of network services struggled to strike a balance between security, performance, and compatibility, hindering further use of core network services, the TEE (Trusted Computing Environment) technology can provide trusted and confidential services in untrusted network environments by verifiable hardware signatures. In this paper, we present a novel trustworthy service architecture with the preservation of security and privacy for addressing messages. The scheme provides a secure enclave to generate authenticatable responses between clients and targets, thus ensuring the privacy of services. We further build a new TEE compilation model to ensure that the built resolver application can provide trusted and secure services within TEE while keeping the availability without the TEE hardware. Experimental results show that our approach can enhance the privacy and security of addressing services such as DNS (Domain Name System) without sacrificing the quality of service and breaking the infrastructures of existing services.

Keywords: privacy; TEE; compatibility; DNS; digital signature

Citation: Zhang, D.; Ren, L.; Gu, Z. Enhancing the Privacy of Network Services through Trusted Computing. *Appl. Sci.* **2022**, *12*, 9191. https://doi.org/10.3390/app12189191

Academic Editors: Hiroyuki Sato and David Megías

Received: 25 July 2022
Accepted: 2 September 2022
Published: 14 September 2022

1. Introduction

With the advent of the Internet and next-generation networks, traditional network services are playing a new role in emerging scenarios such as IoT (Internet of Things) and IoV (Internet of Vehicles) [1–3]. As the infrastructure of the Internet, DNS is playing more roles, including: (i) domain name resolution services; (ii) routing and load balancing for the application layers of the Internet; (iii) verification for mail servers; (iv) discovering and addressing services for digital devices [4,5]. However, the proliferation of connected devices has led to increased opportunities for data and identity leakage due to the coverage of new embedded technologies and interconnected networking.

As a fundamental and widely used Internet service, discovering and addressing services such as DNS and Multicast DNS (mDNS) have to run properly to prevent communication from collapsing. Once basic services such as DNS are attacked, the entire Internet will be affected [6–8]. These services are not only an address translation service but also provide authentication and improve security services for Internet applications. However, current services are concentrated on availability and ignore security [9–11]. The vulnerabilities of services can be exploited to attack network systems and collapse all live hosts and applications [12,13]. When being collected by an eavesdropper with pervasive monitoring, the plaintext information carried in transactions can lead to serious threats to privacy. Attackers may lead applications to fake servers controlled by hackers and commit illegal acts such as phishing fraud and sensitive data leakage.

The massive amount of data composed of various traces of Internet applications provides worthy attack targets for big data analysis technology. The machine learning

algorithm may be utilized to further analyze the application's behaviors according to logs. Data that does not involve sensitive information, such as DNS query records, may also lead to serious data leakage accidents through machine learning methods such as neural networks and feature learning [14]. Another practice challenge is that they often run in an untrusted cloud environment, such as Cloudflare and Google, so the client is not sure whether the resolved result is trusted. Cloud services turn out to be vulnerable to outside threats or privileged hypervisors [15]. Users cannot prevent servers from reserving or leaking communicated messages. Because of the complexity of applications and limitations in systems, data security based on software alone often fails [16,17].

To address the above issues, we devise the usage of TEE [18] as an important way to enhance the security and privacy of the data and execution in network services. The basic idea is to use Intel Software Guard Extensions (SGX) [19] to implement a trusted service called DoTT (Domain service over Trusted Transport Layer Security). DoTT establishes a trusted TLS connection between the addressing client and resolver, which runs in a secure container to prevent eavesdropping and snooping. We further reconstruct the build process of the resolver application to make it run on both the TEE and normal environments for backward compatibility. The contributions of our paper are listed as follows:

1. To enhance the security of DNS, we design a privacy-preserving and trustworthy addressing and discovering service based on TEE technology.
2. To deal with the hardware limitations of SGX and reduce the cost of adopting the new framework, we build a novel compilation model for the widespread deployment of trusted services and the correctness of refactored code.
3. To evaluate compatibility and compare our scheme with existing improvements, we implement a prototype system using Intel SGX. Experimental results show that the proposed DoTT can effectively maintain the low latency of DNS when preserving privacy.

We arrange the rest of the paper as follows. To begin with, we will provide backgrounds and related work on Internet services and TEE in Section 2. In Section 3, we propose a novel privacy-preserving and back-compatible architecture to run network services in untrusted cloud environments. Section 4 evaluates compatibility experiments and compares our scheme with existing improvements. Finally, we give a conclusion in Section 5.

2. Backgrounds and Related Work

2.1. The Addressing and Discovering Service

When an application requests a website or a networking service by a domain name, the application first sends a query request to the recursive resolver. If the resolver caches the appropriate IP address, it will forward the address directly to the application. Otherwise, the resolver queries other name servers (NS) such as root name servers, top-level domain name servers (TLD), and authoritative name servers, as well as hierarchical trees. The iterative queries continue until it retrieves a DNS record containing the requested IP address, which will be forwarded back to the requester in turn. The application can request the actual webpage once the DNS lookup returns the IP address for a queried domain name.

The security of the DNS infrastructure is one of the core requirements for Internet services. To address the drawbacks of DNS, many improvements and extensions follow [12,14]. DNS evolved from a simple name-IP conversion service to a complex and secure resolution service. Table 1 presents the feature comparison of existing methods. The plain DNS denotes the normal DNS protocol. If the performance of the new mechanism is on the same order of magnitude as plain DNS when handling DNS requests, we think it has performance advantages. Deployability denotes deployment friendliness; that is, whether this method needs to update or migrate the existing DNS software. If the proposed DNS scheme is compatible with the existing technologies, such as TLS and HTTPS, we think it is easy to deploy. If one scheme can provide integrity and origin authentication for DNS data, we consider it to have authentication features. The security of the DNS

infrastructure is one of the core requirements for Internet services. The breach in DNS security and privacy will affect the trust-worthiness of the Internet in turn. The DNSSEC scheme [8] is first proposed to complement the original protocol for DNS data integrity and origin authentication. The Internet Engineering Task Force (IETF) further proposes DNS-over-HTTPS (DoH) and DNS-over-TLS (DoT) to standardize DNS encryption and protect DNS from eavesdropping and manipulation of DNS data. With increasing focus on privacy and iterative updating of the software and system, DNS clients supporting DoT and DoH are increasing. However, two significant drawbacks remain. The only score DoTT did not achieve against the plain DNS was performance, which is an essential cost of securing a system. On the other hand, the prototype system we implemented only utilizes SGX features without further optimization. We will discuss more details about DoTT in Section 3.

Table 1. Feature comparisons among previous DNS schemes.

Scheme	Privacy	Authentication	Performance	Deployability
Plain DNS	✗	✗	✓	✓
DNSSEC	✗	✓	✗	✓
DNSCrypt/DNSCurve	✓	✓	✗	✓
ConfidentialDNS	✓	✓	✗	✗
DNS-over-TLS	✓	✗	✗	✓
DNS-over-HTTPS	✓	✗	✗	✓
DoTT (Our)	✓	✓	✗	✓

The first is software vulnerabilities. As mass devices need cross-domain communication, the Internet needs perfect resource addressing technology to ensure that the related information can be efficiently addressed, located, and queried. In the original design and still dominant use of DNS, queries are sent in plaintext. The metadata in a DNS query contains sensitive information, including accessed hostname, websites, as well as IP addresses identifying personal devices. This means that anyone on the network path between personal devices and DNS resolvers can see the query. The breach in DNS security and privacy will affect the trust-worthiness of the Internet in turn. The resource records are not signed and anti-counterfeiting protected. Therefore, the DNS protocol is vulnerable to caching and malicious attacks such as poisoning, eavesdropping, tampering, etc. Further, it may reveal significant information about users and result in a serious threat to individual privacy [13]. The resolver can still associate all queries with the customer's sensitive data. There is a great challenge to ensure the integrity of DNS programs and the reliability of responses. The second is backward compatibility. A large spectrum of improved systems and architectures for DNS have been proposed without considering the compatibility with existing systems. Hence, they are difficult to be adopted by network operators. The use of secured protocols comes with problems, including compromising conventional network architectures. For example, 10 years after DNSSEC was put forward, it has not been deployed on a large scale. The deployment of DNSSEC is only 3% of the total in the second-level domain [12].

DNSSEC [20] is the earliest DNS security protocol to reduce the risk of a man-in-the-middle attack and ensure the integrity of DNS data by issued certificates. Since DNS has not considered the security mechanism in the domain name resolution, the extension gives resolvers the ability to perform authentication and data integrity checks on the replies. The root service of the Internet domain name started to deploy the DNSSEC service in 2010, which marked the development of the domain name service to the security service. Full deployment of DNSSEC ensures that the end-user connects to the actual website or another service that corresponds to a specific domain name. However, we have to deploy this technology at every step of the lookup, from the root servers to the final authoritative servers. Hence, its current usage is still low. Moreover, DNSSEC simply verifies that the

resolved address we are requesting is valid but does not encrypt data. The vulnerability may be exploited by hackers to collect information about users.

DNSCurve [21] is another security extension of DNS based on the existing DNS architecture. It uses the more efficient elliptic curve encryption algorithm to reduce the computational cost for encryption and cryptographically authenticates all DNS responses. Encrypting DNS messages using DNSCurve does not introduce additional query latency since the server's public key is stored in NS records and sent to the client. DNSCrypt is an implementation of DNSCurve and has been deployed on OpenDNS to address privacy issues. ConfidentialDNS uses a similar DNS extension mechanism to offer privacy for the DNS protocol. It proposes a new resource record type, ENCRYPT, to transmit the public key of the DNS server to the client. However, DNSCrypt and ConfidentialDNS have not applied for inclusion in standardized documents, which limits the large-scale deployment.

Unlike the traditional security extension standard, DNS-over-TLS (DoT) [22] focuses more on the encryption of DNS interaction messages. Network applications, including file transfers, web browsing, and remote desktop, rely on the TLS to securely exchange data and handle privacy authentication and integrity. The IETF Request For Comments (RFC) [22] defined a way to encrypt and transmit DNS packets using TLS. DoT is a privacy-preserving mechanism designed to address DNS spoofing and cache pollution and makes data tampering more difficult, which can defend against such attacks by encrypting messages by a secure TLS channel.

DNS-over-HTTPS (DoH) [23] uses HTTPS or HTTP/2 for private connections. The use of HTTPS encryption for web servers has gained widespread acceptance. The resolving results returned by DNS servers are signed, and any misbehavior will be caught. Unlike its competitor DoT, DoH does not encrypt a single query but transmits it through an encrypted tunnel between the client and the server. Hence, we are unable to identify DNS queries from all other web traffic. Although famous web browsers, including Mozilla, have announced the support for DoH in published browsers, it will bypass child pornography and lose visibility over network traffic. The DoH makes it impossible for network operators to maintain control.

2.2. Intel SGX

TEE is a secure container that runs and protects sensitive code and data inside a CPU. As the operating system becomes larger and more complicated, it is impossible to get rid of a diverse set of vulnerabilities [18,24]. Typical encryption algorithms and security mechanisms are suitable for data storage and transmission but often fail to preserve the security of data during execution and usage. While applications can create encrypted memory areas for program execution in a user layer, such as Intel SGX [19] and ARM TrustZone [25]. These areas can be seen as enclaves isolated from any other privileged code in a host. The processor only allows code that is measured and loaded in the enclave to access sensitive data. Applications running in enclaves are free from sniffing and corruption when interacting with normal system interfaces. Even if privileged system software, including BIOS and OS, is compromised, the enclave still ensures that the code and data are not tampered with or monitored, which ensures the integrity and confidentiality of the code and data.

Intel SGX is one of the most widely used TEEs in practice [26]. Intel has added SGX support to most CPUs after its sixth-generation architecture. TrustZone requires signature verification from hardware manufacturers to run in a secure execution environment, which limits its deployment by software developers [25]. The Intel CPU provides a dedicated execution environment isolated from the normal operating system. Developers can divide the core functions of applications into hardened enclaves to improve the security of their applications while reducing the attack surface [27]. To realize the switching between the two environments, bridge functions will pass ECALL/OCALL instructions to enter and exit the enclave environment [28]. The code of E/OCALL has been generated automatically according to a function declaration file. However, mandatory code separation brings many

challenges to the development of SGX applications [29]. Developers often have to refractor existing codes to adapt to SGX-specific development models [30].

SGX supports two types of authentication, local authentication (LA) and remote authentication (RA) [31]. Utilizing the RA feature, an enclave can prove that it has not been compromised and is running on a genuine SGX-enabled hardware platform. The remote authentication protocol involves two intel-controlled services: Quoting Enclave (QE) and an Intel Attestation Service (IAS), which act as authentication providers and verifiers, respectively. As important research progress in the field of trusted computing, Intel SGX offers an efficient solution for anonymous authentication and verification [28,32,33]. Besides shielding systems [34,35], SGX is also used to enhance the security of applications such as machine learning [33–37], intrusion detection framework [36], serverless [26,38], and cloud computing.

This strong isolation of SGX makes it difficult to share code or data between enclaves. The authors of [27] propose a single-address-space approach, which runs all processes in a single enclave and enables memory-sharing in library OSes. To evaluate network trust and incent collection nodes to share security-related data for intrusion detection, the authors propose [36] an SGX-enabled decentralized intrusion detection framework based on blockchain to avoid forking and honestly perform intrusion detection.

The remote authentication in Intel SGX is a key step for relying on parties to build secure connections between a client, which holds the sensitive data and the enclave. To fill this gap between practical developments and formal proof for attestation services in SGX, the authors of [31] adopt a formal approach for the verification of third-party attestation and prove Intel SGX provides the announced confidentiality of secrets and integrity of the evidence.

One of the security concerns about SGX is side-channel attacks. To mitigate the side-channel attack and system call snooping against the SGX technology, the authors of [39] adapt the ORAM protocol to present a data-oblivious filesystem. To further address notorious performance issues of ORAM, the authors of [40] leverage an external device, FPGA, to implement a trusted storage service within a completely isolated environment secure from side-channel attacks.

2.3. The List of Abbreviations

Table 2 provides the list of abbreviations used in this paper.

Table 2. The list of abbreviations used in this paper.

Abbreviation	Term
DNS	Domain Name System
IoT	Internet of Things
IoV	Internet of Vehicles
TEE	Trusted Computing Environment
SGX	Software Guard Extensions
TLS	Transport Layer Security
DoTT	Domain service over Trusted Transport Layer Security
DoT	DNS-over-TLS
DoH	DNS-over-HTTPS
ECALL	Enclave Calls
OCALL	Outside Calls
NS	Name Server
RA	Remote Attestation
QE	Quoting Enclave
IAS	Intel Attestation Service
DCAP	Data Center Attestation Primitives
mDNS	Multicast DNS

3. A Trusted and Backward-Compatible Addressing Service

Security and privacy have become important considerations when using network services in the ever-changing public environment. DNS is being used on kinds of platforms,

operating systems, and applications. It must be hardened to avoid undesired or malicious attacks.

3.1. Trusted Service

To offer confidentiality and integrity for DNS programs, we propose a trusted and backward-compatible DNS based on the Intel SGX. In this service, a resolver running in an enclave is introduced into the traditional DNS infrastructure. After verifying the attestation evidence of a resolver and establishing an authenticated TLS channel, the DoTT encapsulates the encrypted query content in a standard DNS message to keep it from exposure and disclosure to unwanted parties.

The flow of the authenticatable DNS service for privacy-preserving is illustrated in Figure 1. The DNS client is a stub process that can run in a normal environment without the SGX hardware. This stub is responsible for receiving and forwarding DNS resolution requests from various web applications, such as browsers that run on the same host as these applications. Benefiting from the design of DNS, we can enhance the privacy of DNS by simply replacing the regular certificate validation in TLS with a Quote report validation from an IAS. This change has less impact on the existing architecture and is transparent to the upper-layer web applications. We assume that: (i) the DNS stub (denoted as *S*) has deployed a list of trusted DoTT values (denoted as *L*); (ii) there is a channel between the DoTTs (denoted as *D*); (iii) public domain name servers (denoted as *NS*) are safe and faithful; (iv) there are no vulnerabilities and malicious code inside the enclave. This assumption is reasonable since enclave code is often developed by users themselves and has been audited.

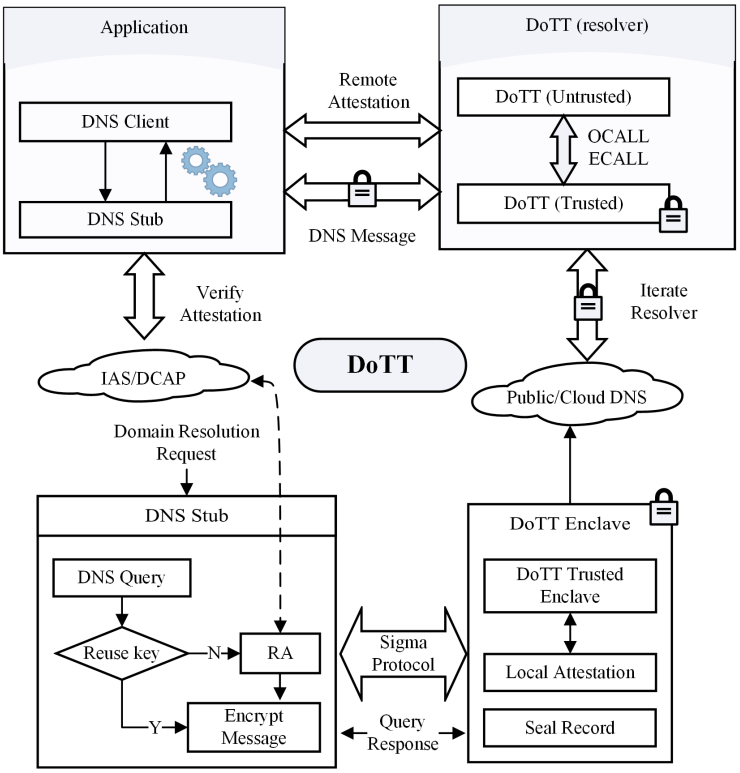

Figure 1. DoTT: The privacy enhancement framework for DNS based on TEE (Intel SGX).

The DoTT consists of two stages: (i) the remote attestation and the establishment of a secure channel between the DNS client (stub) and the DNS resolver (DoTT); (ii) the private DNS query based on the TLS channel. The first stage closely integrates TLS with the remote authentication using SGX features as follows:

1. When web applications initiate a DNS query (denoted as R), the S will check whether a TLS connection has been established.
2. The S directly forwards R to D using the encrypted channel if the connection is established. When D receives the request, it will get the IP address corresponding to the request from the NS as the conventional recursive resolving method and return it to S.
3. The S will challenge D by sending a nonce to it if the connection is not established. To convince R that it is a genuine Intel SGX enclave running on an authenticatable DNS resolver, D first performs the LA with the quoting enclave (donated as Q) in the local machine. If S succeeds in generating an authentication message (denoted as M), Q will sign the hash value of the authentication result, which comprises the public key of D.
4. After receiving the report, S can forward it to the IAS, which is a public webserver to provision a message generated by genuine Intel hardware. We can also adopt the Intel SGX DCAP [31] (Data Center Attestation Primitives) as an alternative to provision attestation services into a user-side database. After registering the genuine information to DCAP during the deployment phase, the whole remote attention procedure can be finished without an Internet connection in DCAP. After validating the reported signature using the certificate of IAS, S will check the enclave identify (MRENCLAVE) is in the list L. Then, the DNS stub can assure that DoTT is running a genuine SGX enclave and uses the public key hash in M for the following TLS connection.
5. Once S confirms the authentication and validation of M, it will exchange a symmetric session key (donated as K) to encrypt subsequent DNS queries. Therefore, S and D can communicate in an authenticated and privacy-preserving way with K.

3.2. Backward-Compatible Service

As shown in Figure 1, DoTT is separated into two parts: the untrusted system runs in normal environments, and the trusted part runs in an enclave. To ensure the minimum attack surface, SGX limits the code that can be run in an enclave. In the SDK provided by SGX, only basic operation functions are included, while file I/O and network communication functions are forcibly separated into untrusted codes. Hence, the existing codes must be refactored to run in SGX. The code separation in the development model of SGX brings challenges to the transplantation of existing software [41]. First, it takes a lot of time to divide the code into trusted and untrusted parts. Second, there is no guarantee that the reconstructed code is completely compatible with the original. The code refactoring risks damaging the existing system.

To avoid the problem of rewriting all DNS codes, we propose a new compiling model, as shown in Figure 2. We first replace the part of the existing code that needs to be run in SGX with a scaffold function. This tool will selectively execute the SGX and normal functions with the same logic according to the runtime environment. For example, the original file reading function fread(args…) can be replaced by common_ecall (read, ret, sgx_fread, fread, args…). common_call will be extended to a branch function during the compilation process. It will first check whether the current host has the SGX feature at the runtime. If it is supported, the function sgx_fread of the SGX version will be called, and the original fread function will be called in turn. The parameters (args) can be transferred from the normal environment to the enclave by using the sgx_edger, a compilation tool provided by Intel SGX SDK through stub functions.

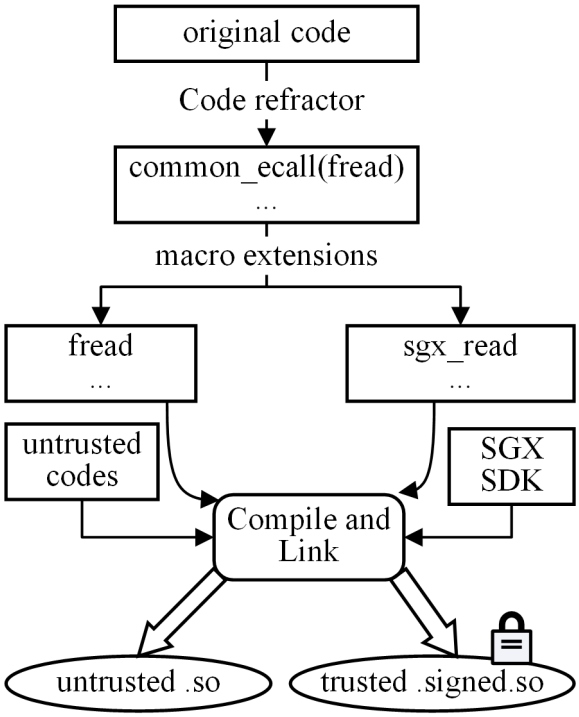

Figure 2. The backward-compatible compilation model for refactoring the code of a DNS resolver.

In addition to the code, we need to refactor the linking process. The SGX technology limits the use of standard libc functions and only provides a small number of system abstractions for developers due to security considerations and design limitations. In addition to simple macro replacement (common_ecall), we also need to move the code to be run in an SGX enclave to a common file. During the linking process, this common code will be compiled into a static (.a) file first and then linked with SGX runtime libraries into a signed resolver.signed.so file. At the same time, we will compile and generate a normal resolver.so file. This compilation and linking process are all performed automatically. We chose Bazel [42], a scalable and extensible build system, to define a new library rule named enclave_library in the implementation. The prototype is enclave_library (name, edl, trusted_hdr, trusted_src, untrusted_src, hdrs, untrusted_deps) where name indicates the generated library name, edl indicates the original SGX interface description file, trusted_hdr and trusted_src, respectively, indicate the header file and source codes to be compiled into trusted .so file, and untrusted_src and untrusted_deps indicate the untrusted source codes dependent files.

Notes that untrusted code can call the code in the trusted file, while trusted code cannot call untrusted code conversely. The compilation model we propose will generate two sets of codes and runtime libraries for an application. In a non-SGX environment, untrusted code plus trusted code is the same as the code logic in the original DNS resolver. To make the non-SGX runtime of DoTT compatible with the authentication process, when the DNS stub launches a challenge, the non-SGX DoTT will return an empty Quote, and then the stub will ignore the report verification and continue the TLS establishment process. This change does not affect the security assumptions in the DoTT.

Through this new compilation model, we only need to make small changes to the original code and move part of the code to a new file. Since we have also generated a non-SGX version of runtime libraries, we can first test DoTT in the normal environment and verify that the logic of the changed program is still correct. Unlike other enclave-specific applications developed for SGX, we can make an application both run in SGX and a normal environment at the same time through the new compilation and extension, thus enhancing the backward compatibility and reducing the risk of code refactoring.

3.3. Optimization for SGX

We use DoTT to encrypt and decrypt the DNS stub and NS to ensure the privacy and security of DNS messages. However, SGX is only an extended instruction set of CPU, which can only perform various arithmetic operations and lacks the network communication function required by TLS. Therefore, we started a network listening process and a network request process in the untrusted part of DoTT, one for receiving DNS query requests, the other for forwarding requests to NS and receiving resolution results. The ECALL/OCALL overhead of SGX will also reduce the processing performance of DoTT. To reduce the forwarding operation of ECALL/OCALL between the system and the enclave, we also started two threads in the DoTT enclave, one of which will immediately respond and decrypt the message after the external process receives the query request. When receiving the resolving result, the other thread will use the session key to perform an encryption operation and return the result to a DNS client in the form of ciphertext.

We set up a cache in the enclave to improve the resolving speed of DoTT. Every time DoTT responds to a DNS query request, it first checks the previous response list. It directly extracts the result from the list and encrypts them if the request has been parsed before, thus omitting the interaction with NSs. However, all available physical memory of an enclave is limited to a maximum of 128 M. Thus, the cache list will soon be full. To overcome this problem, we maintain a cache list with a size of 20 M. If the list exceeds this size limit, we use sgx_seal to save part of the cache on a disk. When a new DNS request is processed next time, if no resolution record is retrieved from the list of caches in memory, we use sgx_unseal to load 1M records from the disk into the enclave at a time. We will initiate a domain name resolution request to NSs only after retrieving the cache in memory and disk. Thus, the interaction with NSs is minimized. With the release of next-generation SGX hardware and its support for dynamic memory [43], the impact of memory on the DoTT will be weakened.

Intel SGX SDK provides a standard library for multithreading. Multithreading is an effective way to achieve concurrency. DoTT is an I/O-intensive application as a networking service. However, it has to utilize the OCALL/ECALL mechanism to handle external network requests. The dual I/O workload may degrade response performance. When the OCALL operation of an SGX function is completed in the background, the DoTT can perform other processing operations. When a response process enters the enclave, DoTT will create a poll object and put it into the waiting queue. The poll uses a red and black tree as the index structure of ECALL and OCALL queues. The poll queue and the scheduler will check whether a thread needs to be awakened due to the callback finish of the system call or enclave response. If there are no ECALL/OCALL executes, the scheduler will be recycled to the thread pool.

3.4. Security Analysis

The DNS protocol did not fully consider network security issues at the beginning of the design., while our proposed DoTT ensures the credibility of the results with the RA and trusted execution features provided by TEE [44]. Although SGX limits the length of the verification report, we can still ensure that the verification results are generated in a trusted environment through hash values. The privacy between DoTT and public root servers and iterate servers is protected by the same TLS. Note that DoTT only plays the role of DNS resolver. Its connection with the NS is an ordinary TLS connection. Although we can refactor the NS to run it in an SGX enclave, SGX only exists in desktop CPUs at present, while NS generally runs in servers equipped with high-end chips, which generally do not yet support SGX features. The IAS and MREnclave ensure that DoTT establishes a trusted TLS connection according to the expected behavior and performs the specified DNS processing without leaking DNS query messages and metadata to external applications. The DNS client already holds a certificate from IAS as well as a trusted list during the deployment process. Therefore, a malicious user cannot tamper with the generator of the evidence and forge the verifier of the evidence at the same time. If the authentication step

in the last section fails, a valid TLS connection cannot be established, and the following DNS query will fail. Finally, we use the disk as a cache to enhance system performance. The contents of the cache are also stored as ciphertext and can only be decrypted by a properly loaded DoTT enclave. Therefore, there is no risk of compromise as well.

4. Experimental Results

To test the feasibility of DoTT, we carried out some experiments to compare DoTT with other DNS enhancement methods based on TLS. We test the performance of the proposed framework in accessing different domain names on the prototype system developed using the C++ language based on SGX SDK. The characteristics of methods in DNS encryption are further compared. We evaluate the latency details of the DNS query using a modified DNS benchmarking tool BIND. The DNS query delay is calculated by averaging the DNS query. All of our evaluations are performed on an Intel Core i7-7700 CPU @3.60 GHz with 16 GB RAM.

We first test the performance of DoTT on a normal host and a host that supports SGX features. As shown in Figure 3, the Plain DNS had the highest performance since no security enhancements were adopted. While DoTT (Non-SGX) indicates the performance of DoTT running on a normal host. There are two reasons why DoTT performance is weaker than the Plain DNS: (i) one is the loss due to the use of TLS encryption; (ii) the other is the extra function calls resulting from the refactoring of the original code to allow two sets of runtime libraries to be generated from the same set of code. However, the loss from direct function calls is still less than the loss from ECALL/OCALL in the DoTT running in a host enabling the SGX feature.

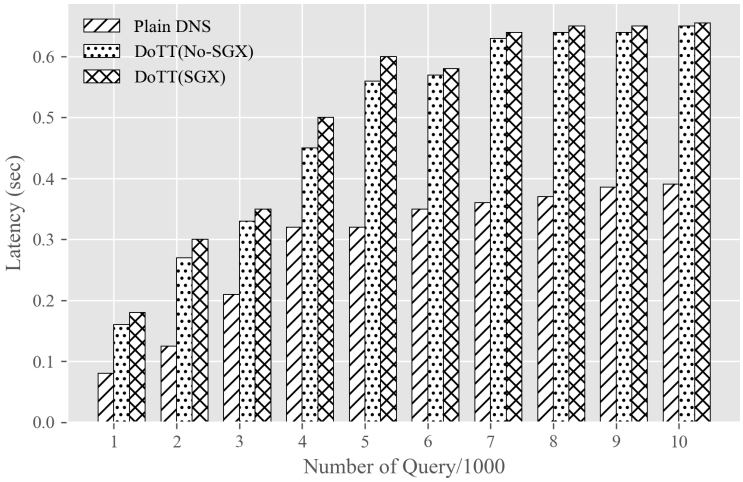

Figure 3. The latency of response among the plain DNS, DoTT (Non-SGX), and DoTT (SGX) resolvers.

As shown in Figure 4, we run the benchmark using 6 threads and keep the number of active DNS queries varying from 1000 to 10,000. We gradually increase the concurrency of DNS queries to simulate an increasing number of resolved requests. The baseline is a plain DNS resolver that only relays DNS queries and responses. The average latency for processing requests in DoTT is about the same as that of a plain DNS resolver. We also use a symmetric secret key to ensure the privacy of the result. With the hardware acceleration of SGX, although our method has an additional layer of encryption to ensure privacy, the response rate is equivalent to that of DNSSEC. For other TLS-based methods, it brings performance degradation for the handshake and exchange of secret keys. Although our method also uses a modified Sigma protocol [44], due to SGX coming with hardware encryption instructions, we can archive a higher response rate.

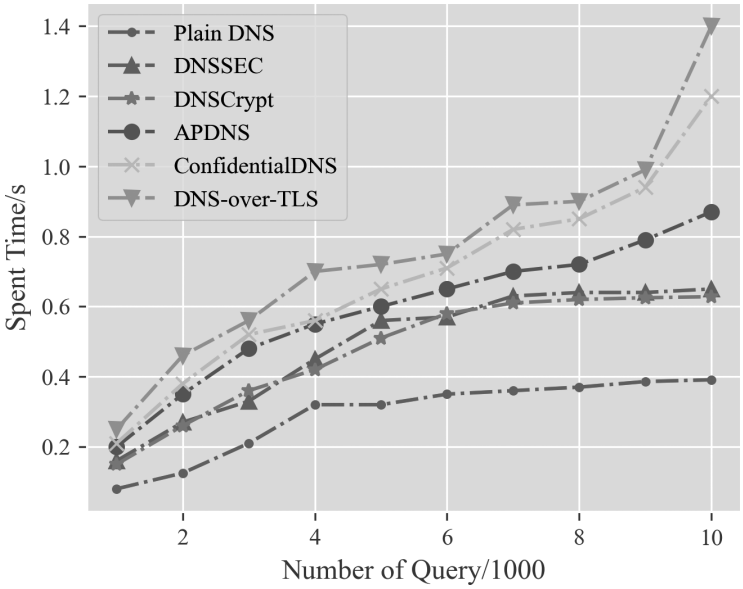

Figure 4. Performance evaluation among kinds of secure DNS resolvers.

To test the performance of enabling TLS connections in DNS on the address resolution, one way is to establish a TLS connection for each connection. The other is to establish a TLS connection in advance and reuse this connection in the next resolving request. When testing, we can control the client stub and resolving server at the same time, and this change is invisible to the user program. We use the curl tool to initiate domain name resolution requests. In order to expand the test coverage, we selected 10 well-known websites as target domain names.

As shown in Figure 5, if the TLS connection is reestablished every time, the latency overhead of DoTT is 18% higher than that of DoT. If the TLS connection is maintained, the overhead of DoTT is only 8% higher than that of DoT. In practice, DNS clients and resolving servers often reuse TLS connections until a TTL expires. If we consider the overhead of TLS in a larger scope; that is, the goal of initiating domain name resolution is to request web resources, then the TLS connection of DNS will have less influence on the whole resource request process. Note that we have not tested the performance of TLS between the resolution server and the domain name server since DoTT and DoT both use the traditional untrusted domain name resolution process at this stage.

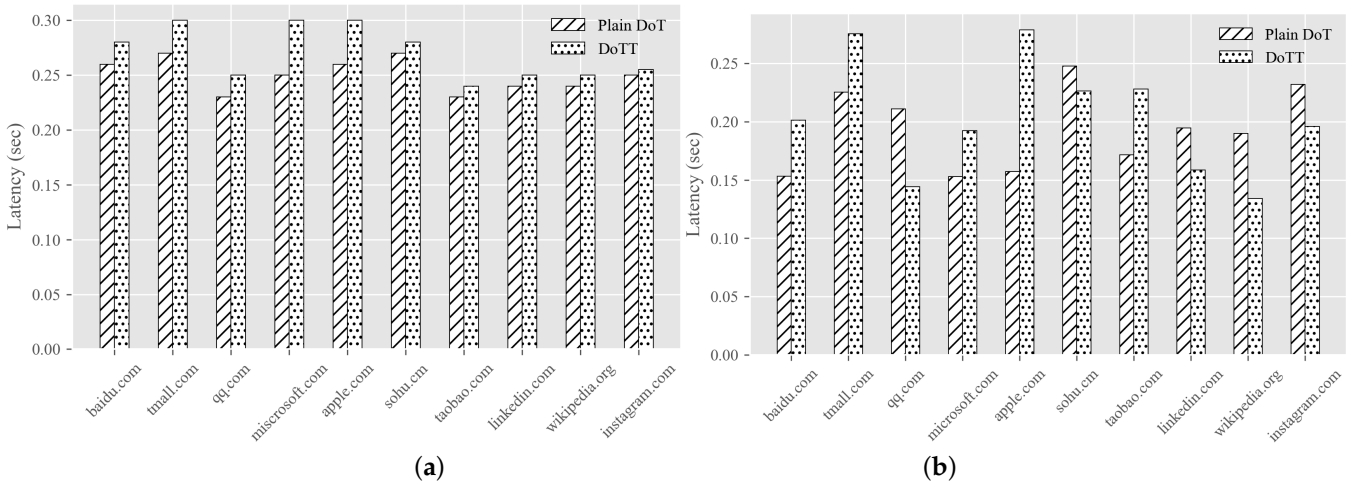

Figure 5. Performance evaluation of establishing TLS connections between DoT and DoTT. (**a**) Establishing TLS connections for each new request, and (**b**) reusing the existing connections for new requests.

We also evaluate the performance of DNS caching in DoTT and DoT in Figure 6. In DNS implementations, resolving servers often results in the speeding up of response requests. To prevent memory from being sniffed and polluted by privileged applications, such as operating systems, SGX uses a separate memory model. We still perform our domain resolution experiments on 10 well-known websites. Figure 6a,b show the evaluation results for caching 10 and 100 domains, respectively. Note that we have used the error bar to show the test results and the standard deviation in addition to the mean value. As expected, the response speed of DoT is significantly higher when the DNS caching feature is turned on, while the response time is almost constant. The response time also gets a significant improvement for DoTT. Although SGX has to perform encryption and decryption operations to prevent OS intervention when exchanging data between CPU and memory, these encryption operations are efficiently implemented by hardware instructions. Therefore, secure memory access in SGX has less impact on applications compared to normal memory operations. On the other hand, we see a large deviation disturbance in the response time of DoTT, which is because the prototype system we implemented only utilizes SGX features without further optimization beyond those of SGX. Further optimization of DoTT will be our future work.

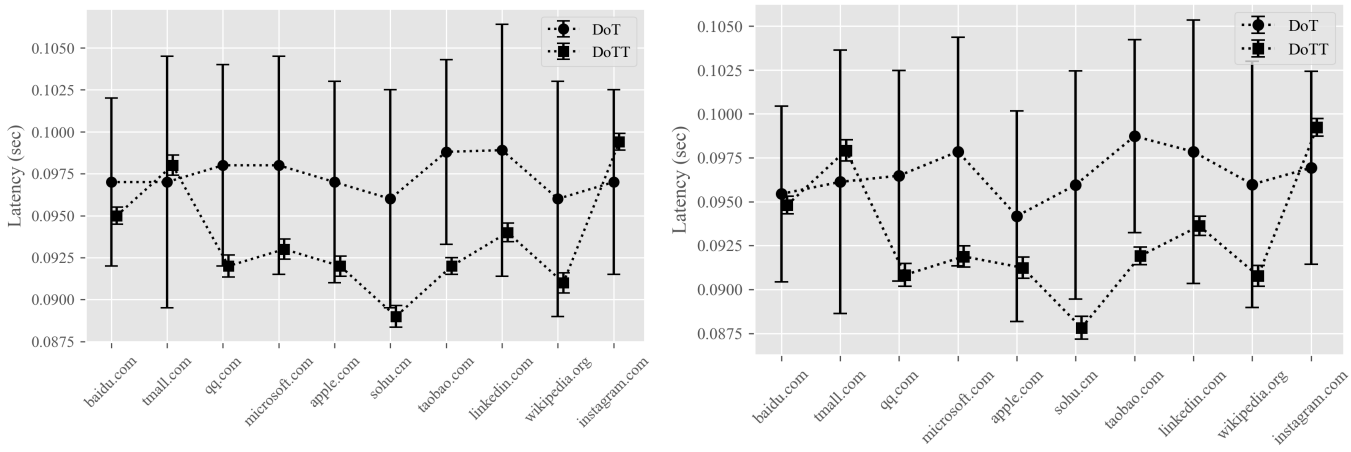

Figure 6. Performance evaluation of caching the resolved results between DoT and DoTT. (**a**) Caching 10 results, (**b**) caching 100 results.

5. Conclusions

To fully utilize the advantages of new technologies in the next-generation networks, key elements of network architectures, including DNS, must be evolved to cope with the challenges brought by the provision of these high-value services. DNS has become one of the most critical components in emerging services. As more and more work pays attention to the security and privacy of services, the problem of addressing privacy leakage is becoming prominent. Based on the analysis of the existing technology of DNS privacy enhancements, a new method of DNS privacy-preserving is proposed. We adopt the TEE technology, which is different from previous methods, and the public-key cryptography infrastructure to encapsulate the attestation flow under the existing DNS architecture, which not only preserves the privacy of DNS but also improves the deployability for the DNS resolver and protects the privacy of network services by hardware-enhanced signatures. To keep the backward compatibility of the built application, we build a new compilation-link model to reduce the refactoring of existing code and conform to the logic of DNS messages in the normal and SGX environments. Experimental results show that DoTT reduces the communication overhead between the DNS client and the DNS resolver and effectively maintains the low latency of DNS when preserving the privacy and security of data.

In future work, we will transplant and test other network services, such as trusted time and databases, into SGX to verify the feasibility and performance overhead of replacing existing services with trusted computing. Secondly, we will test the performance of the proposed mechanism in more comprehensive simulation environments, such as NS2 and OPNET. Last but not least, we will rewrite the protection system using the Rust language, which is widely used in the security field, and evaluate its performance with the current system.

Author Contributions: Conceptualization, D.Z. and Z.G.; methodology, software, validation, and data curation, L.R.;writing—original draft preparation, D.Z.; writing—review and editing, Z.G.; supervision, D.Z.; project administration and funding acquisition, Z.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported in part by the National Key Research and Development Program of China (2019YFB1706003), Natural Science Foundation of China (61902082, U20B2046), Guangdong Key R&D Program of China (2019B010136003), Guangdong Higher Education Innovation Group (2020KCXTD007), Guangzhou Higher Education Innovation Group (202032854), the Guangdong Province Universities and Colleges Pearl River Scholar Funded Scheme (2019), Guangdong Basic and Applied Basic Research Foundation of China (2022A1515011542), and Guangzhou Science and technology program of China (202201010606).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Sahraoui, Y.; Kerrache, C.A.; Korichi, A.; Vegni, A.M.; Amadeo, M. LearnPhi: A Real-Time Learning Model for Early Prediction of Phishing Attacks in IoV. In Proceedings of the 2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 8–11 January 2022.
2. Gu, Z.; Wang, L.; Chen, X.; Tang, Y.; Wang, X.; Du, X.; Guizani, M.; Tian, Z. Epidemic Risk Assessment by a Novel Communication Station Based Method. *IEEE Trans. Netw. Sci. Eng.* **2022**, *9*, 332–344. [CrossRef] [PubMed]
3. He, X.; Wang, J.; Liu, J.; Han, Z.; Lv, Z.; Wang, W. DNS Rebinding Detection for Local Internet of Things Devices. In *Frontiers in Cyber Security*; Springer: Singapore, 2020; pp. 19–29. [CrossRef]
4. Yan, Z.; Lee, J.H. The road to DNS privacy. *Future Gener. Comput. Syst.* **2020**, *112*, 604–611. [CrossRef]
5. Liu, Y.; Jiang, Y.; Ge, N. Design of Personal Terminal DNS Agent. *J. Commun. Inf. Netw.* **2021**, *6*, 251–266. [CrossRef]
6. Shafiq, M.; Tian, Z.; Bashir, A.K.; Du, X.; Guizani, M. CorrAUC: A Malicious Bot-IoT Traffic Detection Method in IoT Network Using Machine-Learning Techniques. *IEEE Internet Things J.* **2021**, *8*, 3242–3254. [CrossRef]
7. Bhushan, K.; Gupta, B.B. Distributed denial of service (DDoS) attack mitigation in software defined network (SDN)-based cloud computing environment. *Ambient. Intell. Hum. Comput.* **2019**, *10*, 1985–1997. [CrossRef]
8. Bumanglag, K.; Kettani, H. On the impact of DNS over HTTPS paradigm on cyber systems. In Proceedings of the 2020 3rd International Conference on Information and Computer Technologies (ICICT), San Jose, CA, USA, 9–12 March 2020; pp. 494–499. [CrossRef]
9. Shah, S.L.; Abbasi, I.A.; Bashier Gism Elseed, A.; Ali, S.; Anwar, Z.; Rajpoot, Q.; Riaz, M. TAMEC: Trusted Augmented Mobile Execution on Cloud. *Sci. Program.* **2021**, *2021*, 5542852. [CrossRef]
10. Gu, Z.; Li, H.; Deng, L.; Du, X.; Guizani, M.; Tian, Z. IEPSBP: A Cost-Efficient Image Encryption Algorithm Based on Parallel Chaotic System for Green IoT. *IEEE Trans. Green Commun. Netw.* **2022**, *6*, 89–106. [CrossRef]
11. Zhang, J.; Tong, W.; Zhu, L.; Ou, W.; Li, X. Evaluating DNS Vulnerability to Cache Injection. In Proceedings of the 2019 IEEE International Conference on Computation, Communication and Engineering (ICCCE), Longyan, China, 8–10 November 2019; pp. 134–137. [CrossRef]
12. Wang, W.T.; Hu, N.; Liu, B.; Liu, X.; Li, S.D. Survey on technology of security enhancement for DNS. *J. Softw.* **2020**, *31*, 7.
13. Zhauniarovich, Y.; Khalil, I.; Yu, T.; Dacier, M. A Survey on Malicious Domains Detection through DNS Data Analysis. *ACM Comput. Surv.* **2018**, *51*, 67:1–67:36. [CrossRef]
14. Aijaz, N.U.; Misbahuddin, M.; Raziuddin, S. Survey on DNS-Specific Security Issues and Solution Approaches. In *Data Science and Security*; Jat, D.S., Shukla, S., Unal, A., Mishra, D.K., Eds.; Springer: Singapore, 2021; Volume 132; pp. 79–89.
15. Liang, H.; Li, M.; Chen, Y.; Yang, T.; Xie, Z.; Jiang, L. Architectural Protection of Trusted System Services for SGX Enclaves in Cloud Computing. *IEEE Trans. Cloud Comput.* **2021**, *9*, 910–922. [CrossRef]

16. Gu, Z.; Hu, W.; Zhang, C.; Lu, H.; Yin, L.; Wang, L. Gradient Shielding: Towards Understanding Vulnerability of Deep Neural Networks. *IEEE Trans. Netw. Sci. Eng.* **2021**, *8*, 921–932. [CrossRef]
17. Jin, Y.; Tomoishi, M.; Fujikawa, K.; Kafle, V.P. A Lightweight and Secure IoT Remote Monitoring Mechanism Using DNS with Privacy Preservation. In Proceedings of the 2019 16th IEEE Annual Consumer Communications Networking Conference (CCNC), Las Vegas, NV, USA, 11–14 January 2019; p. 12. [CrossRef]
18. Zheng, W.; Wu, Y.; Wu, X.; Feng, C.; Sui, Y.; Luo, X.; Zhou, Y. A survey of Intel SGX and its applications. *Front. Comput. Sci.* **2020**, *15*, 3. [CrossRef]
19. McKeen, F.; Rovich, A.I.; Berenzon, A.; Rozas, C.V.; Shafi, H.; Shanbhogue, V.; Savagaonkar, U.R. Innovative instructions and software model for isolated execution. In Proceedings of the HASP '13: The Second Workshop on Hardware and Architectural Support for Security and Privacy, Tel-Aviv, Israel, 23–24 June 2013.
20. Saraj, T.; Yousaf, M. Design and implementation of a lightweight privacy extension of DNSSEC protocol. In Proceedings of the 2017 13th International Conference on Emerging Technologies (ICET), Islamabad, Pakistan, 27–28 December 2017; p. 16. [CrossRef]
21. Anagnostopoulos, M.; Kambourakis, G.; Konstantinou, E.; Gritzalis, S. DNSSEC vs. DNSCurve: A side-by-side comparison. In *Situational Awareness in Computer Network Defense: Principles, Methods and Applications*; IGI Global: Hershey, PA, USA, 2012; pp. 201–220.
22. Dickinson, S.; Gillmor, D.; Reddy, T. *Usage Profiles for DNS over TLS and DNS over DTLS; RFC 7858*; Internet Engineering Task Force (IETF); IETF: London, UK, 2018.
23. Böttger, T.; Felix, C.; Gianni, A.; Leão Fernandes, E.; Tyson, G.; Castro, I.; Uhlig, S. An Empirical Study of the Cost of DNS-over-HTTPS. In Proceedings of the Internet Measurement Conference, Amsterdam, The Netherlands, 21–23 October 2019; pp. 15–21. [CrossRef]
24. Sierra-Arriaga, F.; Branco, R.; Lee, B. Security issues and challenges for virtualization technologies. *ACM Comput. Surv.* **2020**, *53*, 1–37. [CrossRef]
25. Pinto, S.; Santos, N. Demystifying arm trustzone: A comprehensive survey. *ACM Comput. Surv. (CSUR)* **2019**, *51*, 136. [CrossRef]
26. Brenner, S.; Kapitza, R. Trust More, Serverless. In Proceedings of the 12th ACM International Conference on Systems and Storage, New York, NY, USA, 3–5 June 2019; pp. 33–43. [CrossRef]
27. Shen, Y.; Chen, Y.; Chen, K.; Tian, H.; Yan, S. To Isolate, or to Share?: That is a Question for Intel SGX. In Proceedings of the 9th Asia-Pacific Workshop on Systems—APSys'18, Jeju Island, Korea, 27–28 August 2018; p. 18. [CrossRef]
28. Priebe, C.; Muthukumaran, D.; Lind, J.; Zhu, H.; Cui, S.; Sartakov, V.A.; Pietzuch, P. SGX-LKL: Securing the Host OS Interface for Trusted Execution. *arXiv* **2019**, arXiv:1908.11143.
29. Lind, J.; Priebe, C.; Muthukumaran, D.; O'Keeffe, D.; Aublin, P.L.; Kelbert, F.; Reiher, T.; Goltzsche, D.; Eyers, D.; Kapitza, R.; et al. Glamdring: Automatic Application Partitioning for Intel SGX. In Proceedings of the 2017 USENIX Annual Technical Conference (USENIX ATC 17), Santa Clara, CA, USA, 12–14 July 2017; pp. 285–298.
30. Silva, R.; Barbosa, P.; Brito, A. DynSGX: A Privacy Preserving Toolset for Dynamically Loading Functions into Intel (R) SGX Enclaves. In Proceedings of the 2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Hong Kong, 11–14 December 2017; pp. 314–321. [CrossRef]
31. Sardar, M.U.; Faqeh, R.; Fetzer, C. Formal Foundations for Intel SGX Data Center Attestation Primitives. In *Formal Methods and Software Engineering*; Springer: Cham, Switzerland, 2020; pp. 268–283. [CrossRef]
32. Liu, X.; Guo, Z.; Ma, J.; Song, Y. A Secure Authentication Scheme for Wireless Sensor Networks Based on DAC and Intel SGX. *IEEE Internet Things J.* **2021**, *9*, 3533–3547. [CrossRef]
33. Chen, Y.; Luo, F.; Li, T.; Xiang, T.; Liu, Z.; Li, J. A training-integrity privacy-preserving federated learning scheme with trusted execution environment. *Inf. Sci.* **2020**, *522*, 69–79. [CrossRef]
34. Tsai, C.C.; Porter, D.E.; Vij, M. Graphene-SGX: A Practical Library OS for Unmodified Applications on SGX. In Proceedings of the 2017 USENIX Annual Technical Conference (USENIX ATC 17), Santa Clara, CA, USA, 12–14 July 2017; pp. 645–658.
35. Shen, Y.; Tian, H.; Chen, Y.; Chen, K.; Wang, R.; Xu, Y.; Xia, Y.; Yan, S. Occlum: Secure and Efficient Multitasking Inside a Single Enclave of Intel SGX. In Proceedings of the 25th International Conference on Architectural Support for Programming Languages and Operating Systems, Lausanne, Switzerland, 16–20 March 2020; pp. 955–970. [CrossRef]
36. Liu, G.; Yan, Z.; Feng, W.; Jing, X.; Chen, Y.; Atiquzzaman, M. SeDID: An SGX-enabled decentralized intrusion detection framework for network trust evaluation. *Inf. Fusion* **2021**, *70*, 100–114. [CrossRef]
37. Tramer, F.; Boneh, D. Slalom: Fast, Verifiable and Private Execution of Neural Networks in Trusted Hardware. In Proceedings of the International Conference on Learning Representations, Orleans, LO, USA, 6–9 May 2019.
38. Qiang, W.; Dong, Z.; Jin, H. Se-Lambda: Securing Privacy-Sensitive Serverless Applications Using SGX Enclave. In *Security and Privacy in Communication Networks*; Springer: Cham, Switzerland, 2018; pp. 451–470. [CrossRef]
39. Ahmad, A.; Kim, K.; Sarfaraz, M.I.; Lee, B. OBLIVIATE: A Data Oblivious Filesystem for Intel SGX. In Proceedings of the Network and Distributed System Security Symposium, San Diego, CA, USA, 18–21 February 2018. [CrossRef]
40. Oh, H.; Ahmad, A.; Park, S.; Lee, B.; Paek, Y. TRUSTORE: Side-Channel Resistant Storage for SGX using Intel Hybrid CPU-FPGA. In Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, New York, NY, USA, 9–13 November 2020; pp. 1903–1918. [CrossRef]

41. Zhang, D.; Wang, G.; Xu, W.; Gao, K. SGXPy: Protecting Integrity of Python Applications with Intel SGX. In Proceedings of the 2019 26th Asia-Pacific Software Engineering Conference (APSEC), Putrajaya, Malaysia, 2–5 December 2019; pp. 418–425. [CrossRef]
42. Maudoux, G.; Mens, K. Correct, efficient, and tailored: The future of build systems. *IEEE Softw.* **2018**, *35*, 32–37. [CrossRef]
43. McKeen; Alexandrovich, F.; Anati, I.; Caspi, I.; Johnson, D.; Leslie-Hurd, S.; Rozas, C. Intel Software Guard Extensions (Intel SGX) Support for Dynamic Memory Management Inside an Enclave. In Proceedings of the Hardware and Architectural Support for Security and Privacy 2016, New York, NY, USA, 18 June 2016; p. 19. [CrossRef]
44. Scarlata, V.; Johnson, S.; Beaney, J.; Zmijewski, P. *Supporting Third Party Attestation for Intel® SGX with Intel® Data Center Attestation Primitives*; White Paper; Intel Corp.: Santa Clara, CA, USA, 2018.