Pervasive Internet-Wide Low-Latency Authentication

Cyrill Krähenbühl*, Markus Legner†, Silvan Bitterli‡, and Adrian Perrig§
Department of Computer Science, ETH Zurich, Switzerland
Email: *cyrill.kraehenbuehl@inf.ethz.ch, †markus.legner@inf.ethz.ch, ‡silvanbi@student.ethz.ch, §adrian.perrig@inf.ethz.ch

***Abstract*—In a world with increasing simplicity to store, transfer, and analyze large volumes of data, it becomes more and more important that data confidentiality and integrity be preserved in transit by default. Unfortunately, a large security gap exists between unprotected or low-security communication, such as opportunistic encryption and trust-on-first-use (TOFU) security, and high-security communication, such as TLS using server certificates or DNSSEC. Our goal is to reduce this gap and achieve a base layer for authentication and secrecy that is strictly better than TOFU security. We achieve this by designing PILA, a novel authentication method with dynamic trust anchors, which leverages irrefutable cryptographic proof of misbehavior to incentivize benign behavior. We implement PILA extensions for SSH, TLS, and DNS and show that the overhead for a typical SSH and TLS connection establishment is negligible, and that PILA only causes a marginal processing overhead of $\sim$100 µs per DNS response at the endpoints.**

I. Introduction

The revelations from the Snowden case have shown that large-scale Internet wiretapping is indeed occuring. For example, in June 2013 the GCHQ gained access to high-bandwidth fiber-optic cables to collect vast quantities of Internet traffic [31]. These incidents sparked not only the rise of opportunistic encryption protocols such as TCPCrypt [5], but also led to an increased deployment of TLS.

However, opportunistic encryption protocols, such as TCPCrypt or Opportunistic Wireless Encryption (OWE) [20], rely on a principle called trust-on-first-use (TOFU) [38] by assuming that the initial messages of a protocol are legitimate. SSH [40] is a TOFU protocol in the case when the other endpoint's public key is not pre-loaded and the public key fingerprint is not verified during the initial connection [19]. Unfortunately, TOFU protocols only provide secrecy against off-path and passive attackers but not against an active on-path attacker, who can perform a man-in-the-middle (MitM) attack. On the other hand, TOFU protocols are fast, easy to implement, and do not rely on external entities.

For client–server communication in today's Internet, there is an increasing use of certificate-based authentication protocols, which rely on certificate authorities (CA) in the TLS public-key infrastructure (Web PKI), or DNS domain zones in DANE [2]. As Let's Encrypt [24] solved most deployment issues of TLS by providing free and automated TLS server-certificate delivery, the use of TLS increased significantly, showing the desire for secure communication. However, TLS has restricted usability when endpoints do not have CA certificates; examples include communication modes other than client–server communication, such as peer-to-peer communication, or cases where setting up DNS entries for all devices is infeasible, such as Internet-of-Things (IoT) settings. Additionally, DNS resolvers, SSH servers, and IoT devices are often identified by their IP address which precludes domain-based authentication.

We propose *pervasive Internet-wide low-latency authentication* (PILA), a system to reduce the gap between TOFU security and strong authentication mechanisms (Web PKI and DANE). The goal is to create a mechanism that is orthogonal to existing strong authentication mechanisms and *not* to replace them. While current TOFU mechanisms are a big step towards secure communications by limiting the adversary from the entire Internet to an on-path adversary, this still constitutes a large attack surface: endpoints lack control over their traffic's path, and the Internet's routing system is vulnerable to IP prefix hijack attacks [10].

Therefore, in a first step, PILA restricts the adversary. We observe that an autonomous system (AS) provides a natural candidate to authenticate endpoints based on their IP addresses, as the AS already offers IP connectivity to its endpoints. PILA uses ASes in combination with the Resource Public Key Infrastructure (RPKI) [27] to limit the adversary to only the source and destination AS. PILA is incrementally deployable and incurs no computational overhead for intermediate nodes outside the endpoints's ASes.

In a second step, PILA provides accountability and proof in case either AS misbehaves. This provides a strong disincentive to attack, since after a MitM attack is revealed, the attacker is pinpointed by the system. In comparison, in TOFU security, even if a MitM attack is detected, *any* on-path entity could have performed the attack. We can generalize this approach, which we call *trust amplification*, to three principles:

I **Crude Authentication.** Endpoints are authenticated using a crude, relatively low-security approach which reduces the threat model to few entities.
II **Accountability.** Misbehaving entities can be detected through cryptographic proof.
III **Leverage.** Endpoints have means to apply pressure on misbehaving entities in the form of legal recourse, economic detriment, or bad publicity.

The *accountability* of entities trusted in the *crude authentication* in combination with the *leverage*, which disincentivizes misbehavior, inhibits coward attacks, i.e., attacks that are only

launched if the attack will not be detected [29]. In summary, this paper makes the following contributions:

- We introduce an approach called *trust amplification* that uses limited trust into a few entities, accountability, and leverage to provide stronger security guarantees than TOFU approaches.
- We describe PILA, an instantiation of trust amplification using ASes as opportunistically trusted entities and RPKI as global trust anchor to enable pervasive encryption and authentication for all Internet communication.
- We design and implement PILA extensions for SSH, TLS, and DNS. We evaluate their performance to show a negligible overhead for a typical SSH and TLS connection establishment and a marginal per-packet processing overhead of $\sim$100 µs for DNS.

II. Problem Definition

Our goal is to design a system for Internet-wide ubiquitous authentication between endpoints with stronger security guarantees than a TOFU approach. By authenticating endpoints through their IP addresses, the system is applicable to a large number of services (e.g., authentication of services with fixed IP addresses or devices without DNS names). On a similar note, the system should not introduce significant latency and computational overhead to allow a wide range of devices (e.g., resource-constrained IoT devices). In order to be deployed in a network of the size of today's Internet, the system has to scale in terms of computation and network resources. Our goal is *not* to replace existing, well-established authentication approaches, such as TLS with server certificates or DNSSEC. We attempt to improve the security guarantees of communication with weak security properties, and provide a base layer of security.

A. Assumptions

First, we assume that there exists a **global trust anchor**, i.e., a global PKI, agreed on by all entities, which provides keys and certificates to ASes. For this, we will leverage RPKI, which distributes resources in the form of AS numbers and IP address ranges to ASes, see §III. Second, we assume that participating **ASes** are able to authenticate their endpoints. Third, **endpoints** must know which AS they reside in and have access to an authentic version of the trust anchor. Fourth, **time synchronization** with a precision in the order of several minutes is essential for our certificate-based system with certificate lifetimes of several hours.

B. Attacker Model

Analogous to the notion of an honest-but-curious (HBC) attacker [33], we define an curious-but-cautious (CuBC) attacker. We call an entity a CuBC attacker, if the entity wants to read or modify traffic without being detected. Such an attacker does not necessarily follow all protocol steps correctly (e.g., violates the protocol if it is advantageous), but only when they are sure not to be detected. Entities desire undetectability to prevent financial loss (e.g., loss of customers), legal actions, or reduction in trust (e.g., bad publicity or removal from a pool of trusted entities). In terms of capabilities, we consider a Dolev–Yao attacker, i.e., an attacker that can intercept, modify, or inject packets but cannot break cryptography. The endpoints performing the authentication and the global trust anchors are assumed not to be compromised and to correctly follow all protocols steps. Malicious or compromised global trust anchors are relatively easy to detect, since there are only a few global trust anchors that have to be monitored for equivocation. If an endpoint does not follow the protocol correctly or reveals its private key, authentic keys cannot be derived.

Fig. 1: RPKI trust anchor structure abstraction used throughout this paper. Arrows indicate delegation through certificates.

III. Control-Plane Public-Key Infrastructure

The trust-amplification model introduced in this work can provide a baseline of security not just in the current Internet, but also in next-generation Internet architectures that provide an appropriate trust anchor. In particular, the Internet architecture SCION [34] integrates well with our trust-amplification model, since SCION introduces the concept of independent isolation domains and a flexible PKI, in which each such domain can choose its own roots of trust. This work, however, focuses on the current Internet and RPKI as trust anchor.

RPKI distributes resources in the form of AS numbers and IP address ranges. AS-number and IP-address resources can be delegated using X.509 resource certificate extensions [30]. RPKI is hierarchically structured with the Internet Assigned Numbers Authority (IANA) or regional Internet registries (RIRs) as roots of trust, which delegate a subset of their resources to local Internet registries (LIRs). To abstract from different implementations, we assume a generalized trust-anchor structure in Fig. 1 consisting of a trust anchor, a possibly empty delegation chain, a resource issuer, and an AS.

RPKI has been standardized by the IETF in 2012. Initially, there have been concerns about the misaligned incentives of different actors, such as CDN providers, in the Internet ecosystems [39]. However, a more recent measurement study over the lifetime of RPKI in 2019 shows that after a gradual start, RPKI deployment has seen a rapid increase in the last two years [6]. This trend can be observed from the RPKI monitor data as well [32]. As we show in §IV-C, PILA only requires source and destination ASes to support RPKI. In contrast, BGPsec requires *all on-path ASes* to support RPKI and thus provides very limited benefits to early adopters. This recent increase in RPKI deployment favors PILA as more endpoints can profit from an initial PILA deployment. Also, PILA provides additional incentive for ASes to deploy RPKI.

IV. Trust Amplification and PILA

This Section first introduces the trust-amplification model, and then describes PILA, an instantiation of trust amplification based on RPKI and ASes as opportunistically trusted entities providing IP-address-based authentication.

A. Trust-Amplification Model

Our authentication system builds on a trust-amplification model, which is a certificate-based authentication model relying on three key principles: *crude authentication*, *accountability*, and *leverage*. Trust amplification provides a generic model to increase the security of a certificate-based authentication system indirectly by deterring misbehavior of involved certificate-issuing entities. The meaning of misbehavior depends on the actual system used and typically means equivocating by issuing conflicting certificates. Trust amplification guarantees correct authentication if the certificate-issuing entities selected in *crude authentication* consist of CuBC attackers, which only launch attacks that cannot be detected.

Crude Authentication. The certificate-issuing entities are determined by the identity of the endpoint that requests the certificate. The first step is to reduce this number of certificate-issuing entities from all entities to a small subset. Such a reduction is only meaningful if the certificate-issuing entities are not omnipotent, i.e., cannot issue certificates for arbitrary identifiers. Ideally, the entities manage disjoint sets of identifiers which reduces the certificate-issuing entities for an endpoint to a single entity. In the trust-amplification model, the authenticating endpoint establishes a trust relation to a certificate-issuing entity of the endpoint that is authenticated, based on the following two principles.

Accountability. In order to increase trust into a certificate-issuing entity, which might initially be untrusted, certificate-issuing entities are held accountable for their actions. In the trust-amplification model, this property is achieved by generating irrefutable evidence that proves the misbehavior of a certificate-issuing entity to a third party. Important properties are resilience to slander (cannot forge false evidence) and framing (cannot manipulate an entity to produce false evidence itself), such that evidence is necessarily a result of improper behavior of a certificate-issuing entity.

Leverage. As a third principle, misbehavior must be disincentivized. After detecting misbehavior of a certificate-issuing entity M, other entities must have some form of leverage over M. For endpoints that are issued certificates by M, leverage could be economic detriments through loss of customers or legal recourse. For other endpoints, leverage could be a global or local trust rating of certificate-issuing entities based on collected evidence of misbehavior.

Trust amplification is similar to certificate transparency (CT) [26] in that both attempt to deter misbehavior by providing cryptographic proof thereof. However, with trust amplification, the power of each certificate-issuing entity is restricted to a subset of identifiers and they do not have to be trusted globally; this is in stark contrast to the omnipotent highly-trusted certificate authorities in the Web PKI with CT.

B. PILA Overview

PILA provides IP-address-based authentication as an extension to existing protocols, such as TLS, SSH, or DNS(SEC). PILA reduces the attack surface to the endpoints's ASes, and produces proof of equivocating ASes performing MitM attacks on their endpoints. The underlying protocol—the protocol that PILA extends to provide authentication for—must have (or must be extended to have) the property that an entity can authenticate itself using an X.509 certificate.

A PILA workflow where an initiator *A* authenticates a responder *B*, works as follows. First, *B*'s AS obtains a resource certificate for its AS number and IP addresses from RPKI and issues a short-lived certificate to *B* for *B*'s public key and IP address. *B* uses this certificate to, for example, authenticate an SSH or TLS handshake or sign a DNS reply. *A* verifies the authenticity of the handshake or signed reply using the RPKI certificate chain. *A* also keeps track of the used AS and endpoint certificates locally or adds them to an append-only log to retain the irrefutable proof of misbehavior, which can be detected through an out-of-band channel or an external auditor.

C. AS as Opportunistically Trusted Entity

In PILA, trust anchors are fully trusted entities similar to the root key in DNSSEC, but endpoints mostly interact with less, only *opportunistically* trusted entities. We propose to use ASes as opportunistically trusted entities. ASes are not omnipotent, as they operate on a subset of IP-address and AS-number resources, and any misbehavior is cryptographically provable through the issued PILA certificates. We can use ASes to bootstrap connection establishment, and then increase the trust placed into these ASes using trust amplification.

Each entity in the Internet is part of at least one AS, which is under the control of a single administrative entity. This facilitates providing a common service that authenticates endpoints (e.g., using a challenge–response protocol or pre-installed keys and certificates) and issues certificates. Another advantage is the typically close relationship between an endpoint and its AS, which allows for a stronger *leverage* in case of misbehavior. Since it is infeasible for an endpoint to authenticate each AS by itself (there are $\sim$71 000 active ASes according to the CIDR report [4]), RPKI is used as a trust anchor to authenticate ASes. RPKI resource issuers assign an AS a set of IP address prefixes that this AS is allowed to originate. An AS then issues short-lived certificates for its authorized IP address ranges.

Using ASes as opportunistically trusted entities promotes an incremental deployment model of PILA since there is the immediate benefit of endpoints within the AS being able to authenticate themselves.

D. IP-Address-Based Authentication

An advantage of IP-address-based authentication is that it could benefit all devices participating in Internet-wide communication which need to be addressable through an IP address. Endpoint certificates (CERT_E) in PILA identify endpoints by their public IP addresses instead of, e.g., the DNS name as in the Web PKI. The certificates are represented as X.509 resource certificates [23] in order to be compatible with existing PKI technologies. X.509 resource certificates add several extensions, most notably certificate policies [23] and IP-address and AS-number resources [30], which authorize

subdomains to use these resources. Endpoint certificates are issued by the AS of the endpoint and only for IP addresses within the AS's address range delegated by the resource issuer.

The chain of trust is constructed as follows: first, the self-signed trust anchor issues a certificate to the resource issuer (possibly with additional intermediate certificates in the delegation chain); then the resource issuer issues a long-lived certificate to the AS ($CERT_{AS}$) and the AS issues a short-lived certificate ($CERT_E$) to an endpoint; finally, the endpoint then uses the short-lived certificate to authenticate its communication (e.g., SSH or TLS handshake or DNS response). The verification starts from the trust anchor to the endpoint certificate and includes AS-number and IP-address validation for each certificate (an issued certificate must cover a subset of the issuer certificate's resources).

Endpoint Certificates. Endpoints request their certificate ($CERT_E$) from the certificate service of their ASes. An endpoint certificate binds the public key of an endpoint to a globally unique IP address owned by the AS. Endpoint certificates are typically short-lived on the order of hours to allow changing address assignments without the necessity for revocation. In scenarios where a more dynamic address allocation is desirable, certificates can be issued with lifetimes on the order of minutes if the increase in certificate issuance overhead is acceptable.

Additional Local Identifiers. In addition to the IP address, endpoint certificates might contain other (AS-)local identifiers, e.g., a username valid within the AS or a port range for which this certificate is valid, see §IV-F. In order to enable seamless transitions between short-lived certificates, an AS can issue multiple certificates with overlapping validity times as long as the public key and all identifiers are identical. An AS might refuse to add a local identifier to a certificate to protect itself against framing attacks if it cannot verify the correctness of the endpoint's claim of the identifier.

Anycast. Anycast can be used to run a service from multiple locations in the network using a single IP address. The locations of the service instances might reside in one or multiple ASes. In the case that all locations are in different ASes, the service at each location is independently issued its endpoint certificate and has its own private key. Having multiple certificates issued by different ASes for the same IP address is not an issue in PILA since each AS uses its own private key and certificate to issue the endpoint certificates and AS misbehavior does not impact other (benign) ASes.

In the case that at least two service instances are located in the same AS, the service owner has two options: (i) all service instances can use the same endpoint certificate and share a private key; or (ii) the service owner requests a single endpoint certificate for this AS and issues separate certificates for each service instance. The second solution reduces the impact of a private-key compromise but requires the additional intermediate certificate to be sent during connection establishment.

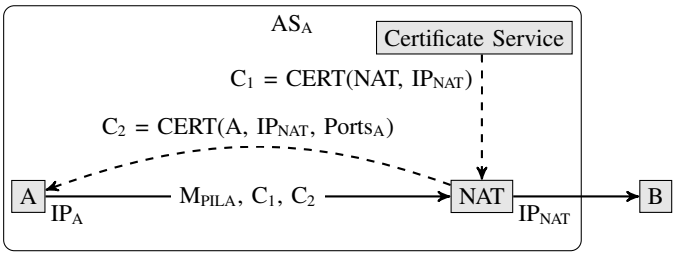

Fig. 2: The NAT device acts as opportunistically trusted entity.

E. Certificate Service

In order to support PILA, an AS needs to deploy a certificate service that generates and distributes short-lived endpoint certificates ($CERT_E$) and is able to authenticate endpoints within the AS. In PILA, issuing endpoint certificates is an automatic process similar to the Automatic Certificate Management Environment (ACME) [3]. Endpoints are authenticated either (1) with a challenge–response protocol, which requires setting up an HTTP server on the client as in ACME [37] or (2) based on a signature of the certificate signing request (CSR) by the endpoint. While the challenge–response protocol is automatic and does not require a pre-existing trust relation between the certificate service and the endpoint, the signature-based authentication allows managing certificates from entities other than the endpoint. Additionally, ASes allow endpoints to retrieve their certificate and possibly cached certificates of other (frequently fetched) ASes. These are the corresponding calls an endpoint can make:

- `getEPCert(IP address, [local identifier], public key)` either returns a short-lived certificate $CERT_E$ or an error message.
- `getASCert(AS number or IP address)` resolves the IP address to the correct AS number if necessary and returns the AS number together with the AS certificate if it is cached. If the certificate is not cached, the endpoint requests it either from the other endpoint or the other endpoint's AS.

F. NAT Devices

There are two issues for endpoints that reside behind NAT devices. Since PILA relies on the fact that ASes can identify endpoints by their IP addresses to distribute per-endpoint certificates, a simple mapping of an IP address to a (single) certificate is impossible. The second issue is that due to the NAT device, both endpoints have a different view of the opposite endpoint's IP address, which breaks IP-address–based authentication. To authenticate endpoints behind NAT devices, endpoints need to be able to use the public IP address of their NAT device as a globally unique identifier. We present an approach for authenticating between endpoints with intermediate NAT devices.

The NAT device requests an endpoint certificate ($CERT_E$) for itself (its public IP address) and acts as a trusted entity, distributing certificates to its endpoints, as depicted in Fig. 2. Endpoints then authenticate themselves by providing the NAT

device's certificate (C_1) in addition to their endpoint certificate (C_2) and the signed message (M_{PILA}). This requires endpoints to trust the NAT device in the same way as an AS, i.e., misbehavior by NAT devices becomes detectable. The certificates issued to endpoints behind the NAT device have the same IP address and additionally specify the outgoing port numbers as a local identifier as described in §IV-D. This allows other endpoints to authenticate an endpoint based on its public IP address and external port number (e.g., an endpoint providing a service on a specific port can request a certificate which covers this port). Port numbers are encoded in an X.509 extension in the same way as IP addresses in resource certificates [30].

Multiple sequential NAT devices are supported as well. Each NAT device issues certificates for NAT devices within its local network, which can in turn issue certificates for endpoints or NAT devices. Each nested NAT device thus requires an additional certificate. This isolates different hosts behind the NAT device and thus simplifies detection of misbehavior if a NAT device issues certificates with overlapping port number ranges for different entities. IPv6 solves IPv4 address shortage, one of the main reasons for the widespread deployment of NAT devices. We expect that with a growing IPv6 adoption, less NAT devices would be required and PILA deployment would become easier.

G. Session Resumption

If the underlying protocol supports session resumption, endpoints can combine the session resumption with a PILA handshake and derive the keying material of the new session from both sources. TLS 1.3 [35], for example, supports combining pre-shared key and certificate-based authentication to increase the security of a session [22]. The derived keying material is authentic if either the pre-shared key derived from previous keying material or the keying material produced by the PILA handshake is authentic and no secret values were leaked. Since PILA reduces the attack surface to the endpoints's ASes, authenticated session resumption over different ASes increases the number of ASes that an attacker has to compromise in order to launch a successful undetected MitM attack.

H. Downgrade Prevention

Whenever an initiator communicates with an unknown responder, an attacker might perform a downgrade attack to reduce the security to a less secure protocol (e.g., TOFU protocol). An attacker attempts to convince the initiator that a responder's AS does not support PILA or that the responder does not allow a specific PILA-supported protocol.

AS Downgrade. AS downgrade is prevented by locally keeping a regularly updated list at each AS containing all PILA-enabled ASes with their certificate service addresses. Endpoints then request certificates for a specific AS or all ASes that originate a specific IP address from their local certificate service, which responds with a signed list of the AS certificates.

Endpoint Downgrade. An AS that supports PILA must provide proof that a service at a given IP address does not allow a specific PILA-supported protocol to assure a sending endpoint that its communication is not being downgraded. An endpoint sends a request including a PILA-supported protocol, an IP address, and the current time as a timestamp. The certificate service replies with a signed proof that contains the hash of the request and a (possible empty) list of certificate entries valid at the requested time. A certificate entry consists of the hash of the certificate and its validity period. The endpoint then verifies the signature and that the returned list is empty before falling back to a non-PILA protocol.

While these approaches for both the AS and endpoint downgrade prevention method work well and are easy to implement, they have a large computational overhead due to the signature operation necessary to create each proof. A more elaborate approach that scales better to a large number of requests is organizing AS and endpoint certificates in public append-only logs as in certificate transparency. The AS certificate log must provide a globally consistent view of all AS certificates, while the endpoint certificate log can be implemented as a separate log per AS. Each log is accompanied by a verifiable log-backed map [15], which provides a verifiable key-value store that can efficiently derive proofs of presence for a specific key-value mapping and proofs of absence for non-existing keys. The log and the log-backed map only require one signature operation per maximum merge delay (MMD) regardless of the number of requests. The log-backed maps allow endpoints to fetch an AS certificate for an AS number and a list of certificate entries from an $\langle$ IP-address, protocol $\rangle$ tuple.

V. Security Analysis

The goal of PILA is to provide an initiator with an authentic X.509 certificate of a responder, in the presence of an attacker that can intercept, reorder, modify, and create arbitrary packets. The underlying protocol uses this certificate to derive an authentic key between the initiator and responder (session-establishment protocol) or to verify the correctness of a message signed by the responder (query–response protocol). PILA provides the initiator with an authentic certificate if the responder's AS is honest or a CuBC attacker and the initiator, responder, and global trust anchors are benign and none of these entities are compromised. In mutual authentication, both endpoints act as responders. The goal of an attacker is to convince the initiator to accept a forged certificate by performing a MitM attack, by downgrading to a non-PILA connection, or by compromising a private key of a certificate in the certificate chain. Additionally, we analyze attacks on AS trust and denial-of-service (DoS) attacks.

MitM Attack. An attacker can perform a MitM attack to impersonate an endpoint by providing a forged certificate to the initiator. For protocols that establish secure sessions, this is done by intercepting the handshake messages and simultaneously creating two separate connections with the initiator and responder. For query–response protocols, the attacker modifies the response and possibly the signature within the response. If the endpoints resume sessions as described in §IV-G, an

attacker has to perform the attack on every session resumption to be successful and stay undetected.

Local Responder-Side Attacker. Attackers in the responder's local network are easily detectable, since the responder can query either the certificate service or the local NAT, see §IV-F, and check for duplicate certificates for its identifiers.

Responder-Side NAT or AS Attacker. A malicious AS or a malicious NAT device on the responder side cannot immediately be detected. They do however create irrefutable cryptographic proof of misbehavior in the form of conflicting endpoint certificates valid at the same point in time. These certificates can be stored locally or published on an append-only log server and later be compared through an out-of-band channel or audited by another entity.

Other Attackers. Other entities, such as a malicious AS or NAT device on the initiator's side or an attacker in the initiator's local network, cannot perform a MitM attack since they cannot forge valid responder certificates.

Downgrade Attacks. An attacker may attempt to convince the initiator connecting to an unknown responder that the responder's AS does not support PILA or that the responder does not allow the PILA-supported protocol. These attacks are prevented due to the mechanisms explained in §IV-H.

Private Key Compromise. The severity of a compromised private key depends on the entity and the lifetime of the certificate belonging to this key. Key compromises of entities in the RPKI delegation chain are relatively easy to detect if abused, since there would be ASes with multiple valid certificates with different public keys, different AS numbers, or different IP address ranges. AS key compromises are similarly easy to detect but only allow forging signed PILA messages within the compromised AS. Endpoint key compromises are less severe, as endpoint certificates are short-lived.

Attacking AS Trust. Attackers might attempt to reduce the trustworthiness of ASes. Slander, i.e., accusing a benign, uncompromised AS of having issued incorrect certificates, is not possible in PILA since an attacker does not possess the AS's private key and thus cannot forge certificates. An attacker might try to frame an AS by requesting incorrect certificates. Incorrect certificates could be certificates for IP addresses already assigned by the AS or not within the control of the AS or certificates with another endpoint's local identifier. ASes prevent such framing attacks by verifying the correctness of an endpoint's claim of identifiers as explained in §IV-D.

Resource-Exhaustion Attacks. Resource-exhaustion attacks attempt to overwhelm computational, storage, or network resources of an endpoint or certificate service. An AS deploying a certificate service can perform ingress filtering to limit external DoS attacks and locate endpoints performing DoS attacks in its own network. Endpoints can deploy typical DoS countermeasures for transport or application layer protocols, for example, DNS cookies [14].

VI. Use Cases

We present three use cases for PILA, which cover a remote login protocol (SSH), a query–response protocol (DNS), and a general session-establishment protocol (TLS).

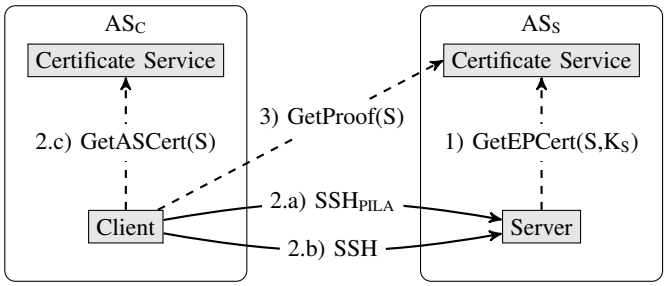

Fig. 3: SSH_{PILA} connection establishment.

A. SSH

The SSH protocol allows clients to establish an encrypted and possibly authenticated session and open a terminal on a remote machine. The SSH protocol is frequently used to connect to machines identified by their IP address and thus is well suited to use PILA as a baseline for security. After initially connecting to a remote machine, SSH associates the machine's address and its public key, and thus all subsequent connections are authentic if the initial connection was authentic. If a client either pre-loads these associations on their machine or verifies the fingerprint (hash of the remote machine's public key) then the authenticity of the initial connection is guaranteed. In many cases, a client simply accepts the provided fingerprint or fails to detect a difference between two long hexadecimal fingerprints [19], [12]. In these cases, PILA mitigates attacks by providing an association between IP addresses and public keys of hosts in PILA-enabled ASes when connecting to a remote machine from a client for the first time.

In SSH_{PILA}, instead of directly authenticating a server by its public key, servers are authenticated by their endpoint certificates. This requires a slight change in the SSH handshake message format. Instead of adding its public key to the final SSH handshake message, a server adds its endpoint certificate which contains the public key used during the SSH handshake.

Figure 3 shows the SSH_{PILA} connection establishment with dashed lines indicating situational or periodic operations. The server periodically requests an endpoint certificate from its AS's certificate service (1) and includes it in the final handshake message. The client initiates an SSH_{PILA} handshake (2.a) which might be dropped by a non-SSH_{PILA} server. Concurrently to this handshake, to reduce the latency, the client initiates a regular SSH handshake in case the server does not support SSH_{PILA} (2.b) and fetches the server's AS certificate if it is not cached (2.c). When the final handshake message is received, the client validates the received endpoint certificate using its RPKI trust anchors and the server's AS certificate. If the SSH_{PILA} handshake fails but the server's AS supports PILA, the client additionally requests an explicit proof that the server does not support SSH_{PILA} from the server's AS (3). It is important to note that the client does not complete the regular handshake until the explicit proof is received to prevent leaking information such as login credentials to a MitM attacker.

Query	Question	A	example.com
	Additional	OPT	EDNS(0)
		TXT	freshness

Reply	Question	A	example.com
	Answer	A	X.X.X.X
	Additional	OPT	EDNS(0)
		TXT	Certificates
		SIG	Signature

Fig. 4: DNS messages exchanged between client and its recursive DNS_{PILA} resolver. The columns depict the DNS query section, resource record type, and resource record value.

B. DNS

In a large-scale survey by Chung et al. in 2017 [7], 88% of all DNSSEC-enabled recursive DNS resolvers returned supposedly DNSSEC-verified responses, without actually verifying the certificate chain. If the certificate chain from the DNS root certificate is not verified before an entry is cached, then DNSSEC does not provide any security and is vulnerable to the same attacks as regular DNS, while providing a false sense of security. The issue seems to be that the resolver is not easily held accountable for the validity of the returned DNSSEC entries. DNS_{PILA} solves this issue by holding resolvers accountable for their DNSSEC responses. Auditors can use the signed DNS_{PILA} responses to prove recursive DNS resolvers's misbehavior (serving unverified DNSSEC-enabled responses) by verifying the DNSSEC responses themselves.

DNS_{PILA}. DNS_{PILA} adds freshness to DNS queries and returns signed DNS replies including necessary certificates as shown in Fig. 4. DNS is a prime example of the benefits of PILA as it is (1) unauthenticated, (2) interception and redirection of requests is widespread [28], and (3) DNS servers are mostly identified by their IP addresses. It is important to note that DNS_{PILA} operates between the client and resolver, unlike DNSSEC, where authoritative nameservers publish DNSSEC entries which are distributed by resolvers. In comparison to DNS over HTTPS (DoH) and DNS over TLS (DoT) which provide secrecy and authenticity, DNS_{PILA} provides authenticity and non-repudiation. DNS_{PILA} is thus to some extent orthogonal to DoH and DoT and could be encapsulated within DoH or DoT to additionally provide secrecy.

A client adds the following resource records (RR) to the DNS query: An EDNS(0) RR_{OPT} [11] to allow payloads larger than 512 B for transmitting the necessary certificates and a TXT record (RR_{TXT}) containing a random value to prevent replay attacks. The server detects PILA support by checking for a DNS_{PILA} RR_{TXT}. If DNS_{PILA} authentication is requested, the server adds an RR_{TXT} with the required certificate (chain) to the response. The server then adds the signature which is calculated over both the query and response in the form of an RR_{SIG} record analogous to a SIG(0) [13] signature. The client checks each field in the response and verifies the RR_{SIG} using the endpoint certificate.

Privacy. There is a privacy concern for disclosing malicious DNS servers as it reveals the browsing behavior of a user. A way to circumvent the privacy implications is by sending a denouncing DNS_{PILA} response to an auditor in a privacy-preserving way, e.g., via TOR.

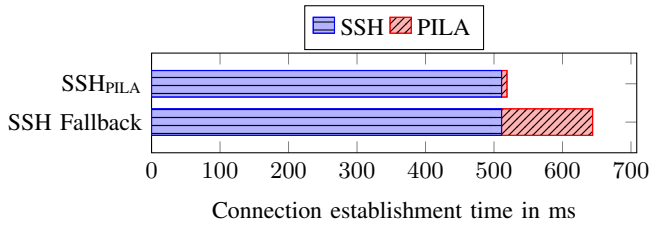

Fig. 5: SSH connection establishment time in ms for successful SSH_{PILA} connections and connections that fall back to regular SSH. The latency is split into the baseline latency of a regular SSH connection establishment and the overhead of SSH_{PILA}.

C. TLS

We define PILA for TLS (TLS_{PILA}) as an example of a secure session-establishment protocol. Our goal is to achieve a baseline of security for persistent connections, which requires authenticating services by default, regardless of whether the endpoint is identified by an IP address or a DNS name. An endpoint uses TLS_{PILA} if neither a `TLSA` resource record nor a Web PKI certificate is available and the destination endpoint is identified via IP address.

In TLS_{PILA}, TLS 1.3 is modified to use PILA endpoint certificates instead of certificates signed by the Web PKI and verify the IP address instead of the domain name of the opposite endpoint. Apart from the handshake, TLS_{PILA} is analogous to SSH_{PILA} except that since TLS requires a certificate, there is no fallback mechanism and thus no `GetProof` request.

VII. Evaluation

We created proof-of-concept implementations for SSH_{PILA}, TLS_{PILA}, and DNS_{PILA} and evaluate their latency, memory, and processing overhead.

A. SSH_{PILA} and TLS_{PILA}

SSH_{PILA} is implemented in golang and the client conforms to the interface in the golang SSH library [18] but provides PILA-specific configuration options. X.509 resource certificates are processed using the Cloudflare RPKI Validator Tools and Libraries [8] and the server implementation is based on the Glider Labs SSH library [25]. The implementation uses the signed list approach to provide AS and endpoint downgrade protection as described in §IV-H and allows both challenge–response and signature-based endpoint-certificate issuance as described in §IV-E. All measurements use the signature-based endpoint-certificate issuance unless stated otherwise. We simulate the RPKI environment by creating a self-signed RPKI trust anchor and delegation chain certificates. The AS certificates, analogous to RPKI certificates, use 2048 bit RSA public keys, while endpoint certificates use 256 bit ed25519 public keys to reduce the size of certificates and signatures. We evaluate the end-to-end latency in a realistic setting using virtual machines located in two data centers. The client and server together with their certificate service are located in New York and Frankfurt, respectively. This leads to a 1 ms

TABLE I: Average processing times of SSH_{PILA} operations in ms at the client, server, and certificate service (CS).

	Client	Server	CS
Handshake Overhead	0.8	0.1	-
`GetEPCert`	-	1.0	17.0
`GetASCert`	4.3	-	8.3
`GetProof`	0.6	-	5.1

and 84 ms round-trip time (RTT) within and between the data centers, respectively.

Figure 5 shows that the increase in SSH connection-establishment time is less than 2% for successful SSH_{PILA} connections. If the client connects to a server that does not support SSH_{PILA} and must request a proof thereof, the overhead is slightly larger (∼26%). For most users, such latency overheads during the infrequent SSH connection establishments are negligible. The memory overhead compared to regular SSH is below 620 kB.

Table I shows the processing overhead of the SSH_{PILA} handshake compared to a regular SSH handshake and of requests from the endpoint to their own AS (`Get*Cert`) and the destination AS (`GetProof`). The measurements for the `Get*` requests to the certificate service exclude the processing time of the network stack. The handshake overhead is less than 1 ms for both endpoints, which contributes only a fraction of the experienced latency observed in the end-to-end evaluation where network delays dominate the overhead. The processing overhead for a client is small and does not pose an issue for most clients, which only initiate few SSH connections.

On the server side, the overhead of requesting an endpoint certificate is low and its amortized cost is further reduced by only fetching the endpoint certificate once. The only additional processing required during the actual handshake is replacing the public key in the final handshake message with the endpoint certificate, increasing the size of the handshake message by ∼ 600 B. Hence SSH_{PILA} incurs virtually no overhead compared to regular SSH at the server. For the certificate service, the overhead per request is less than 10 ms except for issuing the endpoint certificate, which is an infrequent operation done once per certificate lifetime per endpoint. The load on the certificate service could be reduced further by caching AS and endpoint certificates and proofs at the client.

For `GetEPCert`, we evaluate the signature-based endpoint-authentication approach. The challenge–response protocol adds a small additional delay of two RTTs for the TCP and HTTP request and small processing overhead for setting up a minimal HTTP server on the client. This increases the `GetEPCert` latency by < 4 ms.

For the TLS_{PILA} implementation, the overhead of a handshake is less than 700 µs on the client side and negligible on the server side. Apart from the handshake, the remaining overhead is the same as in SSH_{PILA} except that `GetProof` is not used.

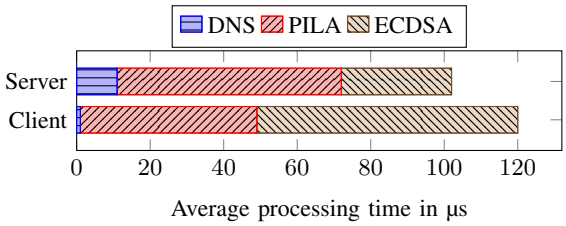

Fig. 6: Average processing times for signing (server) and verifying (client) DNS_{PILA} responses excluding the network stack. The processing is split into a general processing of DNS responses (de- and encoding, domain lookup), PILA-related processing excluding cryptographic operation, and ECDSA P-256 signature creation (server) and verification (client).

B. *DNS_{PILA}*

We implement DNS_{PILA} on top of the well known DNS library miekg [16]. In particular, we look at the performance of both a DNS_{PILA} client and server, as we want to ensure the overhead of DNS_{PILA} is acceptable even for resource-constrained devices. Our implementation uses ECDSA P-256 as a signature algorithm, which is used for SIG(0) [13] resource record in the DNS library. Figure 6 shows the overhead of our implementation for generating the DNS_{PILA} responses at the server and verifying them at the client. Using a typical workstation processor, a DNS_{PILA} response is created and signed in 102 µs and verified in 119 µs.

For clients, even processing hundreds of responses leads to a small overhead of tens of milliseconds. We expect that in a more efficient implementation, which prevents unnecessary memory allocations and packet copying, the cryptographic operations would become the processing bottleneck. Servers employing DNS_{PILA} achieve high packet rates of $\sim 10^4$ pkt/s on a single core. By parallelizing signing operations on multiple cores, the throughput can be increased further.

VIII. Related Work

Anonymous key-exchange protocols do not rely on trusted entities but authenticity is only guaranteed if the initial message(s) were not tampered with, a principle called trust-on-first-use (TOFU). Examples are *TCPCrypt* [5], *Secure Shell (SSH)* (without verifying the authenticity of the public key), and *SMKEX* [9] which uses multiple network paths to impede tampering with the handshake.

Certificate-based protocols rely on entities as trust anchors to authenticate endpoints using cryptographic certificates. PILA belongs to this category since endpoints verify the certificate chain from the RPKI trust anchor to the endpoint certificates. The most widely used certificate-based protocol is HTTPS, which uses TLS in combination with the Web PKI, especially with the rise of *Let's Encrypt* [24] providing automated and free creation TLS server certificates. *DANE* [21] fixes the weakest-link security of the Web PKI by distributing DNSSEC-authenticated TLSA DNS records that allow domain owners to specify Web PKI policies.

Anonymity-based protocols exploit the fact that an attacker cannot distinguish packets sent in an anonymity network infrastructure with different sources or destinations. *DoubleCheck* [1] retrieves self-signed certificates of the receiving endpoint through different routes within the anonymity network and detects MitM attacks if the attacker cannot intercept all requests. In *Plug-and-Play IP Security* [17], a user detects that an attacker is tampering with its traffic by periodically sending packets to itself through the anonymity network.

PISKES [36], intended for DoS prevention, allows an endpoint to efficiently derive a symmetric key to any other endpoint in the Internet, which can then be used for first-packet authentication. Similar to PILA, endpoints are identified by their IP address and each participating AS provides a service to fetch these keys. However, unlike PILA, PISKES can only generate symmetric keys and ASes cannot be held accountable to third parties for their actions. Thus, PISKES only provides the first property, *crude authentication*, of trust amplification.

IX. Conclusion

PILA enables ubiquitous authentication for a wide range of devices and provides significantly improved security properties compared to TOFU approaches. This is achieved through the trust-amplification model in combination with RPKI and IP-address-based authentication. PILA can be used for session-establishment (SSH and TLS) and query–response protocols (DNS). Our proof-of-concept implementations show that PILA can be implemented without significant performance cost with an end-to-end latency increase of less than 2% for a regular connection and less than 26% for a fallback connection in the case of SSH_{PILA} and a per-packet overhead of $\sim$100 µs for DNS_{PILA}. We believe that PILA can form a solid security baseline for an abundance of applications in the absence of strong authentication options.

X. Acknowledgments

We gratefully acknowledge support from ETH, ZISC, and the European Union's Horizon 2020 research and innovation programme under grant agreements No 825310 and 825322.

References

[1] M. Alicherry and A. D. Keromytis. DoubleCheck: Multi-path verification against man-in-the-middle attacks. In *IEEE Symposium on Computers and Communications*, 2009.

[2] R. Barnes. Use Cases and Requirements for DNS-Based Authentication of Named Entities (DANE). RFC 6394, 2011.

[3] R. Barnes, J. Hoffman-Andrews, D. McCarney, and J. Kasten. Automatic Certificate Management Environment (ACME). RFC 8555, 2019.

[4] T. Bates. CIDR report. www.cidr-report.org/as2.0/, March 2021.

[5] A. Bittau, M. Hamburg, M. Handley, D. Mazières, and D. Boneh. The case for ubiquitous transport-level encryption. In *USENIX Security Symposium*, 2010.

[6] T. Chung, E. Aben, T. Bruijnzeels, B. Chandrasekaran, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, R. v. Rijswijk-Deij, J. Rula, and N. Sullivan. RPKI is coming of age: A longitudinal study of RPKI deployment and invalid route origins. In *Internet Measurement Conference*, 2019.

[7] T. Chung, R. Van Rijswijk-Deij, B. Chandrasekaran, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, and C. Wilson. A Longitudinal, End-to-End View of the DNSSEC Ecosystem. In *USENIX Security Symposium*, 2017.

[8] Cloudflare. Cloudflare RPKI validator tools and libraries. https://github.com/cloudflare/cfrpki.

[9] S. Costea, M. O. Choudary, D. Gucea, B. Tackmann, and C. Raiciu. Secure opportunistic multipath key exchange. In *ACM SIGSAC Conference on Computer and Communications Security*, 2018.

[10] J. Cowie. The new threat: Targeted Internet traffic misdirection. https://dyn.com/blog/mitm-internet-hijacking, 2013.

[11] J. Damas, M. Graff, and P. Vixie. Extension Mechanisms for DNS (EDNS(0)). RFC 6891, 2013.

[12] S. Dechand, D. Schürmann, K. Busse, Y. Acar, S. Fahl, and M. Smith. An empirical study of textual key-fingerprint representations. In *USENIX Security Symposium*, 2016.

[13] D. Eastlake, III. DNS Request and Transaction Signatures (SIG(0)s). RFC 2931, 2000.

[14] D. Eastlake, III and M. Andrews. Domain Name System (DNS) Cookies. RFC 7873, 2016.

[15] A. Eijdenberg, B. Laurie, and A. Cutter. Verifiable data structures. https://github.com/google/trillian/blob/master/docs/papers/VerifiableDataStructures.pdf, 2015.

[16] M. Gieben. DNS library in Go. https://github.com/miekg/dns.

[17] Y. Gilad and A. Herzberg. Plug-and-Play IP Security: Anonymity Infrastructure Instead of PKI. *ESORICS*, 2013.

[18] Go contributors. Go cryptography. https://github.com/golang/crypto.

[19] P. Gutmann. Do users verify SSH keys? *Login*, 36, 2011.

[20] D. Harkins and W. Kumari. Opportunistic Wireless Encryption. RFC 8110, 2017.

[21] P. Hoffman and J. Schlyter. The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA. RFC 6698, 2012.

[22] R. Housley. TLS 1.3 Extension for Certificate-Based Authentication with an External Pre-Shared Key. RFC 8773, 2020.

[23] G. Huston, G. Michaelson, and R. Loomans. A Profile for X.509 PKIX Resource Certificates. RFC 6487, 2012.

[24] ISRG. Let's Encrypt. https://letsencrypt.org.

[25] G. Labs. Glider labs SSH. https://github.com/gliderlabs/ssh.

[26] B. Laurie, A. Langley, and E. Kasper. Certificate Transparency. RFC 6962, 2013.

[27] M. Lepinski and S. Kent. An Infrastructure to Support Secure Internet Routing. RFC 6480, 2012.

[28] B. Liu, C. Lu, H. Duan, Y. Liu, Z. Li, S. Hao, and M. Yang. Who is answering my queries: Understanding and characterizing interception of the DNS resolution path. In *USENIX Security Symposium*, 2018.

[29] B. Liu, J. T. Chiang, J. J. Haas, and Y.-C. Hu. Coward Attacks in Vehicular Networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, 14(3), 2010.

[30] C. Lynn, S. Kent, and K. Seo. X.509 Extensions for IP Addresses and AS Identifiers. RFC 3779, 2004.

[31] E. MacAskill, J. Borger, N. Hopkins, N. Davies, and J. Ball. GCHQ taps fibre-optic cables for secret access to world's communications. www.theguardian.com/uk/2013/jun/21/gchq-cables-secret-world-communications-nsa, 2013.

[32] NIST. RPKI Monitor. https://rpki-monitor.antd.nist.gov, March 2021.

[33] A. Paverd, A. Martin, and I. Brown. Modelling and Automatically Analysing Privacy Properties for Honest-but-Curious Adversaries. *Tech. Rep.*, 2014.

[34] A. Perrig, P. Szalachowski, R. M. Reischuk, and L. Chuat. *SCION: A Secure Internet Architecture*. Springer Verlag, 2017.

[35] E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, 2018.

[36] B. Rothenberger, D. Roos, M. Legner, and A. Perrig. PISKES: Pragmatic Internet-scale key-establishment system. In *ACM Asia Conference on Computer and Communications Security*, 2020.

[37] R. Shoemaker. Automated Certificate Management Environment (ACME) IP Identifier Validation Extension. RFC 8738, 2020.

[38] D. Wendlandt, D. G. Andersen, and A. Perrig. Perspectives: Improving SSH-style host authentication with multi-path probing. In *Proceedings of the USENIX Annual Technical Conference*, June 2008.

[39] M. Wählisch, R. Schmidt, T. Schmidt, O. Maennel, S. Uhlig, and G. Tyson. RiPKI: The tragic story of RPKI deployment in the web ecosystem. 2015.

[40] T. Ylonen and C. Lonvick. The Secure Shell (SSH) Protocol Architecture. RFC 4251, 2006.